WELCOME

Modern PL/SQL Code Checking and Dependency Analysis

Philipp Salvisberg

5th October 2011

BASEL BERN 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.trivadis.com
- Member of the trivadis performanceteam

- Main focus on database centric development with Oracle DB
  - Application Performance Management
  - Application Development
  - Business Intelligence
- 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

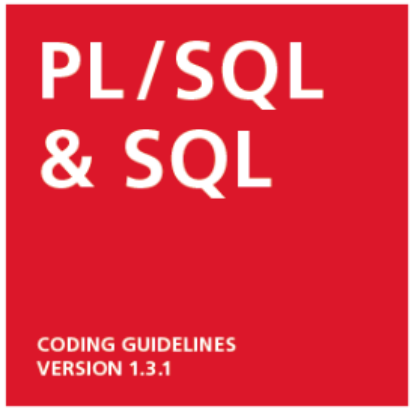
"Roger and his team have done an excellent job of providing a comprehensive set of clear standards that will undoubtedly improve the quality of your code. If you do not yet have standards in place, you should give strong consideration to using these as a starting point."

Steven Feuerstein
PL/SQL Evangelist

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

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

Trivadis PL/SQL & SQL Guideline #25



25. 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

See http://www.trivadis.com/technologie/swiss-it-up/plsql-assessment.html

trivadis makes IT easier.

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

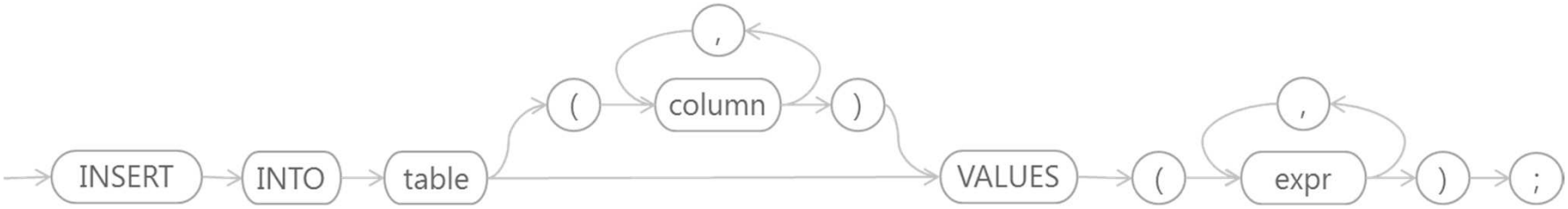
- Eclipse-based Editors
  - Validation and Quick Fixes
  - Syntax Coloring
  - Code Completion
  - Outline View
  - Code Formatting
  - Bracket Matching
- Integration
  - Eclipse Modeling Framework (e.g. for graphical editors)
  - Eclipse Workbench (e.g. for list of problems/warnings)
  - Export into 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::=



