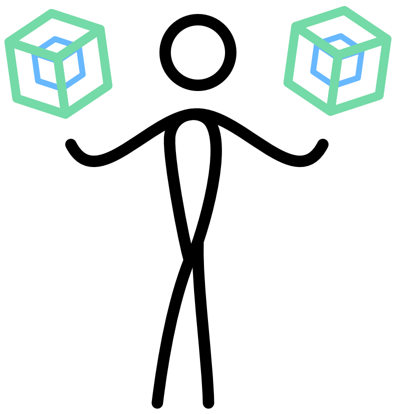
CobolCloud Embedded SQL Pre-compiler Product Brief

Overview

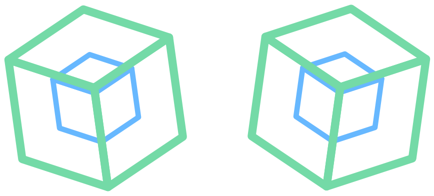
CobolCloud Embedded SQL Pre-compiler (CLDSQL) has been conceived to allow enterprises to preserve their mission-critical COBOL ESQL applications by pre-compiling them without source code modifications and by reproducing the runtime behaviors on which they rely. Architected as a multi-dimensional API, CLDSQL provides the ability to easily swap components in and out and provide access to any data source. Reusable rules and tasks can be directly applied to the programs during the preparation phase without modifying the original COBOL SQL code. Among others, CLDSQL supports PostgreSQL, Oracle, MySQL, and ODBC and is easily extensible to use with other databases.



CLDSQL has two basic modules

CLDSQL comprises two basic modules:

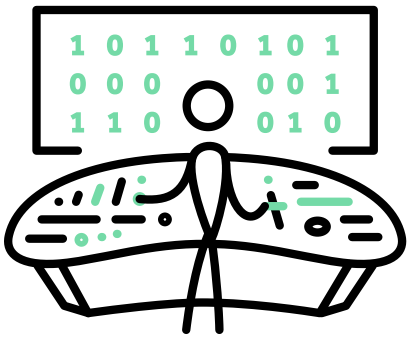
- An intelligent preprocessor that converts the original COBOL ESQL code, analyzes it, and generates code to be used by the CobolCloud compiler. In the pre-processing phase, there is an analysis phase and a translation phase, each of which can be adapted to exploit the capabilities of the database in use most efficiently.
- A series of runtime libraries and tools that interface with the DBMS used by the client. The CLDSQL runtime component allows it to support the most widely used Open Source and proprietary database solutions, while also being easily extensible to newly emerging database solutions.



Both modules are highly modular multi-directional APIs

CLDSQL is modular both during the preparation phase (parsing and generation of ESQL statements) and at runtime. The translator module in CLDSQL can apply transformation rules to rewrite statements (loops, cursor usage, etc.) with the aim of keeping performance intact when migrating from the mainframe environment to open systems.

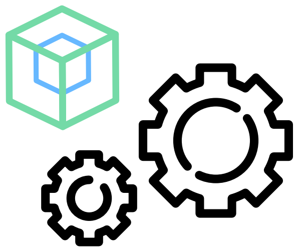
The preprocessor in CLDSQL can point out features and potential "problem areas" in the code that might need particular attention when porting an application from the mainframe to open systems.
CLDSQL can handle natively a plurality of databases at runtime, also simultaneously. Due to its modular nature, database support can be easily augmented or even replaced with custom-tailored modules, to better accommodate features used by the client's application.



CLDSQL allows you to control its behaviors

CLDSQL provides a wide-ranging set of parameters that can be used to control its behavior during both the generation and the runtime phase.
You can decide how character fields should be treated, for instance, or how SQL statement parameters should be generated. You can specify all the connection options supported by the DBMS of your choice. Decide to use the native cursor library provided by your DBMS or the emulated one in CLDSQL. There are also options to control the COBOL syntax that will be used, or how the runtime libraries should be called.

Or you can simply use the defaults implemented in CLDSQL if they suit your process. You can choose to take control, or not!



The CLDSQL transformation engine reduces the need to change your code

Migrating from a mainframe environment to open systems does not just mean "bringing it over" or "making it run". Radically different architectures also imply completely different assumptions about CPU and I/O speed, memory layout, etc. Subtle differences in how an SQL statement is interpreted and executed by each DBMS, can slow a perfectly-written program down to a crawl, or make it consume a huge quantity of resources. By using a CLDSQL transformation rule, it becomes possible to selectively target particular statements or situations within a program and adjust the software's behavior to meet specific requirements. All avoiding the need for direct code modifications and minimizing the risk of regressions or undesired changes in functionality.

The transformation engine integrated in CLDSQL exposes SQL statements, cursors and variable definitions in your code as objects that can be manipulated with the included interpreter. You do not need to modify the COBOL SQL code directly; reusable rules or tasks can be directly applied to your programs during the preparation phase without modifying the original COBOL SQL code. Queries can be manipulated and enriched: you can do things like add or remove columns, modify the function names used by a statement to make them compatible with your DBMS of choice, and much more!

Every software has its own needs in terms of performance, portability, and customization: CLDSQL comes with a predefined set of transformation rules, but new ones can be written to support a given migration project or even for a specific program.

Since you do not need to modify your original code, you can easily test the best strategy for your ESQL:
- Run your original code promptly and easily, then fine-tune your application for maximum performance.
- Or incorporate custom-made rules and tasks into your workflow to get maximum performance and reliability from the start.

