



General SQL Parser Java version Developer's Guide

Version 1.1
Release Date 19 October 2015

Gudu Software Limited
<http://www.sqlparser.com>

DOCUMENT INFORMATION

Gudu Software Limited

General SQL Parser Java version Developer's Guide

Version 1.1

Printed 19 October 2015

Gudu Software Support Information

For support queries, please contact us at info@sqlparser.com.

Access the Gudu Software Web site at the following URL:

www.sqlparser.com

Copyright

Copyright © 2002-2015 Gudu Software, Inc. All rights reserved.

Trademarks

All trademarks and registered trademarks are the property of their respective owners.

Confidentiality

CONFIDENTIAL AND PROPRIETARY INFORMATION OF Gudu Software. The information set forth herein represents the confidential and proprietary information of Gudu Software. Such information shall only be used for the express purpose authorized by Gudu Software and shall not be published, communicated, disclosed or divulged to any person, firm, corporation or legal entity, directly or indirectly, or to any third person without the prior written consent of Gudu Software.

Disclaimer

Gudu Software provides this publication "as is" without warranty of any kind, either express or implied. In no event shall Gudu Software be liable for any loss of profits, loss of business, loss of use or data, interruption of business, or for indirect, special, punitive, incidental, or consequential damages of any kind.

No part of this work covered by copyright herein may be reproduced in any form or by any means—graphic, electronic, or mechanical—including photocopying, recording, taping, or storage in an information retrieval system, without prior written permission of the copyright owner.

This publication is subject to replacement by a later edition. To determine if a later edition exists, contact www.sqlparser.com.

TABLE OF CONTENTS

Document information	2
<i>Table of Contents</i>	3
<i>Preface</i>	5
Conventions	5
Contacts/Reporting problems	7
<i>introducing the General SQL Parser</i>	9
About the General SQL Parser	9
Problems the General SQL Parser Solves	10
Architecture of General SQL Parser	11
General concepts in General SQL Parser	11
GSP Library Components	12
Comparing the General SQL Parser to other products	12
<i>working with the General SQL Parser</i>	13
Creating parser to process SQL script	13
parse tree navigation	13
Accessing source tokens	15
Major classes and types	15
Major built-In features	16
Other useful demos	17
<i>Installing General SQL Parser</i>	18
Downloading the General SQL Parser	18
Configuring the General SQL Parser	18
purchasing the General SQL Parser	19
<i>Table of Figures</i>	20
<i>Index</i>	21

PREFACE

This document acts as a guide to using Gudu Software General SQL Parser for parsing, formatting, modifying, and analyzing the SQL code used to manipulate the data stored in a database.

Target Audience

This document is intended for all users of General SQL Parser, including the following:

- Programmers who develop database-related tools that analyze SQL queries.
- Programmers familiar with Java and SQL for various databases .
- Users of business intelligence or data mining, who perform table or column compact analysis or any other SQL-related analysis.

Pre-Requisite

It is assumed at this point that you have installed Gudu Software General SQL Parser on your machine.

CONVENTIONS

The following tables list the various conventions used in Gudu Software documentation. We follow these conventions to help you quickly and easily identify particular elements, processes, and names that occur frequently in documents.

Typographical conventions



This guide uses the following typographical conventions:

Convention	Description
------------	-------------

Convention	Description
Bold text	Indicates one of the following: <ul style="list-style-type: none">• Screen element• New terminology• A file or folder name• A control in an application's user interface• A registry key• Important information
<i>Italic text</i>	Indicates a reference or the title of a publication.
Monospaced text	Indicates code examples or system messages.
Monospaced bold text	Indicates system commands that you enter.
<i>Hyperlink</i>	Indicates an Internet link to target material.

Graphical conventions

This guide uses the following graphical conventions:

Convention	Description
	Indicates additional information that may be of interest to the reader.
	Indicates cautions that, if ignored, can result in damage to software or hardware.

CONTACTS/REPORTING PROBLEMS

These sections present contact information for a variety of situations.

Sales

For any sales queries, please contact us at sales@sqlparser.com.

Support

For support queries, please contact us at info@sqlparser.com.

Latest updates and information

For the latest updates and information, please visit us at www.sqlparser.com.

