

# ClearSQL for Oracle

## ClearSQL at a Glance

Working with large collections of PL/SQL code files is a complex process. PL/SQL developers can spend much of their time diving into poorly written, sparsely documented legacy code. While this code can be compiled to get the job done, it can remain virtually unreadable, hard to maintain and difficult to debug. To solve

this problem, Conquest Software Solutions provides *ClearSQL* for Oracle, a powerful and flexible tool that gives PL/SQL developers the confidence to obtain, maintain, analyze, fix, format, automatically review PL/SQL scripts and generate clickable flowchart and call tree diagrams and CRUD matrices.

## What is ClearSQL and what can it do for me?

- *ClearSQL* is a code review and quality control tool for Oracle PL/SQL
- With *ClearSQL*'s code review feature, you obtain recommendations for better coding style and check for error-prone places in your project
- *ClearSQL* generates a series of industry standard quality control metrics about PL/SQL source code to help identify potential problems in the development and maintenance of your software and to fine-tune your software development process
- *ClearSQL* reports code deficiencies, creates PL/SQL code Flowcharts and Call Tree diagrams, clickable CRUD1 and CRUD2 type matrices, and exports them to "VDX" (MS Visio XML format) and popular image formats (PNG, GIF, PNG)

[www.myclearsql.com](http://www.myclearsql.com)



## How is ClearSQL positioned?

To be **THE** maintenance tool for PL/SQL legacy code; especially if badly or even undocumented

*ClearSQL* is known as an easy to use and very cost effective tool to keep even legacy PL/SQL code alive and in production. This quote from a experienced *ClearSQL* user, a very large IT consulting company, describes it in the best way:

*"The demands of a VERY busy project with ridiculous time-frames mean that the code needs to be written fast. Having said that, this is exactly the situation that causes the issues that ClearSQL can help with."*

*From a personal note, if we know the code and write it according to a set of good coding standards then ClearSQL has very little problem with what we write. However, the beauty of the tool for me is when I have to work with unfamiliar code that has not been well structured. Its then that the tool is worth its weight in gold."*

## PL/SQL Analyzer, Editor and Flowcharter

Edit, format and analyze your PL/SQL code. *ClearSQL* for Oracle is a full-blown code analyzer and editor for legacy code maintenance. It performs declaration analysis, DML analysis, cursor analysis, control flow analysis, procedure analysis,

exception analysis and module analysis and displays the results in a Structure View, facilitating quick navigation through your code. Remove anomalies, fix and format your PL/SQL automatically. Count metrics and enforce coding standards.

## CRUD Matrix

A CRUD matrix documents the data elements that objects access in a database.

During code analysis CRUD (*Create, Read, Update, Delete*) matrices are created. A CRUD matrix can help test code for large projects and is critical for maintaining large applications. There are two types of CRUD matrices available in *ClearSQL*: **CRUD1** (script level) and

**CRUD2** (database level). CRUD matrices are click-able. Click on any object name in a matrix, and the related code will automatically be loaded and the relevant line of PL/SQL code will be highlighted in the code editor.

## ClearSQL is a code analyzer, and not a simple code formatter!

## Flowchart PL/SQL Code

*ClearSQL* for Oracle takes a package or a stand-alone subroutine and makes a set of Flowchart diagrams that visualize the code execution path. Such diagrams show the conditional branches, loops and jumps, thereby helping to understand the

opaque logic. The visual patterns help to find the points of possible code refactoring or module restructuring, and make the reasons for high values of Cyclomatic Complexity metrics obvious a tremendous aid when working with unknown code.

## Create Call Tree Diagrams

*ClearSQL* for Oracle can also create diagrams of Call Trees from any PL/SQL code. A PL/SQL Call Tree diagram is a

perfect aid to reading and understanding the data flow of legacy code.

