

general\_ledger

Generated by Doxygen 1.8.1.2

Thu Jun 19 2014 23:29:13



# Contents

<b>1</b>	<b>General Ledger.</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Bug List</b>	<b>5</b>
<b>4</b>	<b>Module Index</b>	<b>7</b>
4.1	Modules . . . . .	7
<b>5</b>	<b>Class Index</b>	<b>9</b>
5.1	Class Hierarchy . . . . .	9
<b>6</b>	<b>Class Index</b>	<b>11</b>
6.1	Class List . . . . .	11
<b>7</b>	<b>File Index</b>	<b>13</b>
7.1	File List . . . . .	13
<b>8</b>	<b>Module Documentation</b>	<b>15</b>
8.1	Program configuration module . . . . .	15
8.1.1	Detailed Description . . . . .	15
8.2	Database interaction module . . . . .	16
8.2.1	Detailed Description . . . . .	17
8.2.2	Function Documentation . . . . .	17
8.2.2.1	get_connection . . . . .	17
8.2.2.2	get_database_type . . . . .	17
8.2.2.3	get_field_names . . . . .	17
8.2.2.4	get_row . . . . .	17
8.3	SQL statements module . . . . .	19
8.3.1	Detailed Description . . . . .	19
8.4	General Ledger database module. . . . .	20
8.4.1	Detailed Description . . . . .	20
8.4.2	Function Documentation . . . . .	20
8.4.2.1	decorated_report_from_table . . . . .	20

8.4.2.2	<a href="#">decorated_row</a>	21
8.4.2.3	<a href="#">grow_widths</a>	21
8.4.2.4	<a href="#">max_column_widths</a>	21
8.4.2.5	<a href="#">plain_report_from_table</a>	21
8.4.2.6	<a href="#">plain_row</a>	21
8.4.2.7	<a href="#">separator_row</a>	22
8.5	General purpose utilities.	23
8.5.1	Detailed Description	23
8.5.2	Function Documentation	23
8.5.2.1	<a href="#">content_lines</a>	23
8.5.2.2	<a href="#">join</a>	23
8.5.2.3	<a href="#">next_content_line</a>	24
8.5.2.4	<a href="#">replace</a>	24
8.5.2.5	<a href="#">split</a>	24
8.5.2.6	<a href="#">split</a>	25
8.5.2.7	<a href="#">split_lines</a>	25
8.5.2.8	<a href="#">trim</a>	25
8.5.2.9	<a href="#">trim_back</a>	25
8.5.2.10	<a href="#">trim_front</a>	26
8.6	Database program.	27
8.6.1	Detailed Description	27
8.6.2	Function Documentation	27
8.6.2.1	<a href="#">check_db_parameters</a>	27
8.6.2.2	<a href="#">check_help_and_version</a>	27
8.6.2.3	<a href="#">login</a>	28
8.6.2.4	<a href="#">main</a>	28
8.6.2.5	<a href="#">set_configuration</a>	28
8.7	Reporting program.	29
8.7.1	Detailed Description	29
8.7.2	Function Documentation	29
8.7.2.1	<a href="#">check_db_parameters</a>	29
8.7.2.2	<a href="#">check_help_and_version</a>	29
8.7.2.3	<a href="#">login</a>	30
8.7.2.4	<a href="#">main</a>	30
8.7.2.5	<a href="#">set_configuration</a>	30
8.8	User administration program.	31
8.8.1	Detailed Description	31
8.8.2	Function Documentation	31
8.8.2.1	<a href="#">check_db_parameters</a>	31
8.8.2.2	<a href="#">check_help_and_version</a>	32

8.8.2.3	<a href="#">check_user_password</a>	32
8.8.2.4	<a href="#">enable_user</a>	32
8.8.2.5	<a href="#">get_user</a>	32
8.8.2.6	<a href="#">login</a>	32
8.8.2.7	<a href="#">main</a>	33
8.8.2.8	<a href="#">set_configuration</a>	33
8.8.2.9	<a href="#">set_user_password</a>	33
8.8.2.10	<a href="#">show_user_details</a>	33
<b>9</b>	<b>Class Documentation</b>	<b>35</b>
9.1	<a href="#">genleg::Config Class Reference</a>	35
9.1.1	<a href="#">Detailed Description</a>	35
9.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	35
9.1.2.1	<a href="#">Config</a>	35
9.1.2.2	<a href="#">~Config</a>	36
9.1.3	<a href="#">Member Function Documentation</a>	36
9.1.3.1	<a href="#">add_cmdline_option</a>	36
9.1.3.2	<a href="#">is_set</a>	36
9.1.3.3	<a href="#">operator[]</a>	36
9.1.3.4	<a href="#">populate_from_cmdline</a>	36
9.1.3.5	<a href="#">populate_from_file</a>	37
9.1.4	<a href="#">Member Data Documentation</a>	37
9.1.4.1	<a href="#">m_opts_set</a>	37
9.1.4.2	<a href="#">m_opts_supp</a>	37
9.2	<a href="#">genleg::ConfigBadConfigFile Class Reference</a>	37
9.2.1	<a href="#">Detailed Description</a>	38
9.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	38
9.2.2.1	<a href="#">ConfigBadConfigFile</a>	38
9.3	<a href="#">genleg::ConfigBadOption Class Reference</a>	39
9.3.1	<a href="#">Detailed Description</a>	39
9.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	39
9.3.2.1	<a href="#">ConfigBadOption</a>	40
9.4	<a href="#">genleg::ConfigCouldNotOpenFile Class Reference</a>	40
9.4.1	<a href="#">Detailed Description</a>	41
9.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	41
9.4.2.1	<a href="#">ConfigCouldNotOpenFile</a>	41
9.5	<a href="#">genleg::ConfigException Class Reference</a>	41
9.5.1	<a href="#">Detailed Description</a>	41
9.5.2	<a href="#">Constructor &amp; Destructor Documentation</a>	42
9.5.2.1	<a href="#">ConfigException</a>	42

