

Fighting Bad PL/SQL & SQL with Custom Rules

Philipp Salvisberg
2 October 2026



WARNING

This presentation contains explicit SQL and PL/SQL,
strong opinions on code quality,
and prolonged scenes of Java development.

Viewers will be exposed to Maven, ANTLR parse trees,
and debugging in multiple IDEs.

Some rules may conflict with deeply held coding beliefs.
Side effects may include raised eyebrows,
involuntary refactoring, and an uncontrollable urge
to automate code reviews.

Viewer discretion is advised.
Your code will receive none.

Welcome



Philipp Salvisberg

Grisselbav Founder, CEO, Consultant, Developer

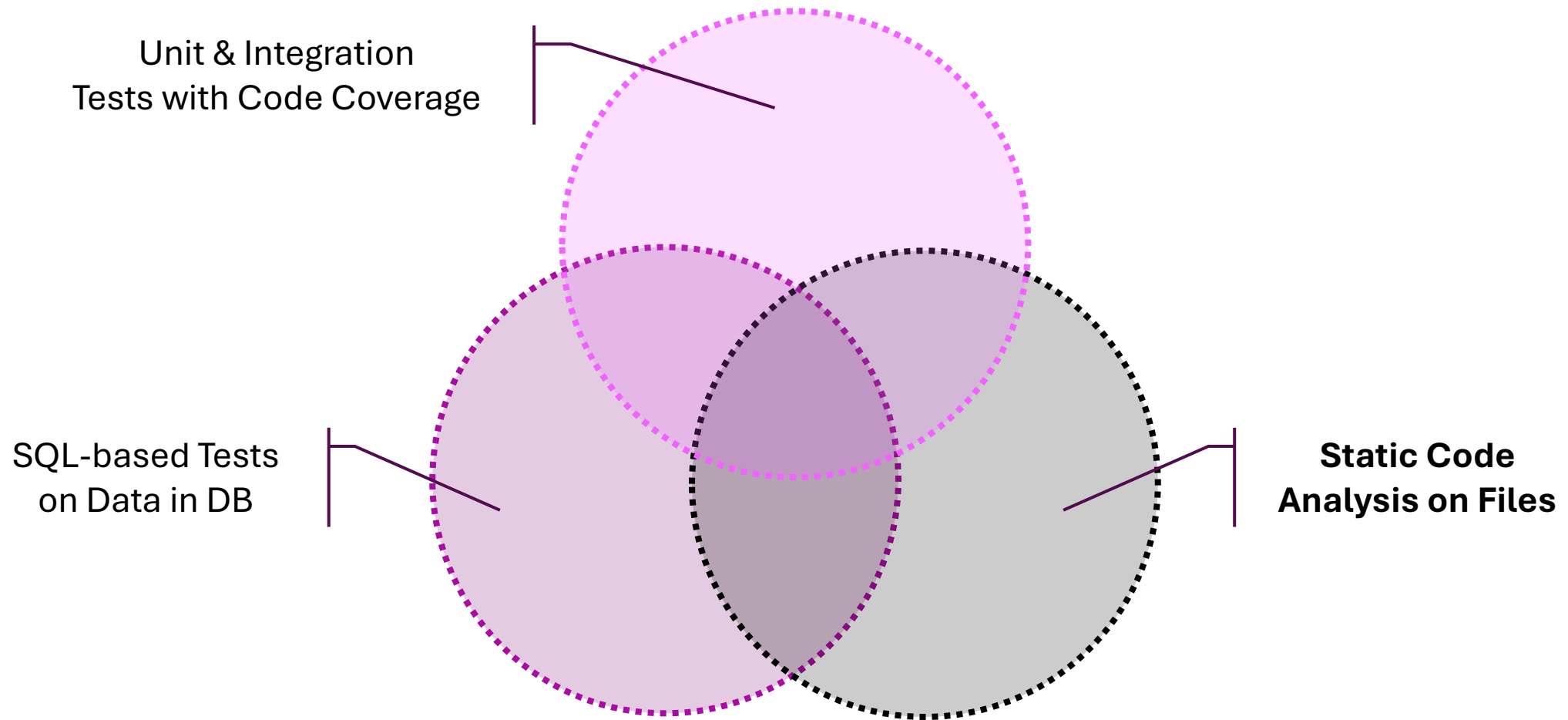
- Database-centric development
- Enforcing high-quality database standards
- Simple & accurate solutions

philipp.salvisberg@grisselbav.com

<https://www.salvis.com/blog/>

Introduction

Quality Assessment of Code in the DB



Rules – The Basis for Static Code Analysis

Core G-3130 FREE

Try to use ANSI SQL-92 join syntax.

SQL • Check ?

Rule Details

Severity Level

 Major ?

Quality Area

Maintainability ?

Fixing times

10 mins
[Quick Fix](#) | Local-high ?