## Click-able Flowcharts, Call Tree Diagrams and CRUD Matrices

As one of its unique features, *ClearSQL* for Oracle supports creation of PL/SQL code Flowcharts, CRUD matrices and Call Tree diagrams that are "click-able". Click on any element within a diagram/matrix, and the related code will automatically be loaded and the relevant line of PL/SQL

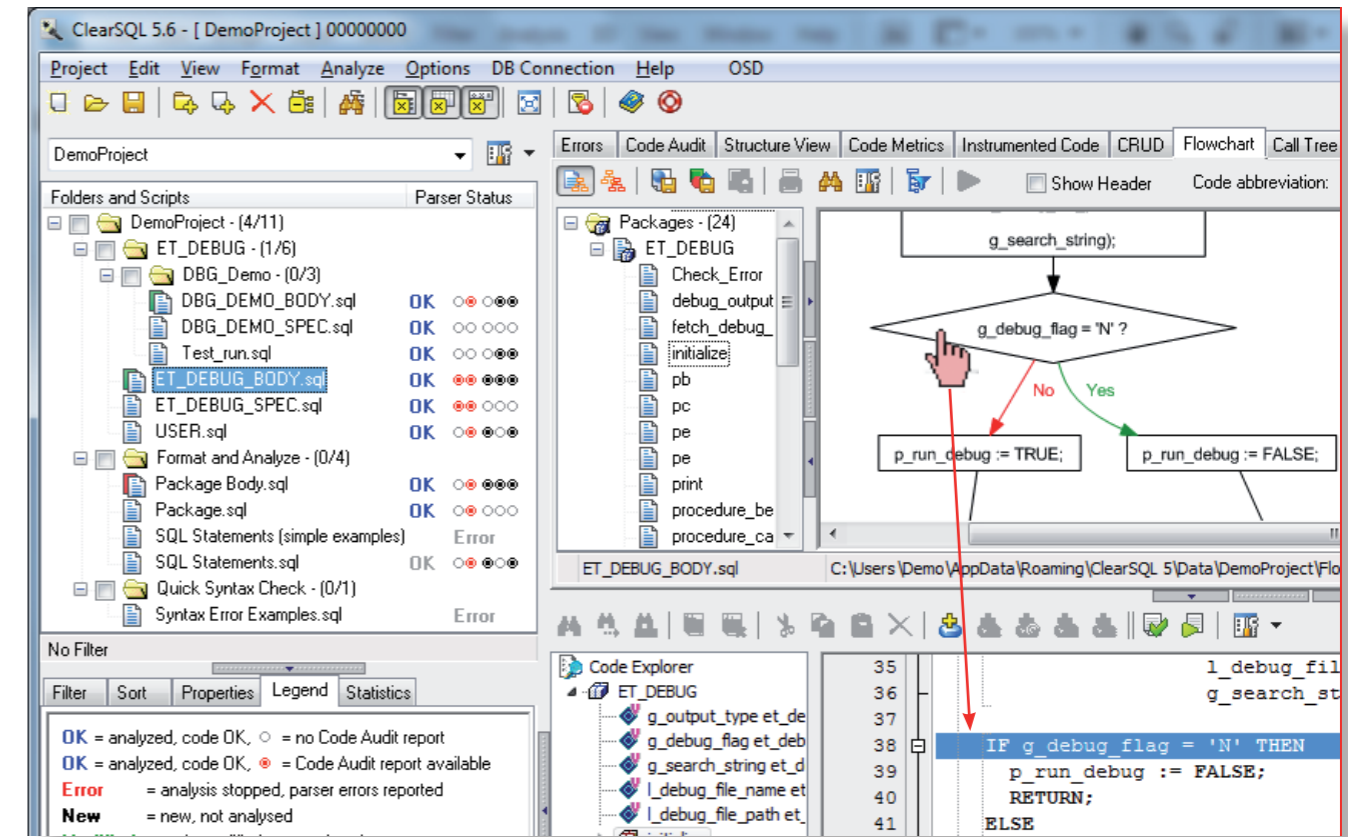
code will be highlighted in the code editor. This is a superior aid for PL/SQL developers for understanding and maintaining unknown code or legacy code. The feature "click-able diagrams/matrices of PL/SQL code" enhances *ClearSQL*'s unique position in its market.