9.6	<a href="#">genleg::ConfigOptionNotSet Class Reference</a>	42
9.6.1	<a href="#">Detailed Description</a>	43
9.6.2	<a href="#">Constructor &amp; Destructor Documentation</a>	43
9.6.2.1	<a href="#">ConfigOptionNotSet</a>	43
9.7	<a href="#">gldb::DBConn Class Reference</a>	43
9.7.1	<a href="#">Detailed Description</a>	44
9.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	44
9.7.2.1	<a href="#">DBConn</a>	44
9.7.2.2	<a href="#">DBConn</a>	44
9.7.2.3	<a href="#">DBConn</a>	44
9.7.3	<a href="#">Member Function Documentation</a>	44
9.7.3.1	<a href="#">operator=</a>	44
9.7.3.2	<a href="#">operator=</a>	44
9.7.3.3	<a href="#">query</a>	44
9.7.3.4	<a href="#">select</a>	45
9.7.4	<a href="#">Member Data Documentation</a>	45
9.7.4.1	<a href="#">m_imp</a>	45
9.8	<a href="#">gldb::DBConnCouldNotConnect Class Reference</a>	45
9.8.1	<a href="#">Detailed Description</a>	46
9.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	46
9.8.2.1	<a href="#">DBConnCouldNotConnect</a>	46
9.9	<a href="#">gldb::DBConnCouldNotQuery Class Reference</a>	46
9.9.1	<a href="#">Detailed Description</a>	47
9.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	47
9.9.2.1	<a href="#">DBConnCouldNotQuery</a>	47
9.10	<a href="#">gldb::DBConnDummy Class Reference</a>	48
9.10.1	<a href="#">Detailed Description</a>	49
9.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	49
9.10.2.1	<a href="#">DBConnDummy</a>	49
9.10.2.2	<a href="#">DBConnDummy</a>	49
9.10.2.3	<a href="#">~DBConnDummy</a>	49
9.10.3	<a href="#">Member Function Documentation</a>	49
9.10.3.1	<a href="#">operator=</a>	49
9.10.3.2	<a href="#">query</a>	49
9.10.3.3	<a href="#">select</a>	49
9.11	<a href="#">gldb::DBConnException Class Reference</a>	50
9.11.1	<a href="#">Detailed Description</a>	50
9.11.2	<a href="#">Constructor &amp; Destructor Documentation</a>	50
9.11.2.1	<a href="#">DBConnException</a>	50
9.12	<a href="#">gldb::DBConnImp Class Reference</a>	51

9.12.1 Detailed Description . . . . .	51
9.12.2 Constructor & Destructor Documentation . . . . .	51
9.12.2.1 DBConnImp . . . . .	51
9.12.2.2 ~DBConnImp . . . . .	51
9.12.3 Member Function Documentation . . . . .	52
9.12.3.1 query . . . . .	52
9.12.3.2 select . . . . .	52
9.13 glDb::DBConnMySQL Class Reference . . . . .	52
9.13.1 Detailed Description . . . . .	53
9.13.2 Constructor & Destructor Documentation . . . . .	53
9.13.2.1 DBConnMySQL . . . . .	53
9.13.2.2 DBConnMySQL . . . . .	54
9.13.2.3 DBConnMySQL . . . . .	54
9.13.2.4 ~DBConnMySQL . . . . .	54
9.13.3 Member Function Documentation . . . . .	54
9.13.3.1 operator= . . . . .	54
9.13.3.2 operator= . . . . .	54
9.13.3.3 query . . . . .	54
9.13.3.4 select . . . . .	54
9.13.4 Member Data Documentation . . . . .	55
9.13.4.1 m_conn . . . . .	55
9.14 genleg::DBSQLDummy Class Reference . . . . .	55
9.14.1 Detailed Description . . . . .	56
9.15 genleg::DBSQLMySQL Class Reference . . . . .	56
9.15.1 Detailed Description . . . . .	56
9.16 genleg::DBSQLStatements Class Reference . . . . .	57
9.16.1 Detailed Description . . . . .	58
9.16.2 Constructor & Destructor Documentation . . . . .	58
9.16.2.1 DBSQLStatements . . . . .	58
9.16.2.2 ~DBSQLStatements . . . . .	58
9.16.3 Member Function Documentation . . . . .	58
9.16.3.1 create_table . . . . .	58
9.16.3.2 create_view . . . . .	58
9.16.3.3 currenttb . . . . .	58
9.16.3.4 currenttb_by_entity . . . . .	58
9.16.3.5 drop_table . . . . .	59
9.16.3.6 drop_view . . . . .	59
9.16.3.7 get_perms . . . . .	59
9.16.3.8 grant . . . . .	59
9.16.3.9 revoke . . . . .	60