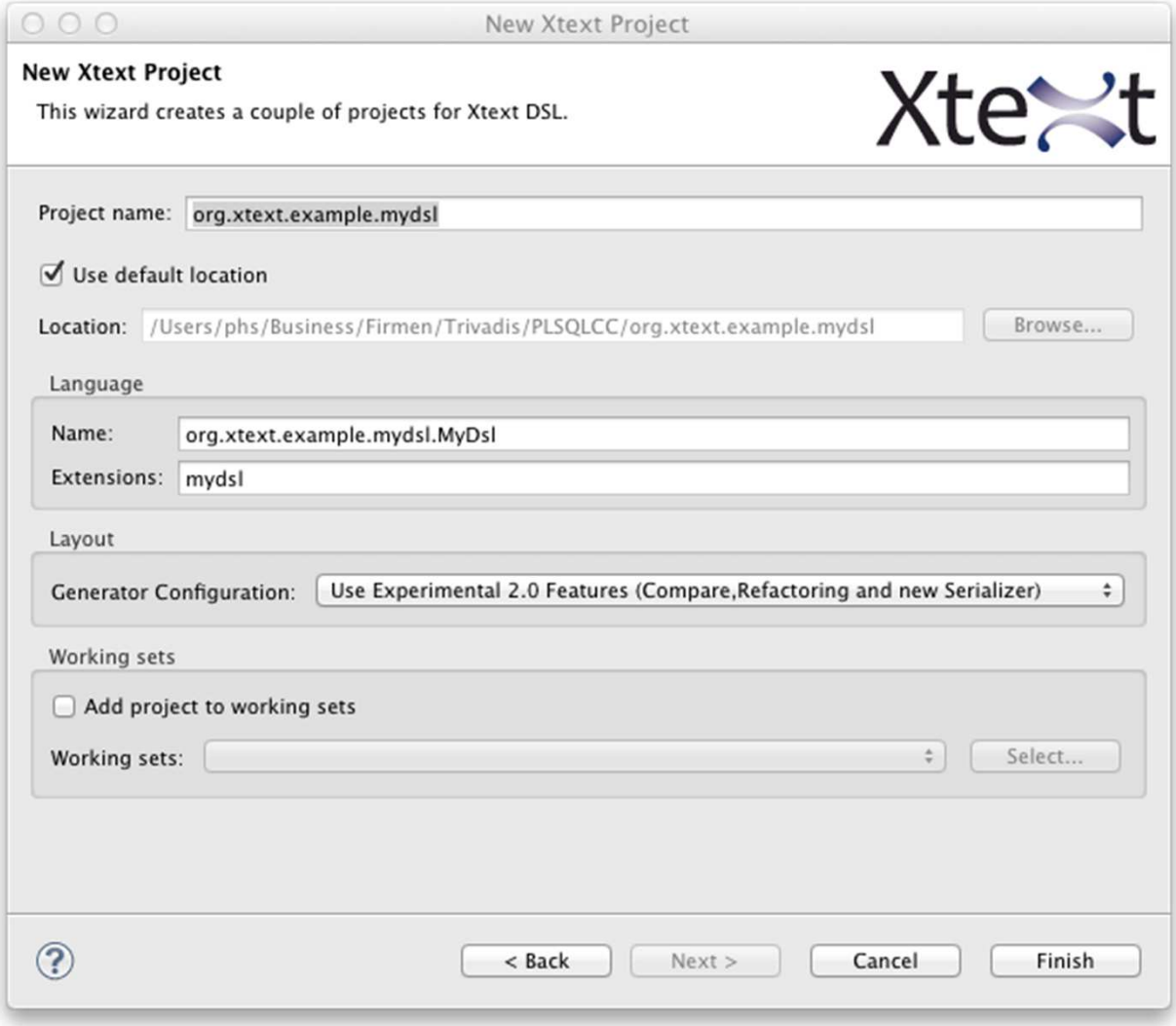

```
INSERT INTO table [( column [, column ]... )]
VALUES ( expr [, expr ]... ) ;
```

plsql_unit::=


```
BEGIN insert_statement END ;
```

Default Xtext Project

DEMO



trivadis
makes IT easier.

Demo Grammar (Xtext)

DEMO

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

MyDsl.xtext

```
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
;
```

Writable | Insert

trivadis makes IT easier.

Eclipse Editors

DEMO

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

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 values (1,2,3);
end;

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

Outline

- <unnamed>
  - t1
  - <unnamed>
    - t2
  - t3

Problems | Javadoc | Declaration | Properties

Property	Value

Writable | Insert | 4 : 6

trivadis makes IT easier.

Validator for Guideline #25

DEMO

Java – org.xtext.example.mydsl/src/org/xtext/example/mydsl/validation/MyDslJavaValidator.java – Eclipse SDK – /Users/phs/B...

MyDsl.xtext | MyDslJavaValidator.java

```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 checkGuideline25(InsertStatement insertStatement) {
        if (insertStatement.getColumns().isEmpty()) {
            warning("Guideline 25 violated: Always specify the target columns when executing an insert command.",
                    MyDslPackage.Literals.INSERT_STATEMENT__TABLE_NAME);
        }
    }
}
```

Writable | Smart Insert | 17 : 1

trivadis makes IT easier.