## Code Editor With Syntax Highlighting and Code Folding

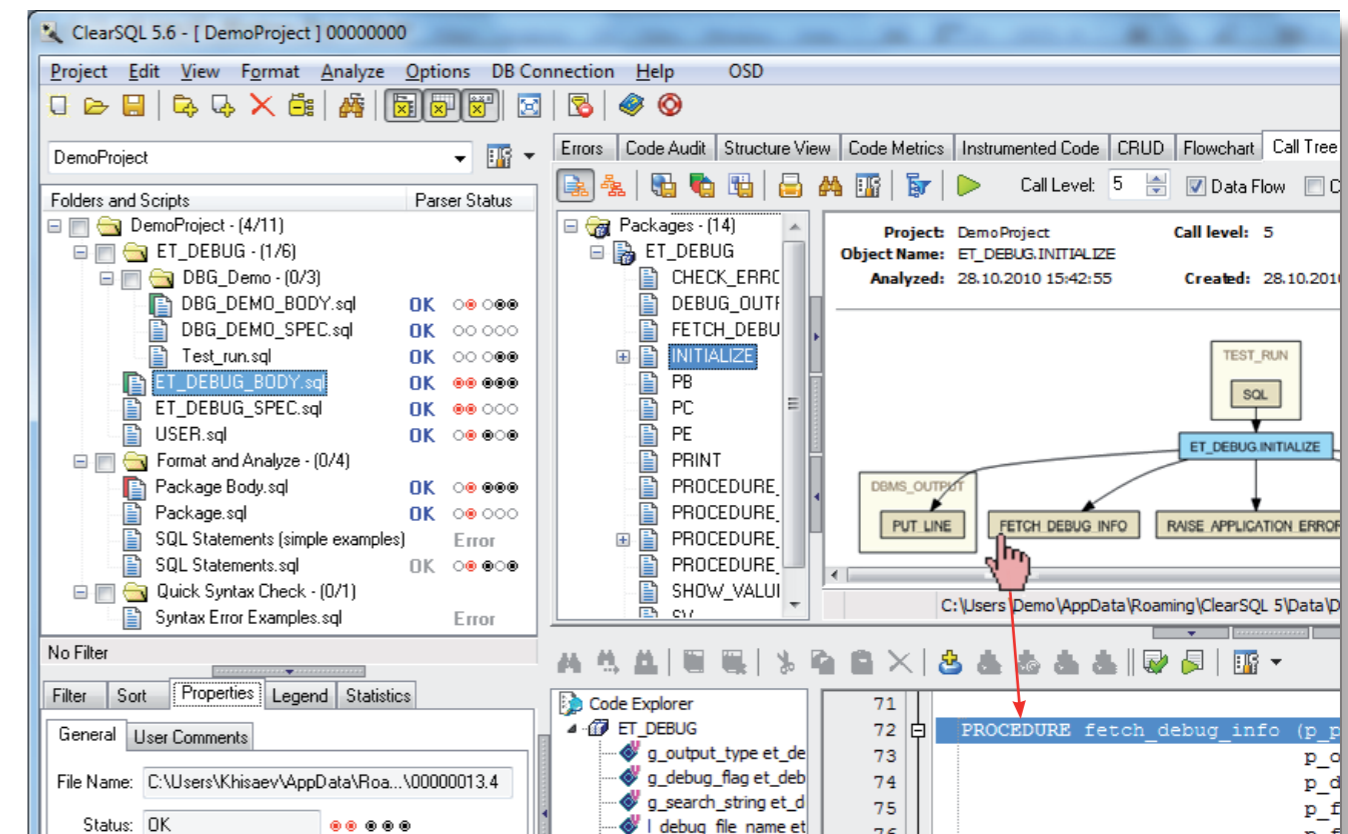
The syntax highlighting and code folding Code Editor provides support for SQL and PL/SQL code and allows to create different versions of the same code. Rapid code-building features like Code

Explorer, Code Insight, smart code completion, dot suggestion, convenient bookmarks and user templates saves time, reduce errors and ensures a consistent programming style.