9.16.3.10	update_user	60
9.16.3.11	user_by_id	60
9.16.3.12	user_by_username	60
9.17	genleg::GLDatabase Class Reference	61
9.17.1	Detailed Description	62
9.17.2	Constructor & Destructor Documentation	62
9.17.2.1	GLDatabase	62
9.17.2.2	~GLDatabase	63
9.17.3	Member Function Documentation	63
9.17.3.1	backend	63
9.17.3.2	create_structure	63
9.17.3.3	create_user	63
9.17.3.4	current_trial_balance_report	63
9.17.3.5	destroy_structure	63
9.17.3.6	get_user_by_id	64
9.17.3.7	get_user_by_username	64
9.17.3.8	grant	64
9.17.3.9	load_sample_data	64
9.17.3.10	report	65
9.17.3.11	revoke	65
9.17.3.12	update_user	65
9.17.4	Member Data Documentation	65
9.17.4.1	m_dbc	65
9.17.4.2	m_sql	65
9.17.4.3	m_tables	65
9.17.4.4	m_views	65
9.18	genleg::GLDBException Class Reference	66
9.18.1	Detailed Description	66
9.18.2	Constructor & Destructor Documentation	66
9.18.2.1	GLDBException	66
9.19	genleg::GLReport Class Reference	66
9.19.1	Detailed Description	67
9.19.2	Constructor & Destructor Documentation	67
9.19.2.1	GLReport	67
9.19.2.2	~GLReport	67
9.19.3	Friends And Related Function Documentation	67
9.19.3.1	operator<<	67
9.19.4	Member Data Documentation	67
9.19.4.1	m_report_text	67
9.20	genleg::GLUser Class Reference	67



9.20.1 Detailed Description . . . . .	69
9.20.2 Constructor & Destructor Documentation . . . . .	69
9.20.2.1 GLUser . . . . .	69
9.20.2.2 ~GLUser . . . . .	69
9.20.3 Member Function Documentation . . . . .	69
9.20.3.1 check_password . . . . .	69
9.20.3.2 enabled . . . . .	70
9.20.3.3 firstname . . . . .	70
9.20.3.4 id . . . . .	70
9.20.3.5 lastname . . . . .	70
9.20.3.6 pass_hash . . . . .	70
9.20.3.7 pass_salt . . . . .	70
9.20.3.8 permissions . . . . .	71
9.20.3.9 set_enabled . . . . .	71
9.20.3.10 set_firstname . . . . .	71
9.20.3.11 set_lastname . . . . .	71
9.20.3.12 set_password . . . . .	71
9.20.3.13 set_username . . . . .	71
9.20.3.14 username . . . . .	71
9.20.4 Member Data Documentation . . . . .	72
9.20.4.1 m_enabled . . . . .	72
9.20.4.2 m_firstname . . . . .	72
9.20.4.3 m_id . . . . .	72
9.20.4.4 m_lastname . . . . .	72
9.20.4.5 m_pass_hash . . . . .	72
9.20.4.6 m_pass_salt . . . . .	72
9.20.4.7 m_perms . . . . .	72
9.20.4.8 m_username . . . . .	72
9.21 glldb::MySQLResult Class Reference . . . . .	72
9.21.1 Detailed Description . . . . .	73
9.21.2 Constructor & Destructor Documentation . . . . .	73
9.21.2.1 MySQLResult . . . . .	73
9.21.2.2 ~MySQLResult . . . . .	73
9.21.2.3 MySQLResult . . . . .	73
9.21.2.4 MySQLResult . . . . .	73
9.21.3 Member Function Documentation . . . . .	73
9.21.3.1 num_fields . . . . .	74
9.21.3.2 operator= . . . . .	74
9.21.3.3 operator= . . . . .	74
9.21.3.4 result . . . . .	74