Validator in Action

DEMO

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

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 values (1,2,3);
end;

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

Guideline 25 violated: Always specify the target columns when executing an insert command.

Outline
- <unnamed>
  - t1
  - <unnamed>
    - t2
  - t3

Problems | Javadoc | Declaration | Properties

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (2 items)				
Guideline 25 violated: Always specify the target columns when...	test.mydsl	/test/src	line: 4 /test/...	Xtext Check (...
Guideline 25 violated: Always specify the target columns when...	test.mydsl	/test/src	line: 7 /test/...	Xtext Check (...

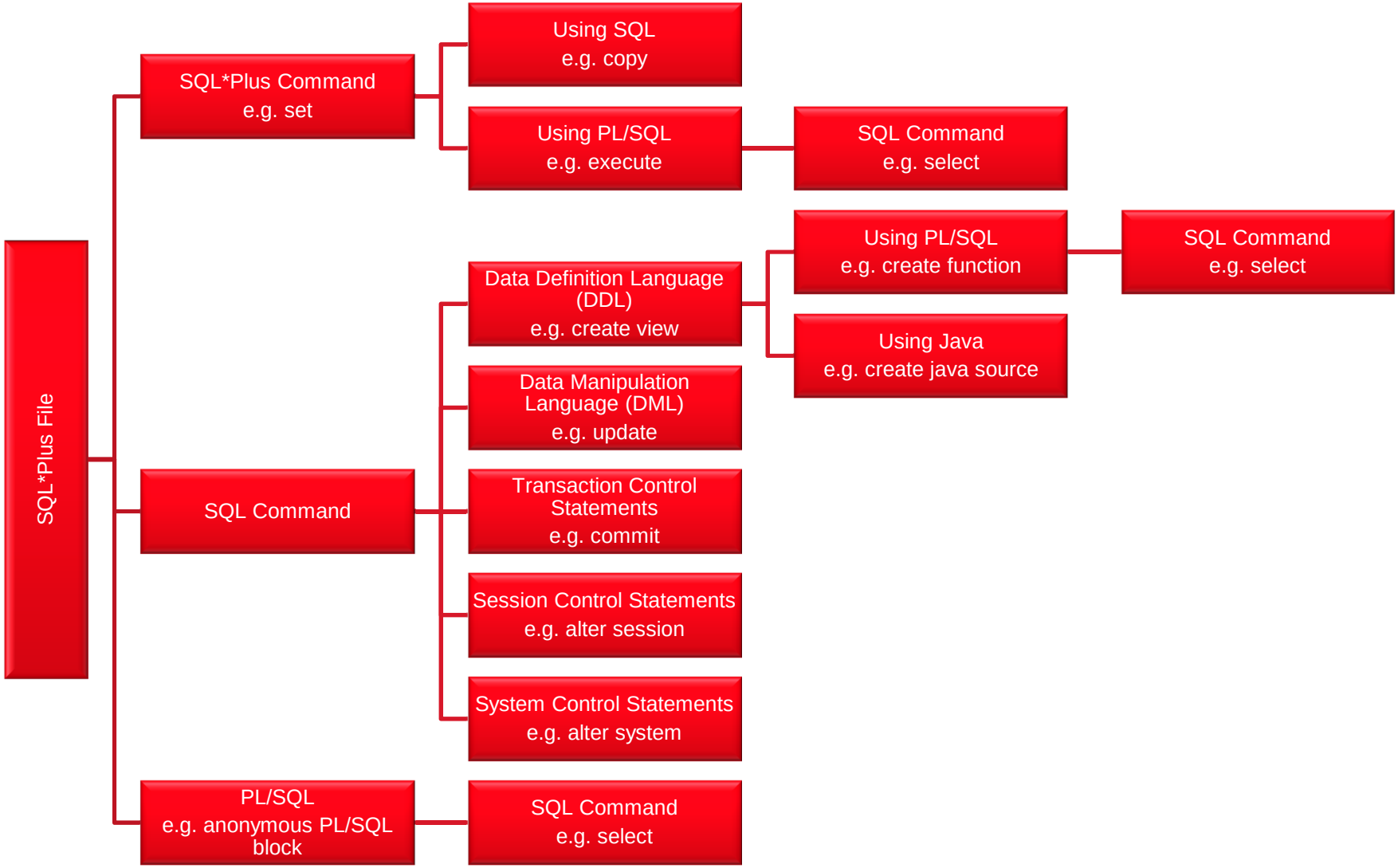
Guideline... command. | Writable | Insert | 7 : 14