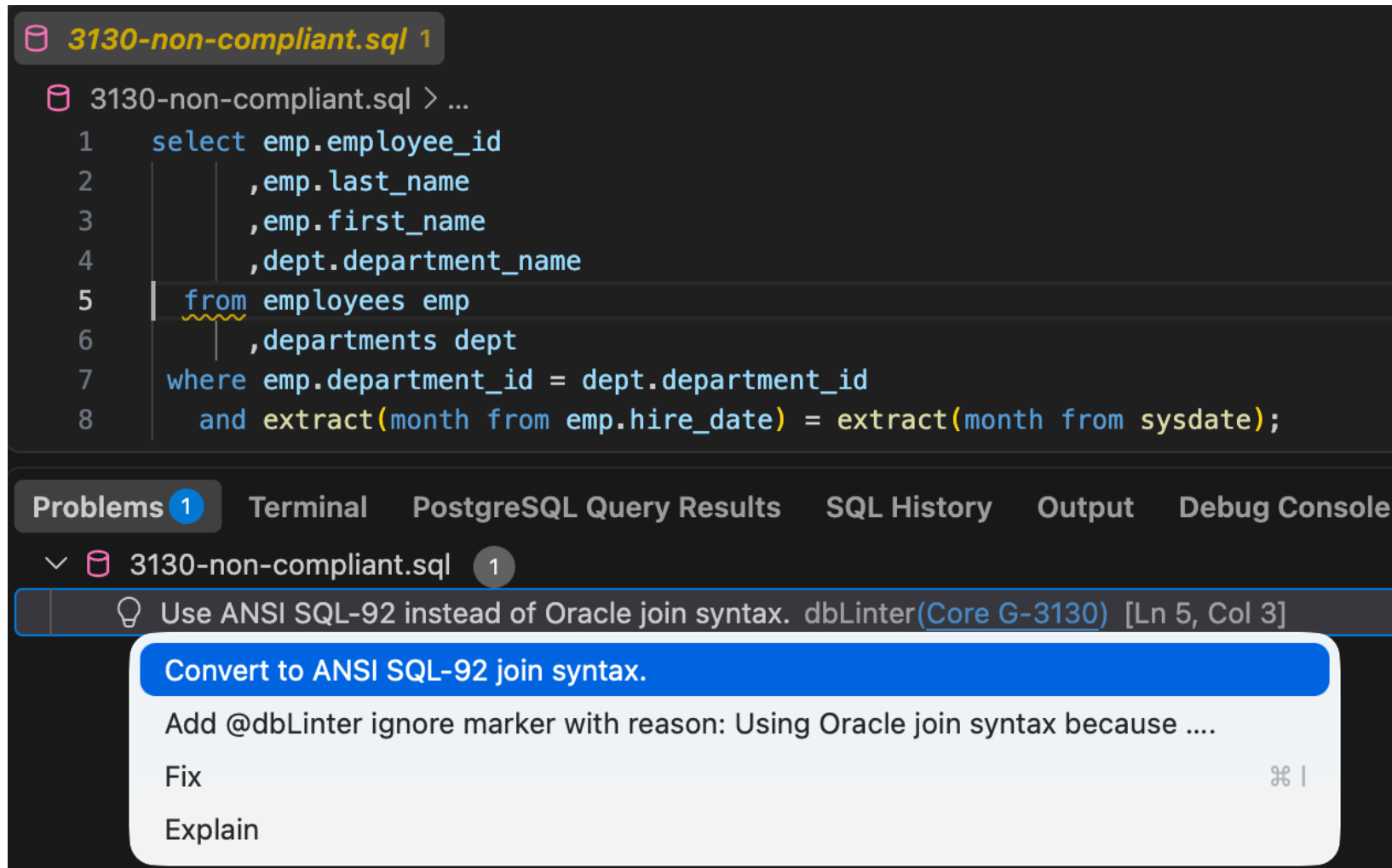
Database Support

OracleDB (v9.0.1 +)
PostgreSQL (v7.0.3 +)

Reason

ANSI SQL-92 join syntax supports the full outer join. A further advantage of the ANSI SQL-92 join syntax is the separation of the join condition from the query filters.

Integration: IDE, CLI, SW Quality Platforms



The screenshot shows an IDE interface with a SQL editor and a Problems panel. The SQL editor contains a query with a linting error on line 5. The Problems panel shows the error message and a suggestion to convert the join syntax to ANSI SQL-92.

```
3130-non-compliant.sql 1
3130-non-compliant.sql > ...
1  select emp.employee_id
2      ,emp.last_name
3      ,emp.first_name
4      ,dept.department_name
5  from employees emp
6      ,departments dept
7  where emp.department_id = dept.department_id
8      and extract(month from emp.hire_date) = extract(month from sysdate);
```

Problems 1 Terminal PostgreSQL Query Results SQL History Output Debug Console

3130-non-compliant.sql 1

💡 Use ANSI SQL-92 instead of Oracle join syntax. dbLinter(Core G-3130) [Ln 5, Col 3]

Convert to ANSI SQL-92 join syntax.

Add @dbLinter ignore marker with reason: Using Oracle join syntax because

Fix

Explain

New Core Or Custom Rule?

Core Rules

- Avoid defining variables that are not used
- Never reference an unknown table/alias in hints
- Never use ROWNUM at the same query level as ORDER BY
- Try to use ANSI SQL-92 join syntax

Custom Rules

- Always restrict rows in views using "tenant_id = current_tenant_id()"
- Always guard soft-delete tables by "deleted_at is null", unless ...
- Never use insert, update, delete, merge outside of PL/SQL packages
- Never use ANSI SQL-92 join syntax

Let's Implement This Custom Rule

Demo R-3131

Never use ANSI SQL-92 join syntax.

SQL • Check 

Rule Details

Severity Level

 Blocker 

Quality Area

Reliability 

Fixing times

10 mins
Local-high 

Database Support

OracleDB (v7.3.4 +)

Reason

ANSI SQL-92 join syntax is not supported by all Oracle Database features. For example, SQL Assertions in Oracle AI Database 26ai require join conditions to be specified in the `WHERE` clause.

ANSI SQL-92 joins have also been associated with optimizer bugs in older Oracle Database releases. Avoiding them can therefore improve compatibility when code must support older database versions.

Register Custom Rule



Rule

* Rule Name

R-3131

* Rule Title ?