9.21.4	Member Data Documentation . . . . .	74
9.21.4.1	m_num_fields . . . . .	74
9.21.4.2	m_result . . . . .	74
9.22	gldb::Table Class Reference . . . . .	74
9.22.1	Detailed Description . . . . .	76
9.22.2	Constructor & Destructor Documentation . . . . .	76
9.22.2.1	Table . . . . .	76
9.22.2.2	Table . . . . .	76
9.22.2.3	Table . . . . .	76
9.22.2.4	Table . . . . .	77
9.22.2.5	~Table . . . . .	77
9.22.3	Member Function Documentation . . . . .	77
9.22.3.1	append_record . . . . .	77
9.22.3.2	append_record . . . . .	77
9.22.3.3	begin . . . . .	77
9.22.3.4	begin . . . . .	77
9.22.3.5	create_from_file . . . . .	78
9.22.3.6	end . . . . .	78
9.22.3.7	end . . . . .	78
9.22.3.8	get_field . . . . .	78
9.22.3.9	get_headers . . . . .	79
9.22.3.10	insert_query . . . . .	79
9.22.3.11	num_fields . . . . .	79
9.22.3.12	num_records . . . . .	79
9.22.3.13	operator= . . . . .	79
9.22.3.14	operator= . . . . .	79
9.22.3.15	operator[] . . . . .	80
9.22.3.16	set_quoted . . . . .	80
9.22.3.17	set_quoted . . . . .	80
9.22.4	Member Data Documentation . . . . .	80
9.22.4.1	m_headers . . . . .	80
9.22.4.2	m_quoted . . . . .	80
9.22.4.3	m_records . . . . .	80
9.23	gldb::TableBadInputFile Class Reference . . . . .	81
9.23.1	Detailed Description . . . . .	81
9.23.2	Constructor & Destructor Documentation . . . . .	81
9.23.2.1	TableBadInputFile . . . . .	82
9.24	gldb::TableCouldNotOpenInputFile Class Reference . . . . .	82
9.24.1	Detailed Description . . . . .	83
9.24.2	Constructor & Destructor Documentation . . . . .	83

9.24.2.1	TableCouldNotOpenInputFile	83
9.25	gldb::TableException Class Reference	83
9.25.1	Detailed Description	84
9.25.2	Constructor & Destructor Documentation	84
9.25.2.1	TableException	84
9.26	gldb::TableField Class Reference	84
9.26.1	Detailed Description	85
9.26.2	Constructor & Destructor Documentation	85
9.26.2.1	TableField	85
9.26.2.2	TableField	85
9.26.2.3	TableField	86
9.26.2.4	TableField	86
9.26.2.5	TableField	86
9.26.2.6	~TableField	86
9.26.3	Member Function Documentation	86
9.26.3.1	length	86
9.26.3.2	operator std::string	86
9.26.3.3	operator+=	86
9.26.3.4	operator+=	87
9.26.3.5	operator=	87
9.26.3.6	operator=	87
9.26.3.7	operator=	87
9.26.3.8	operator=	88
9.26.3.9	operator=	88
9.26.3.10	operator[]	88
9.26.3.11	operator[]	88
9.26.4	Friends And Related Function Documentation	89
9.26.4.1	operator<<	89
9.26.5	Member Data Documentation	89
9.26.5.1	m_data	89
9.27	gldb::TableMismatchedRecordLength Class Reference	89
9.27.1	Detailed Description	90
9.27.2	Constructor & Destructor Documentation	90
9.27.2.1	TableMismatchedRecordLength	90
9.28	gldb::TableNoSuchField Class Reference	90
9.28.1	Detailed Description	91
9.28.2	Constructor & Destructor Documentation	91
9.28.2.1	TableNoSuchField	91
9.29	gldb::TableNoSuchRecord Class Reference	92
9.29.1	Detailed Description	92

9.29.2	Constructor & Destructor Documentation	92
9.29.2.1	TableNoSuchRecord	93
9.30	gldb::TableRow Class Reference	93
9.30.1	Detailed Description	94
9.30.2	Constructor & Destructor Documentation	94
9.30.2.1	TableRow	94
9.30.2.2	TableRow	94
9.30.2.3	TableRow	94
9.30.2.4	TableRow	94
9.30.2.5	TableRow	95
9.30.2.6	TableRow	95
9.30.2.7	TableRow	95
9.30.2.8	~TableRow	95
9.30.3	Member Function Documentation	95
9.30.3.1	append_field	95
9.30.3.2	append_field	95
9.30.3.3	append_field	96
9.30.3.4	append_field	96
9.30.3.5	append_field	96
9.30.3.6	begin	96
9.30.3.7	begin	96
9.30.3.8	end	96
9.30.3.9	end	96
9.30.3.10	operator=	97
9.30.3.11	operator=	97
9.30.3.12	operator[]	97
9.30.3.13	operator[]	97
9.30.3.14	print	98
9.30.3.15	record_string	98
9.30.3.16	record_string	98
9.30.3.17	size	98
9.30.4	Member Data Documentation	98
9.30.4.1	m_fields	98
<b>10</b>	<b>File Documentation</b>	<b>99</b>
10.1	lib/config/config.cpp File Reference	99
10.1.1	Detailed Description	99
10.2	lib/config/config.h File Reference	100
10.2.1	Detailed Description	101
10.3	lib/config/config_getopt.cpp File Reference	101