Gudu Software Web site

Access the Gudu Software Web site at the following URL:

www.sqlparser.com

INTRODUCING THE GENERAL SQL PARSER

This chapter covers the following topics:

- What the General SQL Parser is
- What problems the General SQL Parser solves
- Why you would use the General SQL Parser
- Architecture of General SQL Parser
- How the General SQL Parser compares to alternatives

ABOUT THE GENERAL SQL PARSER

General SQL Parser for Java is a Java class library that provides algorithms and components for parsing, analyzing, formatting and modifying SQL query from various database vendors without connecting to a database instance. Its intended use is for database related application developers that are working on projects dealing with SQL queries.

Supported Databases:

- Amazon Redshift
- Apache Hive
- DB2
- EMC Greenplum
- Informix
- Microsoft ACCESS
- Microsoft SQL Server
- MySQL
- Netezza

- Oracle
- PostgreSQL
- Sybase
- Teradata

Why do people choose to use General SQL Parser?

Adding a home-grown SQL parser to your applications is a difficult problem. Not only is decoding SQL grammar difficult, but database vendors are constantly releasing new versions of their databases. Maintaining your own SQL parser is time-consuming, error-prone, and costly.

General SQL Parser helps your applications stay current with the latest versions of database programs.

People can save hundreds of hours of development time, improving your applications and speeding them to your customers.

PROBLEMS THE GENERAL SQL PARSER SOLVES

The General SQL Parser solves many common and difficult SQL related problems, including but not limited to the following:

- Checks SQL syntax offline so that you can validate syntax without connecting to a database.
- Formats SQL with more than 100 highly customizable format options.
- Fetches table and column from complicated SQL query along with operations against those table and columns.
- Rename table and column.
- Gets data lineage by analyzing SQL scripts and doing column impact analysis.
- Prevents SQL injection attacks.
- Rewrites SQL query dynamically inside your application.
- Translates SQL query between different databases.
- Performs an in-depth analysis of SQL scripts and provides full access to the SQL query parse tree nodes.