Never use ANSI SQL-92 join syntax.

Always..., Never..., Avoid..., Try...

* Rule Reason

ANSI SQL-92 join syntax is not supported by all Oracle Database features. For example, SQL Assertions in Oracle AI Database 26ai require join conditions to be specified in the 'WHERE' clause.

ANSI SQL-92 joins have also been associated with optimizer bugs in older Oracle Database releases. Avoiding them can therefore improve compatibility when code must support older database versions.

* Software Quality Area

Reliability X v

* Severity Level

Blocker X v

* Severity

Error X v

* Fixing Time

Local-high X v

* Profile

Strict X v

Tags

SQL (Core) 1 v

Create tag

* Rule Has Check



* Rule Has Quick Fix



* Rule Check Is Free



dbLint

Rules

Configuration

Administration

Example Set

Name

OracleDB

Cancel

Non Compliant

Search

Compliant Exar

Search

Edit Non Compliant Example

Non Compliant Example

Number

1

Source Code

1

2

3

4

5

6

7

8

select emp.employee_id

,emp.last_name

,emp.first_name

,dept.department_name

from employees emp

join departments dept

on dept.department_id = emp.department_id

where extract(month from emp.hire_date) = extract(month from sysdate);

Ln 6, Col 3

Explanation

Diagnostic Markers

Actions

Edit

Add Row

Reset

		Start Line ↑≧1	Start Col ↑≧2	End Line	End Col	Message
☒		6	3	6	7	Use Oracle instead of ANSI SQL-92 join syntax.

1 rows selected

|<

<

1

>

>|

1 - 1 of 1

Cancel

Delete

Apply Changes

12

Generate Java Validator Project

Generate Java Validator Project

```
dblint gen-java \  
  --repoUrl=https://api.dblinter.app \  
  --tenantName=Demo \  
  --userName=philipp.salvisberg+42@gmail.com \  
  --accessToken=${DBLINTER_DEMO_ACCESS_TOKEN} \  
  --configName=Default \  
  --indent=4 \  
  --outputName=dbLinter-Demo-Custom-Validator \  
  --tenantFilter=Demo \  
  --ruleFilter=.+ \  
  --groupId=com.grisselbav \  
  --packageName=com.grisselbav.demo.validator
```

Result of "dblint gen-java"

```
find dbLinter-Demo-Custom-Validator -type f | sort
```

```
dbLinter-Demo-Custom-Validator/.gitignore
dbLinter-Demo-Custom-Validator/.mvn/wrapper/maven-wrapper.properties
dbLinter-Demo-Custom-Validator/mvnw
dbLinter-Demo-Custom-Validator/mvnw.cmd
dbLinter-Demo-Custom-Validator/pom.xml
dbLinter-Demo-Custom-Validator/README.md
dbLinter-Demo-Custom-Validator/src/main/java/com/grisselbav/demo/validator/DemoR3131.java
dbLinter-Demo-Custom-Validator/src/test/java/com/grisselbav/demo/validator/AbstractApxTest.java
dbLinter-Demo-Custom-Validator/src/test/java/com/grisselbav/demo/validator/AbstractJdbcTest.java
dbLinter-Demo-Custom-Validator/src/test/java/com/grisselbav/demo/validator/AbstractTest.java
dbLinter-Demo-Custom-Validator/src/test/java/com/grisselbav/demo/validator/DemoR3131Test.java
dbLinter-Demo-Custom-Validator/src/test/resources/application.properties
```

README.md

dbLinter Custom Validator for Demo

Project Structure

- `src/main/java/com/grisselbav/demo/validator` : A class for each dbLinter rule.
- `src/test/java/com/grisselbav/demo/validator` : A test class for each dbLinter rule.
- `src/test/resources/application.properties` : Optional JDBC configuration for rule checks that may improve results when using read-only database access.

Prerequisites

- JDK 17 or newer.
- Ensure the environment variable `JAVA_HOME` points to that JDK.

The Maven wrapper script (`mvnw`) simplifies the build process, so no Maven installation is required.

Build and Test

Build the JAR with tests:

```
./mvnw clean package
```

Build the JAR without tests:

```
./mvnw clean package -DskipTests=true
```

Run a single test:

```
./mvnw -Dtest=DemoR3131Test test
```

DemoR3131Test.java

```
19 public class DemoR3131Test extends AbstractTest {
29     @Nested
30     public class OracleDB {
31
32         @Test
33         public void non_compliant_1() {
34             parse("""
35                 select emp.employee_id
36                     ,emp.last_name
37                     ,emp.first_name
38                     ,dept.department_name
39                 from employees emp
40                 join departments dept
41                 on dept.department_id = emp.department_id
42                 where extract(month from emp.hire_date) = extract(month from sysdate);
43             """);
44             check("Demo", "R-3131");
45             Assertions.assertEquals(1, issues.size());
46             // issue 1
47             var issue1 = issues.get(0);
48             Assertions.assertEquals("""
49                 Use Oracle instead of ANSI SQL-92 join syntax.""", issue1.getMessage());
50             Assertions.assertEquals(6, issue1.getRange().getStartLine());
51             Assertions.assertEquals(3, issue1.getRange().getStartCol());
52             Assertions.assertEquals(6, issue1.getRange().getEndLine());
53             Assertions.assertEquals(7, issue1.getRange().getEndCol());
54         }
55     }
56 }
```

DemoR3131.java

```
1  /*
2   * generated by dbLinter CLI
3   */
4  package com.grisselbav.demo.validator;
5
6  > import ...
7
12
13  /**
14   * Demo R-3131: Never use ANSI SQL-92 join syntax.
15   */
16  public class DemoR3131 extends AbstractCheck { no usages
17      @Check(tenant = "Demo", rule = "R-3131") no usages
18      public List<CheckIssue> checkPlsqlStatement(IslandSqlParser.FileContext ctx) {
19          // TODO: Implement one or more checks with different, suitable IslandSqlParser contexts.
20          return checkIssues;
21      }
22  }
```

How to find the
right one?