trivadis makes IT easier.

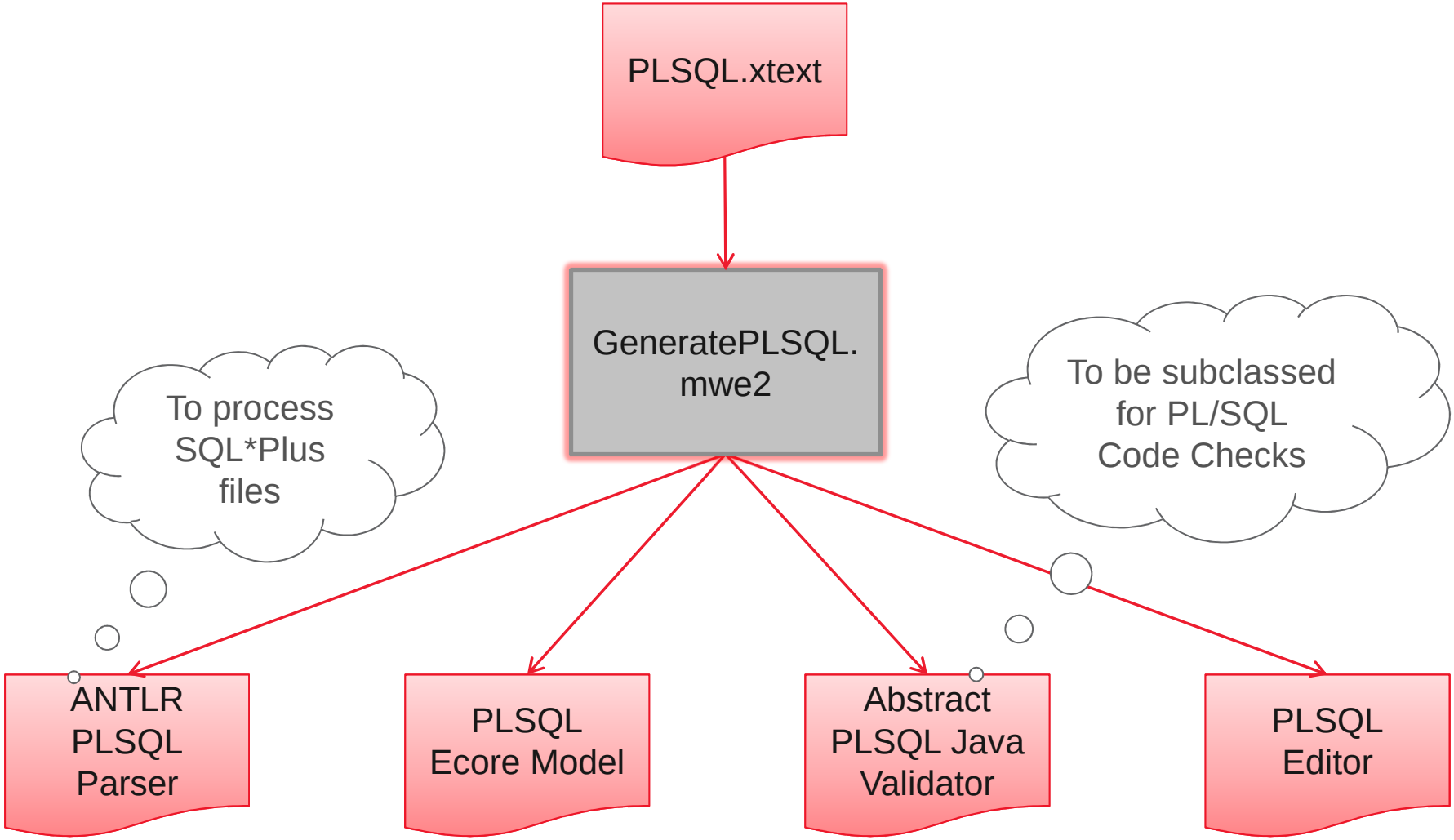
AGENDA

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

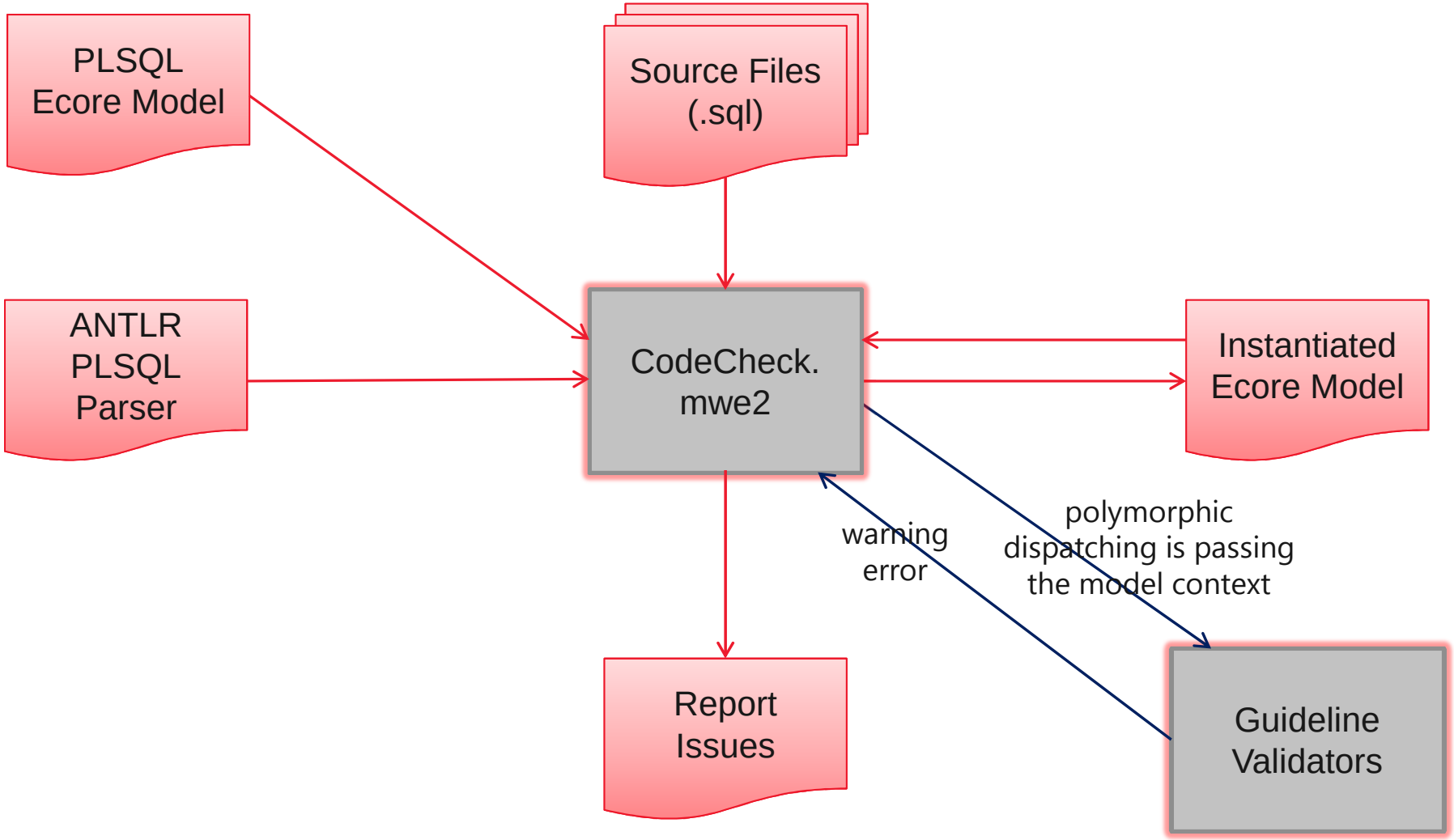
Content of a SQL*Plus File



Generate PL/SQL Grammar via Xtext



Apply Code Checks (via Command Line)



Source, Model & Warning for Guideline #25

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

line 2 - Guideline 25 violated: Always specify the target columns when executing an insert command.

Generic Editor - guideline_25.sql Guideline ...t command.

Model

- platform:/resource/test/src/guideline_25.sql
  - PLSQL File
    - Plsql Block
      - Body
        - Insert Statement
          - Single Table Insert
            - Insert Into Clause
              - Dml Table Expression Clause
            - Values Clause
              - Function Or 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 Nextval Value
                - Function Or Parenthesis Parameter
                - Simple Expression Name Value p_mesg_type
                - Function Or Parenthesis Parameter
                - Simple Expression Name Value p_mesg_name
                - Function Or Parenthesis Parameter
                - Simple Expression Name Value p_mesg_text
                - Function Parameter List Suffix