10.3.1 Detailed Description . . . . .	101
10.3.2 Macro Definition Documentation . . . . .	102
10.3.2.1 _XOPEN_SOURCE . . . . .	102
10.4 lib/database/data_structures.h File Reference . . . . .	102
10.4.1 Detailed Description . . . . .	103
10.5 lib/database/database.h File Reference . . . . .	103
10.5.1 Detailed Description . . . . .	105
10.6 lib/database/dbconn.cpp File Reference . . . . .	105
10.6.1 Detailed Description . . . . .	105
10.7 lib/database/dbconn.h File Reference . . . . .	106
10.7.1 Detailed Description . . . . .	107
10.8 lib/database/dbconnimp.h File Reference . . . . .	107
10.8.1 Detailed Description . . . . .	109
10.9 lib/database/table.cpp File Reference . . . . .	109
10.9.1 Detailed Description . . . . .	109
10.10lib/database/table.h File Reference . . . . .	110
10.10.1 Detailed Description . . . . .	111
10.11lib/database/tablefield.cpp File Reference . . . . .	112
10.11.1 Detailed Description . . . . .	112
10.12lib/database/tablefield.h File Reference . . . . .	112
10.12.1 Detailed Description . . . . .	114
10.13lib/database/tablerow.cpp File Reference . . . . .	114
10.13.1 Detailed Description . . . . .	114
10.14lib/database/tablerow.h File Reference . . . . .	115
10.14.1 Detailed Description . . . . .	116
10.15lib/database_imp/database_imp.h File Reference . . . . .	116
10.15.1 Detailed Description . . . . .	118
10.16lib/database_imp/dummy/dbconn_dummy_imp.cpp File Reference . . . . .	118
10.16.1 Detailed Description . . . . .	119
10.17lib/database_imp/dummy/dbconn_dummy_imp.h File Reference . . . . .	119
10.17.1 Detailed Description . . . . .	121
10.18lib/database_imp/mysql/dbconn_mysql_functions.cpp File Reference . . . . .	121
10.18.1 Detailed Description . . . . .	122
10.19lib/database_imp/mysql/dbconn_mysql_imp.cpp File Reference . . . . .	122
10.19.1 Detailed Description . . . . .	123
10.20lib/database_imp/mysql/dbconn_mysql_imp.h File Reference . . . . .	123
10.20.1 Detailed Description . . . . .	125
10.21lib/database_imp/mysql/dbconn_mysql_result.cpp File Reference . . . . .	125
10.21.1 Detailed Description . . . . .	126
10.22lib/database_imp/mysql/dbconn_mysql_result.h File Reference . . . . .	126

10.22.1 Detailed Description . . . . .	127
10.23lib/dbsql/dbsql.h File Reference . . . . .	127
10.23.1 Detailed Description . . . . .	128
10.24lib/dbsql/dbsql_dummy.h File Reference . . . . .	128
10.24.1 Detailed Description . . . . .	129
10.25lib/dbsql/dbsql_implementations.h File Reference . . . . .	130
10.25.1 Detailed Description . . . . .	130
10.26lib/dbsql/dbsql_mysql.h File Reference . . . . .	131
10.26.1 Detailed Description . . . . .	132
10.27lib/dbsql/dbsqlstatements.cpp File Reference . . . . .	132
10.27.1 Detailed Description . . . . .	132
10.28lib/dbsql/dbsqlstatements.h File Reference . . . . .	133
10.28.1 Detailed Description . . . . .	134
10.29lib/gldb/gldatabase.cpp File Reference . . . . .	134
10.29.1 Detailed Description . . . . .	135
10.29.2 Function Documentation . . . . .	135
10.29.2.1 boolstring_to_bool . . . . .	135
10.30lib/gldb/gldatabase.h File Reference . . . . .	135
10.30.1 Detailed Description . . . . .	136
10.31lib/gldb/gldb.h File Reference . . . . .	137
10.31.1 Detailed Description . . . . .	138
10.32lib/gldb/glexception.h File Reference . . . . .	138
10.32.1 Detailed Description . . . . .	139
10.33lib/gldb/glreport.cpp File Reference . . . . .	139
10.33.1 Detailed Description . . . . .	140
10.34lib/gldb/glreport.h File Reference . . . . .	141
10.34.1 Detailed Description . . . . .	142
10.35lib/gldb/gluser.cpp File Reference . . . . .	142
10.35.1 Detailed Description . . . . .	143
10.36lib/gldb/gluser.h File Reference . . . . .	143
10.36.1 Detailed Description . . . . .	144
10.37lib/gldb/gluser_pass.cpp File Reference . . . . .	144
10.37.1 Detailed Description . . . . .	145
10.37.2 Macro Definition Documentation . . . . .	145
10.37.2.1 _XOPEN_SOURCE . . . . .	145
10.37.3 Function Documentation . . . . .	146
10.37.3.1 generate_salt . . . . .	146
10.38lib/pgutils/pgutils.h File Reference . . . . .	146
10.38.1 Detailed Description . . . . .	146
10.39lib/pgutils/stringhelp.cpp File Reference . . . . .	147