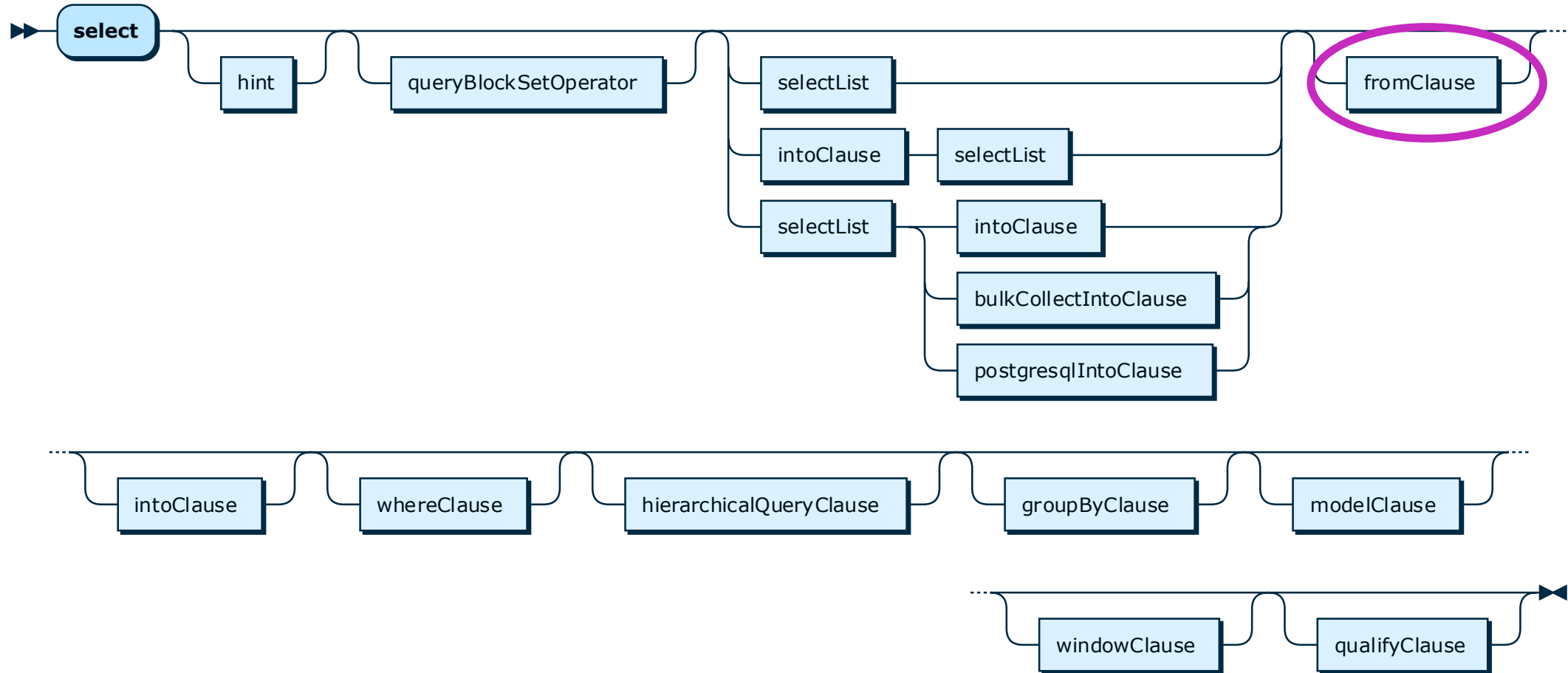
Register context:
type in parse tree
to check

IslandSQL

Grammar & Parse Tree

queryBlock

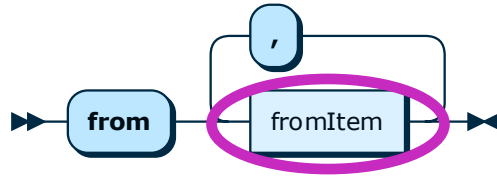
```
select *  
  from emp  
 join dept using (deptno);
```