ARCHITECTURE OF GENERAL SQL PARSER

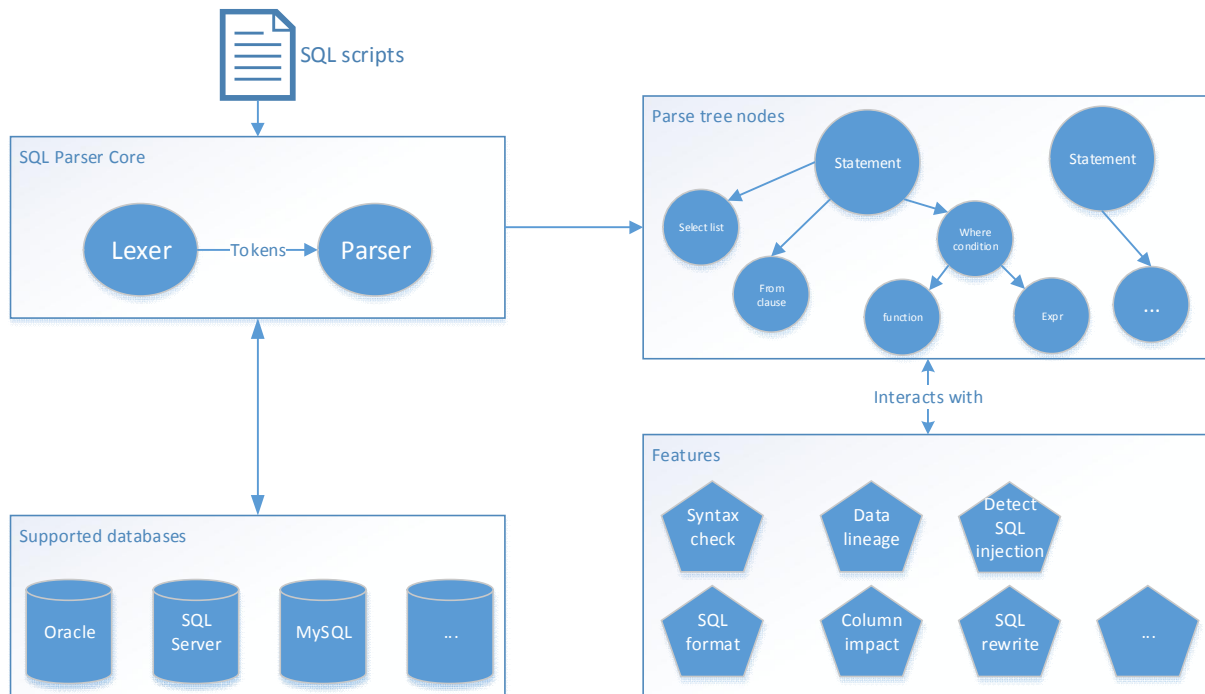


Figure 1: Architecture of General SQL Parser.

GENERAL CONCEPTS IN GENERAL SQL PARSER

Tokens, after reading input SQL script, General SQL parser breaks a stream of text into tokens, usually by looking for whitespace (tabs, spaces, new lines), then attaches extra context such as position to the tokens. Token usually stands for the basic elements of SQL, such as literals, comments, operators and so on.

Parse tree nodes, General SQL Parser then takes the stream of tokens from the lexer and turns it into an abstract syntax tree representing the SQL query represented by the original text. Typical parse tree nodes including select list, from clause, join condition, datatypes and so on.

Statements is the top level parse tree node, it is the first type of parse tree node you get after successfully parsing a SQL script.

GSP LIBRARY COMPONENTS

Java package name	Functionality
gudusoft.gsqlparser	Main parser object, TGSqlParser
gudusoft.gsqlparser.nodes	Common parse tree nodes
gudusoft.gsqlparser.nodes.db_vendor_name	Vendor specific parse tree nodes
gudusoft.gsqlparser.stmt	Common SQL statement objects
gudusoft.gsqlparser.stmt.db_vendor_name	Vendor specific SQL statement objects

COMPARING THE GENERAL SQL PARSER TO OTHER PRODUCTS

When comparing the General SQL Parser to other products, consider the following points:

- Other products are available only for a single language. The General SQL Parser is available for major programming languages, including C#, VB.NET, Java, C/C++, Delphi, and Visual Basic.
- Other products are available only for a single database. The General SQL Parser is available for major databases, including SQL Server, Oracle, DB2, MySQL, Informix, Sybase, Teradata, Netezza, PostgreSQL, Hive, Greenplum, Redshift and Access databases.
- The SQL parsers of database vendors are embedded into the database. There is not API available for use by third-party developers and vendors.
- The General SQL Parser's powerful SQL parse and analysis capabilities enable it to parse complex SQL statements that other parsers cannot handle.
- Makers of other parsers actually recommend the General SQL Parser:

WORKING WITH THE GENERAL SQL PARSER

CREATING PARSER TO PROCESS SQL SCRIPT

Class TGSqlParser offers one constructor to create a new instance with support for specified database vendor.

```
// create a new sql parser with support for Oracle database
TGSqlParser sqlparser = new TGSqlParser(EDbVendor.dbvoracle);
```

After create a new instance of sql parser, set input SQL script.

```
sqlparser.sqltext = "select ename,sal from emp";
```

Then, Parse SQL script

```
int ret = sqlparser.parse();
if (ret == 0){
    System.out.println("Success");
}else{
    System.out.println(sqlparser.getErrorMessage());
}
```

Please make sure always check return value after parsing SQL script and move forward only return value is zero which means SQL script was parsed successfully.

PARSE TREE NAVIGATION

After successfully parse a SQL script. We can fetch detailed information about this SQL script including but not limited to SQL type, tables, columns and so on.

```
int ret = sqlparser.parse();
if (ret == 0){
    for(int i=0;i<sqlparser.sqlstatements.size();i++){
        // process each SQL statement
        analyzeStmt(sqlparser.sqlstatements.get(i));
    }
}
```

```

    }
} else {
    System.out.println(sqlparser.getErrorMessage());
}

```

Statement is always the first parse tree node we get from the parser, since there are hundreds type of SQL statement, the first thing is to do is determine the type of fetched SQL statement by checking `sqlstatementtype` which type of `ESqlStatementType`.

For detailed information about each SQL statement, please check API document.