10.39.1 Detailed Description . . . . .	147
10.40lib/pgutils/stringhelp.h File Reference . . . . .	147
10.40.1 Detailed Description . . . . .	149
10.41progs/gl_db/gl_db_main.cpp File Reference . . . . .	149
10.41.1 Detailed Description . . . . .	150
10.42progs/gl_report/gl_report_main.cpp File Reference . . . . .	150
10.42.1 Detailed Description . . . . .	152
10.43progs/gl_user/gl_user_main.cpp File Reference . . . . .	152
10.43.1 Detailed Description . . . . .	153





## Chapter 1

# General Ledger.

General Ledger will be a fully-featured, multi-user, open-source general ledger system. The project is in the early stages of development.



## Chapter 2

## Todo List

File [gluser\\_pass.cpp](#)

Implement a better form of password encryption. In particular, these functions are not re-entrant, and only use the first 8 characters of the password.



## Chapter 3

# Bug List

**Member `gldb::Table::Table` (`const Table &table`)**

'explicit' removed from here after failure to compile at end of MySQL query function.



## Chapter 4

# Module Index

### 4.1 Modules

Here is a list of all modules:

Program configuration module . . . . .	15
Database interaction module . . . . .	16
SQL statements module . . . . .	19
General Ledger database module. . . . .	20
General purpose utilities. . . . .	23
Database program. . . . .	27
Reporting program. . . . .	29
User administration program. . . . .	31





## Chapter 5

# Class Index

### 5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

genleg::Config . . . . .	35
genleg::ConfigException . . . . .	41
genleg::ConfigBadConfigFile . . . . .	37
genleg::ConfigBadOption . . . . .	39
genleg::ConfigCouldNotOpenFile . . . . .	40
genleg::ConfigOptionNotSet . . . . .	42
gldb::DBConn . . . . .	43
gldb::DBConnException . . . . .	50
gldb::DBConnCouldNotConnect . . . . .	45
gldb::DBConnCouldNotQuery . . . . .	46
gldb::DBConnImp . . . . .	51
gldb::DBConnDummy . . . . .	48
gldb::DBConnMySQL . . . . .	52
genleg::DBSQLStatements . . . . .	57
genleg::DBSQLDummy . . . . .	55
genleg::DBSQLMySQL . . . . .	56
genleg::GLDatabase . . . . .	61
genleg::GLDBException . . . . .	66
genleg::GLReport . . . . .	66
genleg::GLUser . . . . .	67
gldb::MySQLResult . . . . .	72
gldb::Table . . . . .	74
gldb::TableException . . . . .	83
gldb::TableBadInputFile . . . . .	81
gldb::TableCouldNotOpenInputFile . . . . .	82
gldb::TableMismatchedRecordLength . . . . .	89
gldb::TableNoSuchField . . . . .	90
gldb::TableNoSuchRecord . . . . .	92
gldb::TableField . . . . .	84
gldb::TableRow . . . . .	93



## Chapter 6

# Class Index

### 6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">genleg::Config</a>	Configuration options class . . . . .	35
<a href="#">genleg::ConfigBadConfigFile</a>	Exception class for badly formed configuration file . . . . .	37
<a href="#">genleg::ConfigBadOption</a>	Exception class for bad provided option . . . . .	39
<a href="#">genleg::ConfigCouldNotOpenFile</a>	Exception class for when conf file cannot be opened . . . . .	40
<a href="#">genleg::ConfigException</a>	Configuration module exception base class . . . . .	41
<a href="#">genleg::ConfigOptionNotSet</a>	Exception class for option not set . . . . .	42
<a href="#">gldb::DBConn</a>	Database connection class . . . . .	43
<a href="#">gldb::DBConnCouldNotConnect</a>	Could not connect to database exception class . . . . .	45
<a href="#">gldb::DBConnCouldNotQuery</a>	Could not execute database query exception class . . . . .	46
<a href="#">gldb::DBConnDummy</a>	Dummy database implementation class . . . . .	48
<a href="#">gldb::DBConnException</a>	Base database connection exception class . . . . .	50
<a href="#">gldb::DBConnImp</a>	Abstract database implementation base class . . . . .	51
<a href="#">gldb::DBConnMySQL</a>	MySQL database implementation class . . . . .	52
<a href="#">genleg::DBSQLDummy</a>	Dummy SQL statements class . . . . .	55
<a href="#">genleg::DBSQLMySQL</a>	MySQL SQL statements class . . . . .	56
<a href="#">genleg::DBSQLStatements</a>	SQL statements class . . . . .	57
<a href="#">genleg::GLDatabase</a>	General ledger database class . . . . .	61
<a href="#">genleg::GLDBException</a>	Base general ledger database exceptionc class . . . . .	66
<a href="#">genleg::GLReport</a>	General ledger report class . . . . .	66