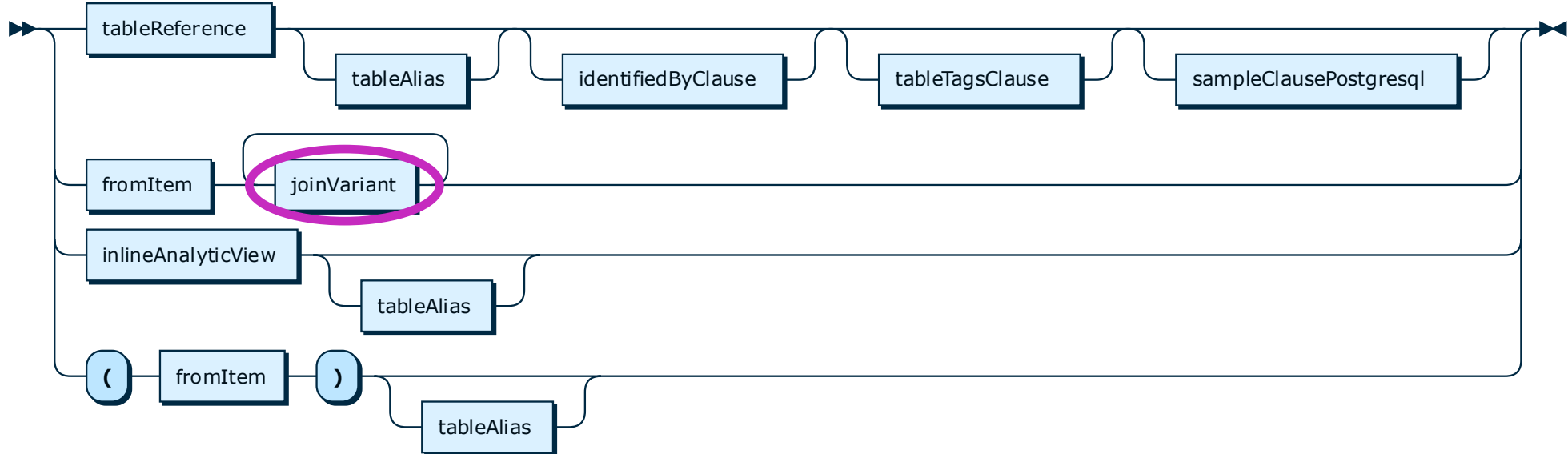
fromClause

```
select *  
  from emp  
 join dept using (deptno);
```

fromClause:



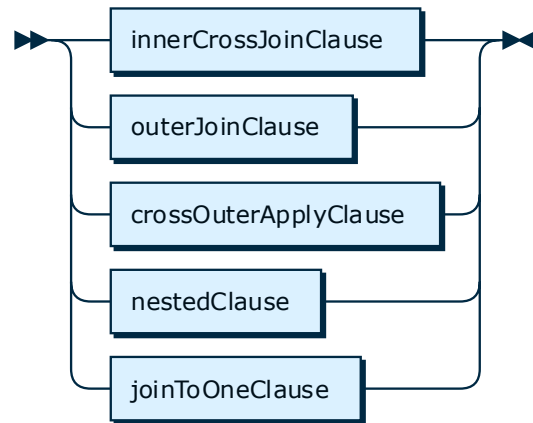
fromItem:



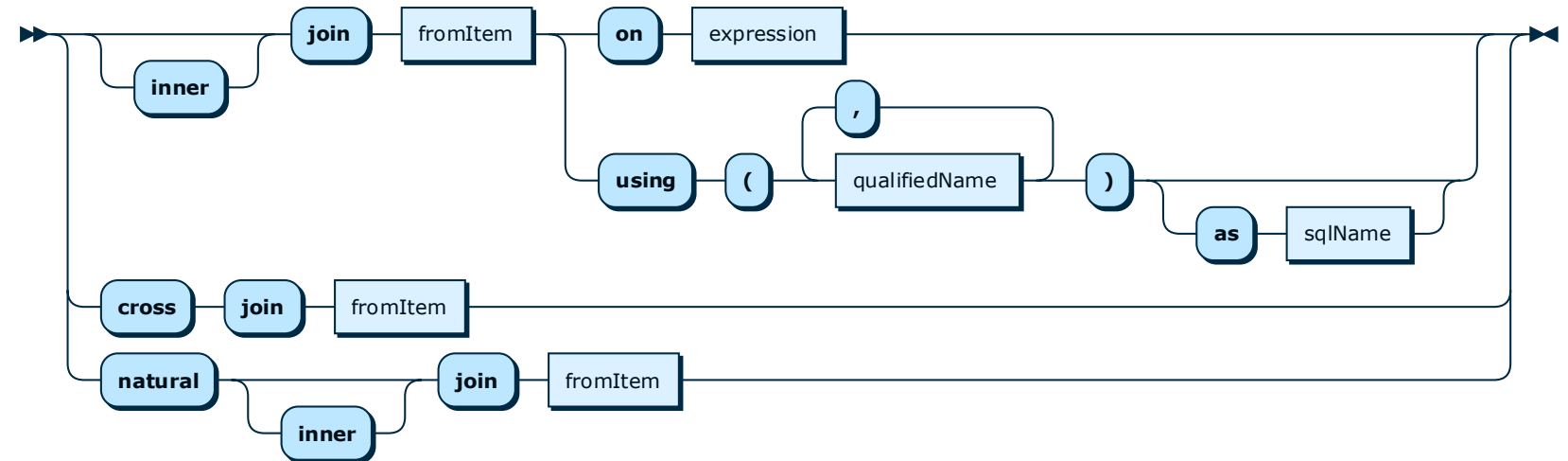
joinVariant

```
select *  
  from emp  
  join dept using (deptno);
```

joinVariant:



innerCrossJoinClause:

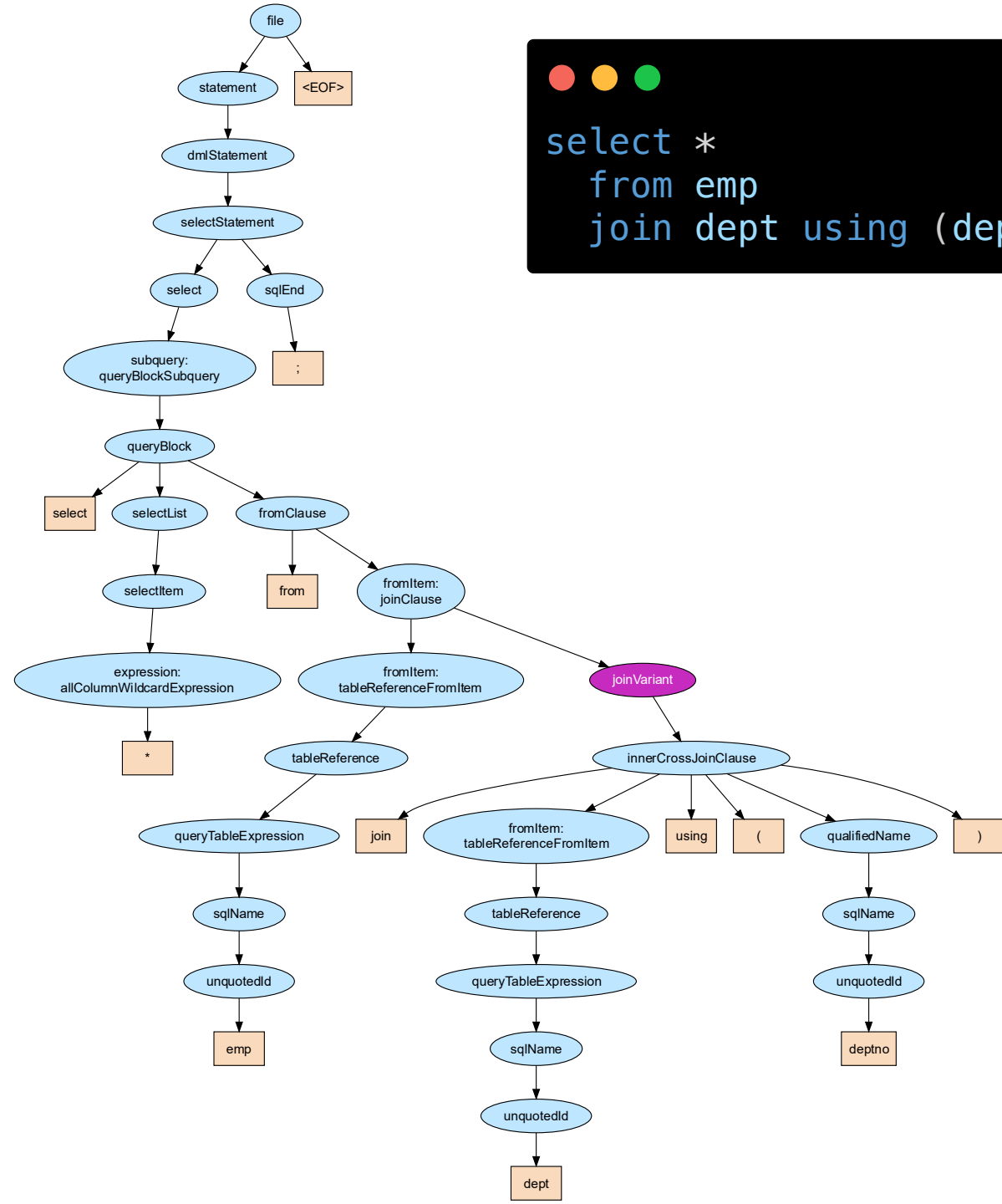


Parse Tree

```

file
  statement
    dmlStatement
      selectStatement
        select
          subquery:queryBlockSubquery
            queryBlock
              K_SELECT:select
                selectList
                  selectItem
                    expression:allColumnWildcardExpression
                      AST:*
            fromClause
              K_FROM:from
                fromItem:joinClause
                  fromItem:tableReferenceFromItem
                    tableReference
                      queryTableExpression
                        sqlName
                          unquotedId
                            ID:emp
                joinVariant
                  innerCrossJoinClause
                    K_JOIN:join
                      fromItem:tableReferenceFromItem
                        tableReference
                          queryTableExpression
                            sqlName
                              unquotedId
                                ID:dept
                    K_USING:using
                      LPAR:(
                        qualifiedName
                          sqlName
                            unquotedId
                              ID:deptno
                      RPAR:)
            sqlEnd
              SEMI:;
          <EOF>