Visitor pattern

General SQL Parser supports visitor pattern to help you iterate all parse tree nodes quickly and easily. In order to use visitor pattern, you need to create a class descend from `TParseTreeVisitor` like this:

```
class sampleVisitor extends TParseTreeVisitor
```

Then create an instance of this class pass to `accept` method of parse tree node class.

```

sampleVisitor sv = new sampleVisitor();
for(int i=0;i<sqlparser.sqlstatements.size();i++){
    sqlparser.sqlstatements.get(i).accept(sv);
}

```

There are two skeleton methods in `TParseTreeVisitor` for each parse tree node. Such as

```

public void preVisit(TResultColumn node){}
public void postVisit(TResultColumn node){}

```

You can add your own business code into these two skeleton methods to achieve what you need.

For example, if you only want to process select statement, then you can add your own code to these 2 skeleton methods.

```

public void preVisit(TSelectSqlStatement stmt){}
public void postVisit(TSelectSqlStatement stmt){}

```

Expression visitor

Due to the complexity of expression in SQL. GSP provides additional expression visitor to traverse expression in pre/in/post order.

```
public interface IExpressionVisitor {
    public boolean exprVisit(TParseTreeNode pNode,boolean isLeafNode);
}
```

We can create special visitor based on `IExpressionVisitor` to traverse expression easily in pre/in/post order.

For example, you can create a column visitor to find out all columns inside an expression like this:

```
columnVisitor cv = new columnVisitor(statement);
expression.postOrderTraverse(cv);
```

Please check `columnVisitor` in shipped demo for more information.

ACCESSING SOURCE TOKENS

Tokens generated by SQL parser for input SQL script is stored in `sourcetokenlist` which is type of `TSourceTokenList` of class `TGSqlParser`. Even if there are syntax errors in input SQL script, you can always access source tokens generated by lexer of this SQL parser. This token list includes tokens of all SQL statements if there is more than one statement in the script. You can fetch tokens for each SQL statement in `sourcetokenlist` of class `TCustomSqlStatement`.

MAJOR CLASSES AND TYPES

class `TGSqlParser`

This is the main class of GSP library. Use instance of this class to parse SQL query and generate parse tree nodes for further processing.

enum `EDbVendor`

This is the type of database vendors, such as Oracle, SQL Server, MySQL and so on. Must be used when initialize an instance of `TGsqlParser` class.

class `TCustomSqlStatement`

This is the base class of all SQL statements. Descendant classes such as `TSelectSqlStatement`, `TUpdateSqlStatement`, `TTruncateStatement`, `TDeleteSqlStatement`, `TInsertSqlStatement` and other statement class represents specific SQL statement accordingly.

enum ESqlStatementType

This is enum value to determine the type of a SQL statement.

abstract class TParseTreeNode

This is the base class of all parse tree nodes. Descendant classes such as `TAliasClause`, `TColumnDefinition`, `TConstant`, `TExpression` and other class represents specific SQL clauses accordingly.

enum ENodeType

This is enum value to determine the type of a SQL clause.

class TSourceToken

This is the class represents source token of input SQL query.

enum ETokenType

This is enum value to determine the type of source token.

Class TExpression

This is the class represents SQL expression.

enum EExpressionType

This is enum value to determine the type of SQL expression.

Please check Javadoc for more class and types information.

MAJOR BUILT-IN FEATURES

With fully accessible parse tree nodes of decoded SQL script, you can do any analysis against parsed SQL script as you like. In order to save your time, we provided a bunch of built-in functions that you help you to achieve what you need quickly and easily. All those demos can be found under demo directory shipped together with this library.

Offline SQL Syntax Check

With our vendor-specific offline SQL syntax check, you can validate SQL syntax without connecting to the database server. Syntax errors can be detected before executing SQL on your production database server. This is especially useful if your SQL was dynamically built based on user input.

SQL Formatter

You can easily integrate our SQL formatter into your application to generate a color-coded, professional, and intuitive SQL layout that gives your product a professional look and feel. Our highly customizable SQL formatter offers more than 100 format options, including text, HTML output.

We know how difficult it is to build a flexible and stable SQL formatter. If you need to support more than one SQL dialect – such as Oracle, SQL Server, DB2, and MySQL – then this library could be the smartest investment you ever make, saving you hundreds of hours of coding.