<a href="#">genleg::GLUser</a>	
General ledger user class . . . . .	67
<a href="#">gldb::MySQLResult</a>	
MySQL result structure class . . . . .	72
<a href="#">gldb::Table</a>	
Database table class . . . . .	74
<a href="#">gldb::TableBadInputFile</a>	
Could not connect to database exception class . . . . .	81
<a href="#">gldb::TableCouldNotOpenInputFile</a>	
Could not connect to database exception class . . . . .	82
<a href="#">gldb::TableException</a>	
Base database connection exception class . . . . .	83
<a href="#">gldb::TableField</a>	
Database table field class . . . . .	84
<a href="#">gldb::TableMismatchedRecordLength</a>	
Mismatched record length exception class . . . . .	89
<a href="#">gldb::TableNoSuchField</a>	
No such field exception class . . . . .	90
<a href="#">gldb::TableNoSuchRecord</a>	
No such record exception class . . . . .	92
<a href="#">gldb::TableRow</a>	
Database table row class . . . . .	93

## Chapter 7

# File Index

### 7.1 File List

Here is a list of all documented files with brief descriptions:

lib/config/ <a href="#">config.cpp</a>	
Implementation of program configurations class . . . . .	99
lib/config/ <a href="#">config.h</a>	
Interface to program configurations class . . . . .	100
lib/config/ <a href="#">config_getopt.cpp</a>	
Implementation of command line functionality . . . . .	101
lib/database/ <a href="#">data_structures.h</a>	
Main interface to database data structures . . . . .	102
lib/database/ <a href="#">database.h</a>	
User interface to database functionality . . . . .	103
lib/database/ <a href="#">dbconn.cpp</a>	
Implementation of database connection class . . . . .	105
lib/database/ <a href="#">dbconn.h</a>	
Interface to database connection base class . . . . .	106
lib/database/ <a href="#">dbconnimp.h</a>	
Interface to abstract database implementation base class . . . . .	107
lib/database/ <a href="#">table.cpp</a>	
Implementation of database table data structure . . . . .	109
lib/database/ <a href="#">table.h</a>	
Interface to database table data structure . . . . .	110
lib/database/ <a href="#">tablefield.cpp</a>	
Implementation of database table field class . . . . .	112
lib/database/ <a href="#">tablefield.h</a>	
Interface to database table field class . . . . .	112
lib/database/ <a href="#">tablerow.cpp</a>	
Implementation of database table row data structure . . . . .	114
lib/database/ <a href="#">tablerow.h</a>	
Interface to database table row data structure . . . . .	115
lib/database_imp/ <a href="#">database_imp.h</a>	
Interface to database implementation factory function . . . . .	116
lib/database_imp/dummy/ <a href="#">dbconn_dummy_imp.cpp</a>	
Implementation of Dummy database connection implementation class . . . . .	118
lib/database_imp/dummy/ <a href="#">dbconn_dummy_imp.h</a>	
Interface to dummy database connection implementation class . . . . .	119
lib/database_imp/mysql/ <a href="#">dbconn_mysql_functions.cpp</a>	
Implementation of MySQL implementation factory function . . . . .	121
lib/database_imp/mysql/ <a href="#">dbconn_mysql_imp.cpp</a>	
Implementation of MySQL database connection implementation class . . . . .	122