```

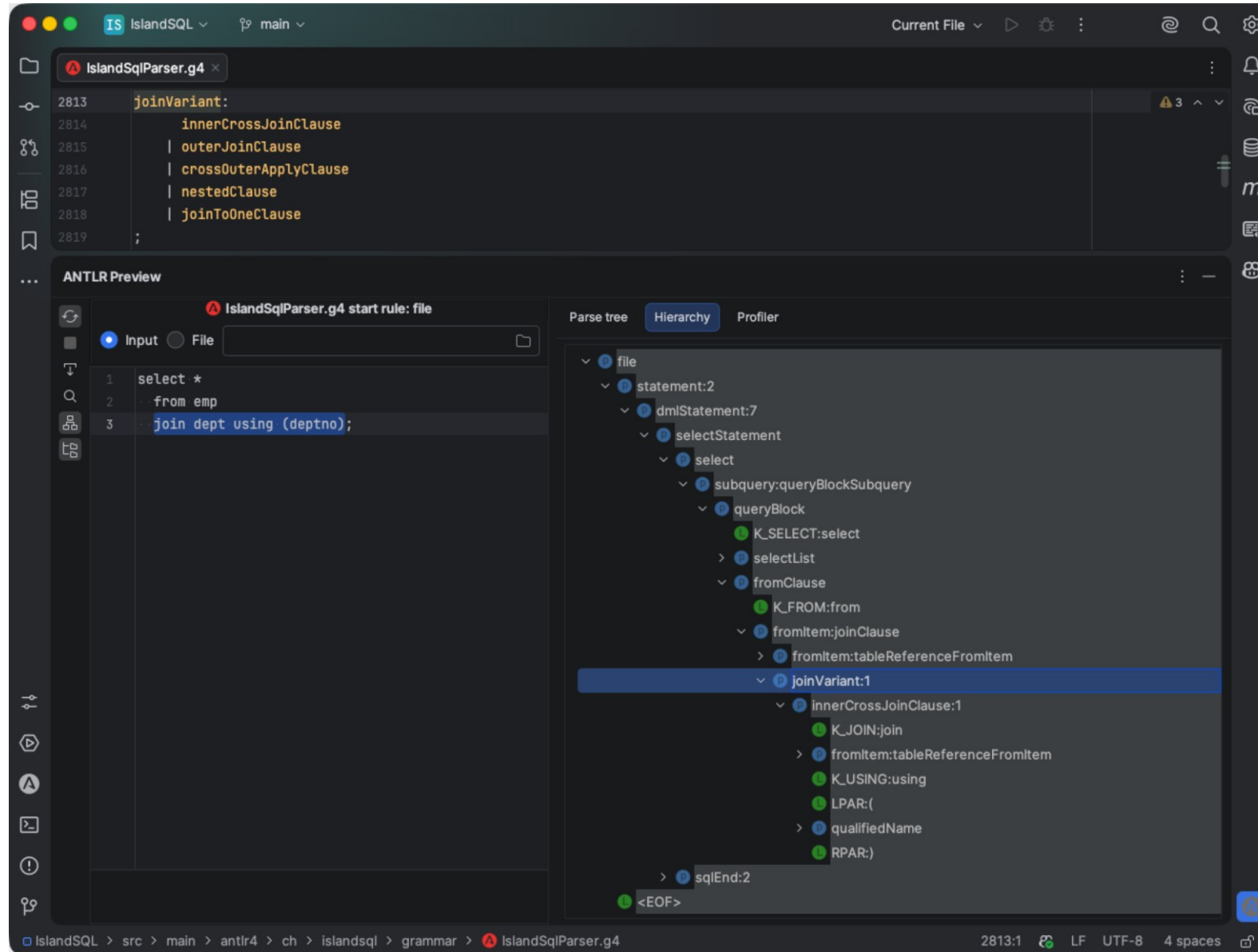


```

select *
from emp
join dept using (deptno);

```

IntelliJ IDEA ANTLR4 Plugin



Complete Project and Test (Demo)

<https://github.com/Grisselbav/dbLinter-Demo-Custom-Validator>

Register Issue

DemoR3131.java

```
/**
 * Demo R-3131: Never use ANSI SQL-92 join syntax.
 */
public class DemoR3131 extends AbstractCheck {
    @Check(tenant = "Demo", rule = "R-3131")
    public List<CheckIssue> checkPlsqlStatement(IslandSqlParser.JoinVariantContext ctx) {
        addIssue()
            .setMessage("Use Oracle instead of ANSI SQL-92 join syntax.")
            .setRange(ctx.start);
        return checkIssues;
    }
}
```

Register Issue With Quick Fixes

```

● ● ● DemoR3131.java

/**
 * Demo R-3131: Never use ANSI SQL-92 join syntax.
 */
public class DemoR3131 extends AbstractCheck {
    @Check(tenant = "Demo", rule = "R-3131")
    public List<CheckIssue> checkPlsqlStatement(IslandSqlParser.JoinVariantContext ctx) {
        addIssue()
            .setMessage("Use Oracle instead of ANSI SQL-92 join syntax.")
            .setRange(ctx.start)
            .addQuickFix("Convert to Oracle join syntax", () ->
                List.of(new Replacement(new Range(ctx),
                    ctx.start.getText() +
                        " /* TODO: convert to Oracle join */"), true)
            .addDbLintIgnoreQuickFix(ctx.start, "Keep ANSI SQL-92 join because ...")
            .addFalsePositiveQuickFix(ctx.start);
        return checkIssues;
    }
}
```

Build Project

```
./mvnw clean package
```

```
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.grisselbav:custom-validator >-----
[INFO] Building dbLinter Validator 1.0.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] ...
[INFO] --- surefire:3.5.4:test (default-test) @ custom-validator ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] T E S T S
[INFO]
[INFO] Running com.grisselbav.demo.validator.DemoR3131Test
[INFO] Running com.grisselbav.demo.validator.DemoR3131Test$OracleDB
[INFO] Running com.grisselbav.demo.validator.DemoR3131Test$OracleDB$Solution
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.051 s -- in com.grisselbav.demo.validator.DemoR3131Test$OracleDB$Solution
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.781 s -- in com.grisselbav.demo.validator.DemoR3131Test$OracleDB
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.787 s -- in com.grisselbav.demo.validator.DemoR3131Test
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.5.0:jar (default-jar) @ custom-validator ---
[INFO] Building jar: dbLinter-Demo-Custom-Validator/target/custom-validator-1.0.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 2.420 s
[INFO]
```

Publish Validator

```
./mvnw deploy

[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.grisselbav:custom-validator >-----
[INFO] Building dbLinter Validator 1.0.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] ...
[INFO]
[INFO] --- exec:3.6.3:exec (import-validator) @ custom-validator ---
Importing validator ...
Validator imported successfully from dbLinter-Demo-Custom-Validator/target/custom-validator-1.0.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.543 s
[INFO] -----
```