Extract and rename table and column

Precisely determining and renaming every table and column in SQL scripts that include considerable nesting sub-queries is a complex procedure. Determine which tables and column have Create, Read, Update, Delete, and Insert operations against them.

Data lineage and column impact analyses

Analyzes contents of scripts, extract data lineage and data flows from scripts, revealing the path of data between database tables, views, individual columns. Perform reliable impact analyses, and trace data to their origin.

Avoid SQL Injection Attacks

Worried about being vulnerable to SQL injection attacks on your Java applications? Our operational library automatically detects malicious SQL segments.

OTHER USEFUL DEMOS

We have created many useful demos to help you make better use of this library. In addition, we frequently add demos upon request from users. You can find more demos at <http://www.dpriver.com/blog/list-of-demos-illustrate-how-to-use-general-sql-parser/>.

INSTALLING GENERAL SQL PARSER

This chapter covers the following topics:

- How to download or purchase the General SQL Parser
- How to install the General SQL Parser
- How to configure the General SQL Parser

DOWNLOADING THE GENERAL SQL PARSER

You can download a FREE Trial Version of the General SQL Parser for evaluation purposes. The FREE Trial Version processes SQL queries with fewer than 10,000 characters. The FREE Trial Version expires 90 days after downloading. Visit the <http://www.sqlparser.com/download.php> web site.

CONFIGURING THE GENERAL SQL PARSER

The configuration of the General SQL Parser depends on the development environment. This section gives instructions on configuring the General SQL Parser for several development environments.

Configuring the General SQL Parser for a Java development environment

To configure the General SQL Parser for a Java development environment, follow these instructions:

1. Ensure that you have an up-to-date version of the Java JDK installed on your computer. Visit the <http://www.oracle.com/technetwork/java/javase/downloads/index.html> web site. Select the JDK download. On the next page, select your computer's operating system. Download and install the Java JDK as instructed.
2. Open a DOS command window, also called a Command Prompt window.
3. Navigate to the `dist` folder. If you unzipped the contents of the zip file to the folder `C:\tmp`, then the `dist` folder is located at `C:\tmp\gsp_java_trial\dist\`.
4. Enter these commands at the DOS command prompt:

```
set java_home="C:\Program Files\Java\<jdk_folder>"  
where jdk_folder is the folder created when installing the Java JDK in step 1 above  
set path=%path%;%java_home%\bin;
```

```
set classpath=%classpath%;%java_home%\jre\lib\rt.jar;.\gsp.jar
```

5. To compile one of the included demos, follow these instructions:
 - a. The included demo programs are in the folder
C:\tmp\gsp_java_trial\dist\demos. For this procedure, we use the checksyntax demo program.
 - b. Enter this command at the DOS command prompt:
javac demos\checksyntax\checksyntax.java
 - c. Create a simple test SQL file called test.sql in the
C:\tmp\gsp_java_trial\dist folder containing this sample SQL statement:
select count(*) from tab
 - d. Run the compiled checksyntax demo program with the test.sql file by entering this command at the DOS command prompt:
java demos.checksyntax.checksyntax test.sql
 - e. The demo program gives output such as “Check syntax ok!”.
6. You can compile other demos, or your own .java programs, similarly.

PURCHASING THE GENERAL SQL PARSER

To purchase license for the General SQL Parser, follow these instructions: Visit the <http://www.sqlparser.com> web site.

TABLE OF FIGURES

Figure 1: Architecture of General SQL Parser..... 11

INDEX

Access, 2, 7, 12
C, 12, 18, 19
C++, 12
contact information, 7
DB2, 12, 17
Delphi, 12
demos, 17, 19
event, 2
Java, 5, 12, 17, 18
JDK, 18
MySQL, 12, 17
Oracle, 12, 17
parse tree, 10
Parser, 1, 2, 5, 9, 10, 11, 12, 13, 18, 19
PostgreSQL, 12
scripts, 10
segments, 17
SQL grammar, 10
SQL injection, 10, 17
SQL Server, 5, 12, 17
syntax, 10, 17, 19
Teradata, 12
Trial Version, 18
Visual Basic, 12