        - Run Command

Excerpt of Grammar for Insert Statement

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

InsertPlusHintsAndComments returns InsertStatement hidden(WS, NL/*, SL_COMMENT, ML_COMMENT, CONTINUE_LINE*/):
    {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 #25

```java
@Check
public void checkGuideline25(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 FunctionOrParenthesisExpression)) {
                        // must be a record variable
                        ignore = true;
                    }
                }
            }
            if (!ignore) {
                warning("Guideline 25 violated: Always specify the target columns when executing an insert command.",
                        intoClause.getDmlExpressionClause(), null,
                        GUIDELINE_25, serialize(NodeModelUtils.getNode(insert)
                                .getParent()));
            }
        }
    }
}
```

```oracle
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;
/
```

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:

/Users/phs/Business/Firmen/Trivadis/OracleWorld_20111005/demo/codecheck.jar

Browse...

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:

Browse...

< Back | Next > | Cancel | Finish

Command Line Interface

DEMO

```
$ java -jar codecheck.jar path=sql
Parsing and validating code...

Issues for file 'guideline_25.sql':
  line     2 - Guideline 25 violated: Always specify the target columns when executing an insert command.
1 issue found.
Issues for file 'guideline_47.sql':
  line     5 - Guideline 47 violated: Never handle unnamed exceptions using the error number.
1 issue found.
Issues for file 'guideline_54.sql':
  line     4 - Guideline 54 violated: Always use a string variable to execute dynamic SQL.
1 issue found.

... 3 issues found in 3 files (within 1.847 seconds).
```

Console Strategy

guideline_25.sql - 1 issue:

line 2 - Guideline 25 violated: Always specify the target columns when executing an insert command.

```
INSERT INTO app_messages
VALUES
(mesg_seq.nextval,
p_mesg_type,
p_mesg_name,
p_mesg_text)
```

guideline_47.sql - 1 issue:

line 5 - Guideline 47 violated: Never handle unnamed exceptions using the error number.

```
when others then
if sqlcode = -1 then
null;
end if;
```

guideline_54.sql - 1 issue:

line 4 - Guideline 54 violated: Always use a string variable to execute dynamic SQL.

```
execute immediate 'select mesg_seq.nextval from dual' into l_next_val
```