## Clickable Flowchart



## Clickable Call Tree Diagram



## Automated Code Review

Catch bugs early in the development cycle and avoid common mistakes. Produce code that is easy to read, understand and maintain. The nearly 60 code

review rules detect the misuse of GOTO, parameter declaration problems, unreferenced parameters, missing or misplaced RETURN, unlabeled END, etc.

## Auto Fix Known PL/SQL Problems

*ClearSQL* for Oracle adds missing END and LOOP labels, defines IN para-

meters and re-parenthesizes complex operations.

## Enforce Coding Standards

Improve readability and standardize code written by others.

Enforce naming standards with regular expressions.

## Format Your Code Automatically

Set margins and indentation. Insert blank lines. Apply standard casing to keywords and identifiers.

Tidy up procedure headers. Align SQL and PL/SQL statements.

## Verify PL/SQL Syntax Without Executing in Oracle

In many cases, minor syntax “pseudo errors” can be worked around by commenting out or renaming. Since *ClearSQL* works in a copy of the actual

code, it's easy to temporarily fix these pseudo errors (commenting or renaming) to allow full code analyzing, formatting and diagrams to be built.

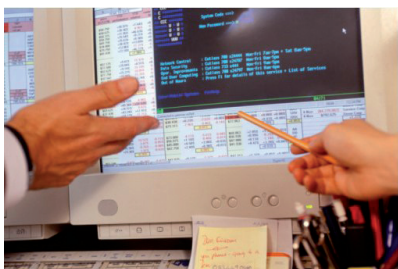
## Quality Control Code Metrics

Determine the complexity of your system. Identify potential problem areas based on complexity, size and modularity. Available metrics: lines of code (LOC), effective LOC, logical statements LOC, comment lines, blank lines, physical lines, McCabe cyclomatic

complexity v(G), number of input parameters, number of return points, interface complexity, functional complexity, Halstead Program Length, Vocabulary, Volume, Difficulty and Effort, and Maintainability Index.

## Additional Features

- Project Wizards creates *ClearSQL* projects based on files and DB objects easily with a few clicks
- Project-Tree Manager filters and sorts scripts by a variety of predefined filters and sorting rules; and also displays script properties and its legend
- Built-in Version Control System manages multiple versions of scripts for easy change tracking and retrieval
- Logs on to an Oracle database by using TNS, LDAP and Direct connections
- Pre-processor directives let you generate different versions of the same code and include/exclude debug code
- Creates complex analysis reports in a convenient HTML file format, including Parser errors and warnings, code Flowcharts and more...
- Syntax Checker verifies your PL/SQL syntax without executing it on Oracle
- Import from Database lets you connect to your Oracle database and import your files directly; large number of files can be imported with a few clicks as well
- Integrated „Online Support Desk“ allows a user to direct communicate with Conquest's Technical Support and keeps *ClearSQL* up-to-date with its auto-update feature



[www.myclearsql.com](http://www.myclearsql.com)

# ClearSQL - FAQ

## Who will be a typical user of ClearSQL?

- PL/SQL developers who maintain, fix errors, modify or provide support for their own, unknown or “legacy” PL/SQL code
  - hundreds of thousands of Oracle databases worldwide containing billions of legacy PL/SQL scripts
  - in many cases there is no documentation available and the original developer is no longer present
- PL/SQL development teams who use *ClearSQL* to implement coding standards, code quality insurance, and code formatting to easier maintain their code; and to speed up PL/SQL code development and test cycles
- PL/SQL developers responsible to support thousands of PL/SQL files and looking for a tool who is able to analyze and document those scripts only with a few clicks