Quality and security

CLDSQL supports SSL/TLS connections to PostgreSQL or other DBMSs of choice, using the DBMS-provided mechanisms (including authentication and authorization) for maximum security and reliability.
Every release of CLDSQL passes a thorough testing phase that checks for problems and vulnerabilities and is digitally signed for proper verification.

Integration

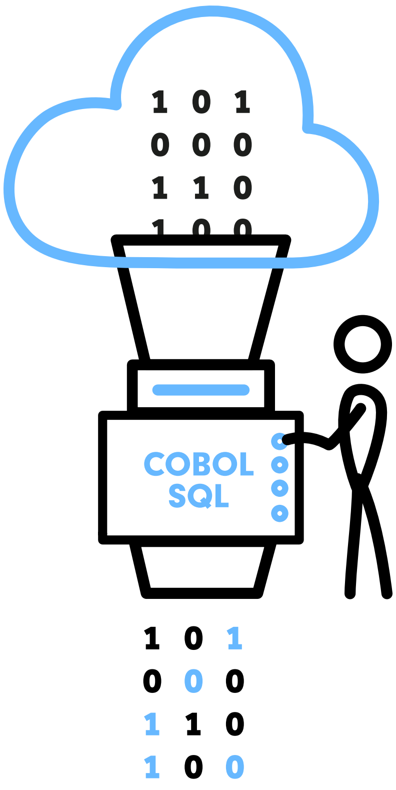
CLDSQL can use information generated by other tools of the CobolCloud suite (profiler and compiler) to analyze the code and generate the best possible data access strategy for each scenario.
The "preparation" components of CLDSQL (parser and transformation libraries) can output a wealth of information about the programs that goes well behind a standard COBOL listing, ready to be automatically processed by other tools (other preprocessors, debuggers, profilers, etc.) or used to integrate COBOL programs with other business processes.

Extensible logging system

CLDSQL uses an advanced logging system that can be tailored and extended to support multiple logging targets and filters. When needed, whether for debugging purposes, or simply to "keep tabs" on the application, custom logging modules can be written and added to the CLDSQL runtime logging system.

Cross-platform

CLDSQL is natively available on Windows and Linux. Conformance tests are separately executed – before each release - in each environment to ensure a consistent and flawless experience.



Make your COBOL applications shine in the Cloud

Thanks to its extensible and pluggable architecture, CLDSQL enables application to easily connect with databases residing in a shared or private cloud environment. Its low disk and memory footprint, together with its low overhead and its fast startup and connection time, make it ideal for containerized applications. CLDSQL supports the major databases on standard cloud architectures, including enhanced connection and service modes.

Features	Benefits
• CLDSQL supports PostgreSQL, ODBC, MySQL, Oracle, SQLite databases, and is easily extensible to use other databases	• The runtime mechanism provides access to the most widely used DBMS solutions, and is easily extensible: more DB engines can be added by writing a small runtime library
• CLDSQL has two basic modules	• CLDSQL comprises two basic modules: • a preprocessor that converts the original COBOL ESQL code, analyzes it and generates code to be used by the CobolCloud compiler • a series of runtime libraries and tools that interface with the DBMS used by the client
• CLDSQL is a highly modular, multi-directional API	• CLDSQL is modular both during the preparation phase (parsing and generation of ESQL statements) and at runtime. The translator module in CLDSQL can apply transformation rules to rewrite statements (loops, cursor usage, etc.) with the aim of keeping performance intact when migrating from the mainframe environment to open systems • The preprocessor in CLDSQL can point out features and potential "problem areas" in the code that might need particular attention • Due to its modular nature, database support can be easily augmented or even replaced with custom-tailored modules, to better accommodate features used by the client's application
• CLDSQL allows you to control its behaviors	• CLDSQL provides a wide-ranging set of parameters that can be used to control its behavior during both the generation and the runtime phase • Its unique combination of detection / transformation rules allows you to modify the generated code • There are also options to control the COBOL syntax that is used, or how the runtime libraries are called • Defaults implemented by CLDSQL can be simply used, if they suit your requirements
• The CLDSQL transformation engine reduces the need to change the code	• CLDSQL is equipped with a powerful analysis and transformation system based on a collection of rules that can be used as is, modified according to new needs, or with new rules that can be created by the user
• Quality and security	• CLDSQL supports SSL/TLS connections to PostgreSQL or other DBMSs of choice, using the DBMS-provided mechanisms (including authentication and authorization) for maximum security and reliability. Every release of CLDSQL passes a thorough testing phase that checks for problems and vulnerabilities

Features	Benefits
• Integration	• CLDSQL can use information generated by the CobolCloud compiler and profiler to analyze the code and generate the best possible data access strategy for each scenario.
• Extensible logging system	• CLDSQL uses an advanced logging system that can be tailored and extended to support multiple logging targets and filters. When needed, whether for debugging purposes, or simply to "keep tabs" on the application, custom logging modules can be written and added to the CLDSQL runtime logging system
• Cross-platform	• CLDSQL is natively available on Windows and Linux platforms
• Make the COBOL applications shine in the Cloud	• Thanks to its extensible and pluggable architecture, CLDSQL enables applications to easily connect with databases residing in a shared or private cloud environment. Ideal for containerized applications, CLDSQL supports the major databases on standard cloud architectures