HTML Strategy

AGENDA

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

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;
```

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;
```

unit_cost is not part of a column expression

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

Dependency Analysis

DEMO

```java
private static void process(String refOwner, String refName,
        String... refColumn) throws Exception {
    level++;
    printQuerying(refOwner, refName, refColumn);
    // need Locally defined prepared statement to handle recursive calls
    PreparedStatement prepStmt;
    String query = "SELECT d.owner, d.name, dbms_metadata.get_ddl(d.type, d.name, d.owner)||';' as view_ddl "
            + "FROM dba_dependencies d INNER JOIN dba_views v ON v.OWNER = d.owner AND v.VIEW_NAME = d.name "
            + "WHERE d.referenced_owner = ? AND d.referenced_name = ? AND d.type = 'VIEW'";
    prepStmt = conn.prepareStatement(query);

    // Set binds
    prepStmt.setString(1, refOwner);
    prepStmt.setString(2, refName);

    // Query Oracle Dictionary
    ResultSet rs = prepStmt.executeQuery();
    while (rs.next()) {
        // instantiate and parse view
        View view = new View(rs.getString(1), rs.getString(2),
                rs.getClob(3));
        processView(view, refOwner, refName, refColumn);
    }
    prepStmt.close();
    rs.close();
    level--;
}
```

```
querying dba_dependencies for views using SH.COSTS...
analyzing column expressions of SH.PROFITS to find use of UNIT_COST...
*** found usage in column TOTAL_COST ***
*** found usage in column UNIT_COST ***
   querying dba_dependencies for views using SH.PROFITS...
   analyzing column expressions of SH.GROSS_MARGINS to find use of TOTAL_COST, UNIT_COST...
   *** found usage in column GROSS_MARGIN_PERCENT ***
   *** found usage in column GROSS_MARGIN ***
      querying dba_dependencies for views using SH.GROSS_MARGINS...
      analyzing column expressions of SH.REVENUES to find use of GROSS_MARGIN_PERCENT, GROSS_MARGIN...
         querying dba_dependencies for views using SH.REVENUES...
   analyzing column expressions of SH.SALES_ORDERED_BY_GM to find use of TOTAL_COST, UNIT_COST...
      querying dba_dependencies for views using SH.SALES_ORDERED_BY_GM...
```

AGENDA

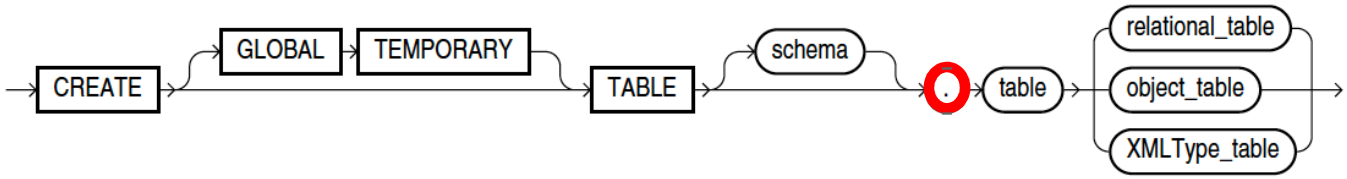
1. Introduction
2. Xtext Live – Parsing & Validating
3. Finalize Grammar, Checks and Tooling
4. Dependency Analysis
5. Challenges
6. Conclusion

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

- Unquoted Identifiers may conflict with keywords of other grammars
  - "describe" is a keyword, but not a reserved word in SQL (valid for table etc.)
  - Abbreviatory notation of SQL*Plus, e.g. run command (r | ru | run)
- 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
- Reduced grammar in the area of less interesting statements
  - AlterTable: 'alter' 'table' text=GenericText SqlCmdEnd ;

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

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

Conclusion

PL/SQL & SQL

Tooling

- The grammar to parse SQL*Plus files is huge
  - a solution to reduce/separate the grammars is necessary to make the development process feasible
  - since Xtext 2.0.1 the size restrictions ceased to apply
- 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 the future PL/SQL CodeChecker 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

trivadis
makes IT easier.