Test in VS Code

The screenshot shows the VS Code editor with a file named `3131-non-compliant.sql`. The file contains two SQL queries. The first query on line 3 uses `join` with `using (deptno)`, which is underlined in red. The second query on line 10 uses `join` with `on dept.department_id = emp.department_id`, which is also underlined in red. Below the editor, the **Problems** panel shows two errors: "Use Oracle instead of ANSI SQL-92 join syntax. dbLinter(Demo R-3131) [Ln 3, Col 3]" and "Use Oracle instead of ANSI SQL-92 join syntax. dbLinter(Demo R-3131) [Ln 10, Col 3]". A context menu is open for the second error, offering the following options: "Convert to Oracle join syntax" (highlighted in blue), "Add @dbLinter ignore marker with reason: Keep ANSI SQL-92 join because", "Add @dbLinter ignore marker with reason: False positive.", "Fix all R-3131 problems like 'Convert to Oracle join syntax'", "Fix all problems.", "Fix", and "Explain".

```
3131-non-compliant.sql 2
3131-non-compliant.sql > ...
1  select *
2    from emp
3    join dept using (deptno);
4
5  select emp.employee_id
6         ,emp.last_name
7         ,emp.first_name
8         ,dept.department_name
9    from employees emp
10   join departments dept
11      on dept.department_id = emp.department_id
12  where extract(month from emp.hire_date) = extract(month from sysdate);
```

Problems 2 Terminal PostgreSQL Query Results SQL History Output Debug Console

3131-non-compliant.sql 2

- ⊗ Use Oracle instead of ANSI SQL-92 join syntax. dbLinter(Demo R-3131) [Ln 3, Col 3]
- 💡 Use Oracle instead of ANSI SQL-92 join syntax. dbLinter(Demo R-3131) [Ln 10, Col 3]

Convert to Oracle join syntax

- Add @dbLinter ignore marker with reason: Keep ANSI SQL-92 join because
- Add @dbLinter ignore marker with reason: False positive.
- Fix all R-3131 problems like 'Convert to Oracle join syntax'
- Fix all problems.
- Fix
- Explain

Core Messages

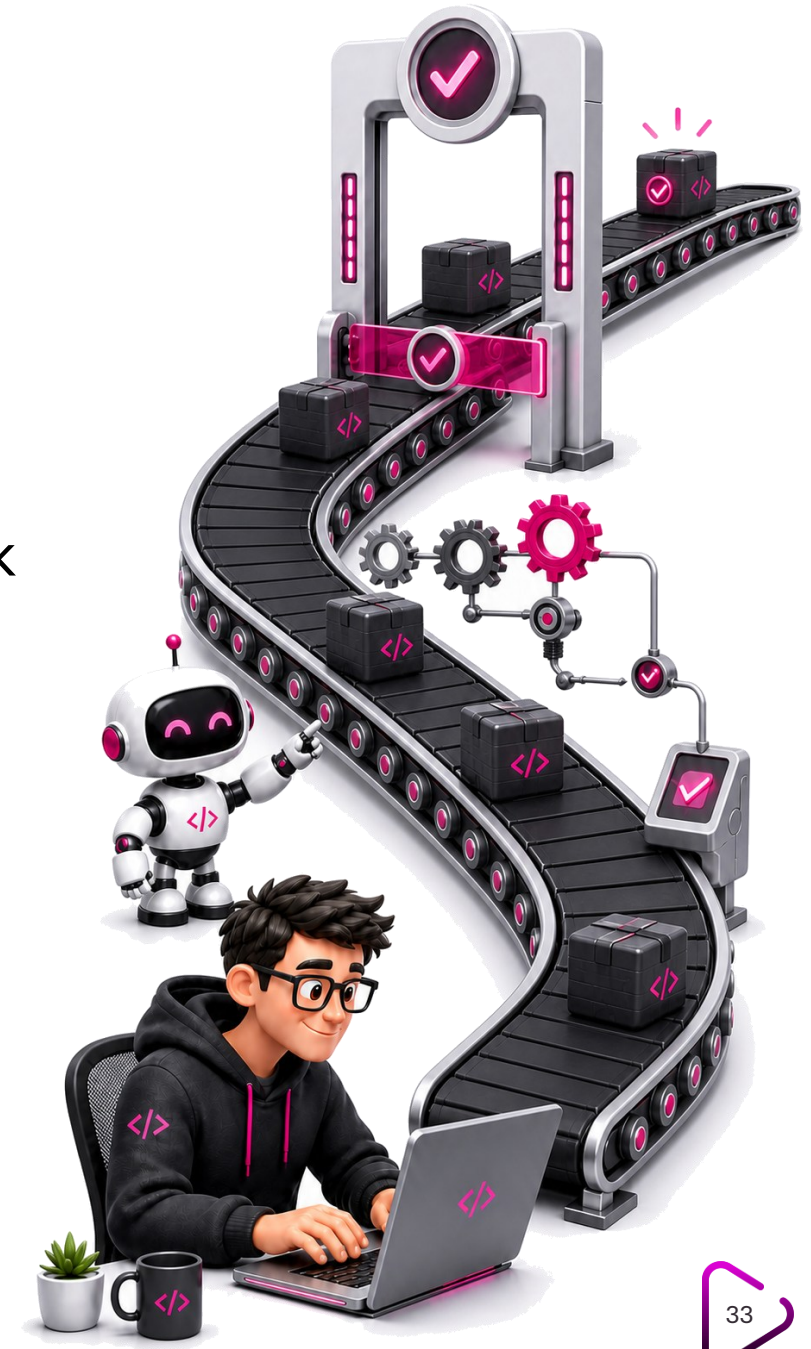
The Right Home for Every Rule

- Propose broadly useful rules in the [dbLinter GitHub repository](#)
- Suggest improvements to existing rules
- Define custom rules for project-specific or conflicting requirements



Make Your Rules Executable

- Give developers feedback while they are writing code
- Apply the same standard in the IDE and build pipelines
- Enforce those rules through quality gates
- Implementing custom rules is easier than you might think



Thank you!