[www.myclearsql.com](http://www.myclearsql.com)

## What makes ClearSQL unique in the market?

- Intensive market investigations did not provide information of any widely known product that could be considered a “real” competitor to *ClearSQL*
- It performs PL/SQL code analysis, code formatting, auto-correction of known common code errors, supporting PL/SQL version from 2.1 to 11.2
- It automatically analyzes large projects with thousands of PL/SQL scripts with the click of a button
- It creates Flowcharts, Call Tree diagrams and CRUD matrices from PL/SQL code accelerating the understanding of the Logic and Inter-Code Calling Relationships of unfamiliar or legacy code
- Flowchart, Call Tree diagram and CRUD matrix are clickable; with a click on a diagram or matrix element, it loads that code to the editor and highlights the corresponding line of code
- A Syntax Checker verifies PL/SQL syntax without executing it in Oracle (=offline)
- Provides complex code metrics information to identify potential problem areas based on complexity, size and modularity

## How can I integrate ClearSQL into my database development environment?

*ClearSQL* works independently from any software development environment. Code is imported into *ClearSQL* either from the file system or directly from a database. By importing, the code is physically moved into *ClearSQL*'s internal storage. By duplicating code in this fashion, *ClearSQL* allows you to

experiment with the code without having to worry about unwanted side effects on your software development environment. Only when exporting files from *ClearSQL*, will changes made while working with *ClearSQL* take permanent effect outside of *ClearSQL*'s own file storage system.

### **Will *ClearSQL*'s parser accept all my PL/SQL code?**

Unfortunately, Oracle has not published reference grammar for its PL/SQL language that allows third-party tool to parse code against the reference grammar. The grammar documented in Oracle's PL/SQL reference guide is not complete and cannot serve as reference grammar. In addition, the PL/SQL gram-

mar differs between Oracle versions. Therefore, we cannot guarantee that your PL/SQL code will pass *ClearSQL*'s parser. As we have a strong commitment to help our customers, we always will work with them to resolve any compatibility issues may come across when working with *ClearSQL*.

### **Sometimes, *ClearSQL* is very particular about keyword usage. Why?**

Both ANSI SQL and Oracle SQL use two types of keywords – reserved and non-reserved. For example, the keyword SELECT is considered reserved in both ANSI SQL and Oracle SQL, and – as a consequence – you cannot name any entity (program variable, table name, column name ..) SELECT. With some keywords, ANSI SQL and Oracle SQL disagree about a particular keyword's

reserved status. For example the keyword MEMBER, e.g. is reserved in ANSI SQL but non-reserved in Oracle. *ClearSQL*'s own parser has been constantly improved and enhanced over its 8 years of development and being in use. If our users are facing any problems in this area, we usual provide a fix within 1-2 weeks after the problem report was received.

### **What exactly does the *ClearSQL* parser check?**

The *ClearSQL* parser is not a replacement, substitute or alternative for Oracle's PL/SQL compiler. To support *ClearSQL*'s functionality (formatting of PL/SQL code, Flowcharting, Call Tree Diagrams and CRUD matrices), the parser will check syntax and static semantics on a per-file basis. The parser will not

perform any inter-file checking, e.g. calls to entries in other packages will not be checked for correct parameter usage. It is our medium-term goal to enhance the parser to check the complete static semantics of PL/SQL, including complete interfile checking.

### **Why does *ClearSQL* come with it's own preprocessor? Isn't Oracle's PL/SQL preprocessor good enough?**

*ClearSQL*'s preprocessor serves a different purpose than Oracle's PL/SQL preprocessor. *ClearSQL* uses its own preprocessor to generate diagram variants per preprocessor settings. To be able to do that, the preprocessor had

to be closely integrated with the parser. There is no separate preprocessor phase in *ClearSQL*, as preprocessor directives work as extensions to the language accepted by *ClearSQL*'s parser.