lib/database_imp/mysql/dbconn_mysql_imp.h	
Interface to MySQL database connection implementation class	123
lib/database_imp/mysql/dbconn_mysql_result.cpp	
Implementation of MySQL result structure resource handle class	125
lib/database_imp/mysql/dbconn_mysql_result.h	
Interface to MySQL result structure resource handle class	126
lib/dbsql/dbsql.h	
User interface to DBSQL module	127
lib/dbsql/dbsql_dummy.h	
Interface to dummy SQL statement class	128
lib/dbsql/dbsql_functions.h	??
lib/dbsql/dbsql_implementations.h	
Aggregation header for DBSqlStatements implementations	130
lib/dbsql/dbsql_mysql.h	
Interface to MySQL SQL statement class	131
lib/dbsql/dbsqlstatements.cpp	
Implementation of SQL statement class	132
lib/dbsql/dbsqlstatements.h	
Implementation of SQL module standalone functions	133
lib/gldb/gldatabase.cpp	
Implementation of General Ledger database class	134
lib/gldb/gldatabase.h	
Interface to General Ledger database class	135
lib/gldb/gldb.h	
User interface to General Ledger database module	137
lib/gldb/glexception.h	
Interface to General Ledger base exception class	138
lib/gldb/glreport.cpp	
Implementation of report class	139
lib/gldb/glreport.h	
Interface to report class	141
lib/gldb/gluser.cpp	
Implementation of user class	142
lib/gldb/gluser.h	
Interface to user class	143
lib/gldb/gluser_pass.cpp	
Implementation of password functions for user class	144
lib/pgutils/pgutils.h	
Aggregate interface to general utility functions	146
lib/pgutils/stringhelp.cpp	
Implementation of string helper functions	147
lib/pgutils/stringhelp.h	
Interface to string helper functions	147
progs/gl_db/gl_db_main.cpp	
Main functionality for gl_db program	149
progs/gl_report/gl_report_main.cpp	
Main functionality for gl_report program	150
progs/gl_user/gl_user_main.cpp	
Main functionality for gl_user program	152

## Chapter 8

# Module Documentation

### 8.1 Program configuration module

#### Classes

- class [genleg::ConfigException](#)  
*Configuration module exception base class.*
- class [genleg::ConfigOptionNotSet](#)  
*Exception class for option not set.*
- class [genleg::ConfigBadOption](#)  
*Exception class for bad provided option.*
- class [genleg::ConfigCouldNotOpenFile](#)  
*Exception class for when conf file cannot be opened.*
- class [genleg::ConfigBadConfigFile](#)  
*Exception class for badly formed configuration file.*
- class [genleg::Config](#)  
*Configuration options class.*

#### Enumerations

- enum [genleg::Argument](#)  
*Enumeration class for option argument specifications.*

#### 8.1.1 Detailed Description

Module for getting options from the command line and configuration files.

## 8.2 Database interaction module

### Classes

- class [gldb::DBConnException](#)  
*Base database connection exception class.*
- class [gldb::DBConnCouldNotConnect](#)  
*Could not connect to database exception class.*
- class [gldb::DBConnCouldNotQuery](#)  
*Could not execute database query exception class.*
- class [gldb::DBConn](#)  
*Database connection class.*
- class [gldb::DBConnImp](#)  
*Abstract database implementation base class.*
- class [gldb::TableException](#)  
*Base database connection exception class.*
- class [gldb::TableNoSuchField](#)  
*No such field exception class.*
- class [gldb::TableNoSuchRecord](#)  
*No such record exception class.*
- class [gldb::TableMismatchedRecordLength](#)  
*Mismatched record length exception class.*
- class [gldb::TableBadInputFile](#)  
*Could not connect to database exception class.*
- class [gldb::TableCouldNotOpenInputFile](#)  
*Could not connect to database exception class.*
- class [gldb::Table](#)  
*Database table class.*
- class [gldb::TableField](#)  
*Database table field class.*
- class [gldb::TableRow](#)  
*Database table row class.*
- class [gldb::DBConnDummy](#)  
*Dummy database implementation class.*
- class [gldb::DBConnMySQL](#)  
*MySQL database implementation class.*
- class [gldb::MySQLResult](#)  
*MySQL result structure class.*

### Functions

- [DBConnImp \\* gldb::get\\_connection](#) (const std::string &database, const std::string &hostname, const std::string &username, const std::string &password)  
*Creates and returns a pointer to a database implementation.*
- std::string [gldb::get\\_database\\_type](#) ()  
*Returns the name of the compiled-in database type.*
- static [TableRow get\\_field\\_names](#) (MySQLResult &result)  
*Gets field names from a MySQL result structure.*
- static [TableRow get\\_row](#) (MySQLResult &result, MYSQL\_ROW row)  
*Creates a TableRow from a MySQL result row.*



### 8.2.1 Detailed Description

Module for interacting with the database.

### 8.2.2 Function Documentation

#### 8.2.2.1 DBConnImp \* glldb::get\_connection ( const std::string & database, const std::string & hostname, const std::string & username, const std::string & password )

Creates and returns a pointer to a database implementation.

The implementation of this function is provided by the individual database implementations. One database implementation is compiled into the program at any one time. Multiple database systems are, or will be, supported, and not every system will possess the libraries and headers to compile every implementation. Therefore, only one implementation is compiled in at a time. The fact that each database implementation will implement this function to return the correct derived class prevents any attempt to compile unsupported library code. This would not be feasible if we were to simply provide each implementation as a subclass.

##### Parameters

<i>database</i>	The name of the database to which to connect.
<i>hostname</i>	The hostname of the computer running the database.
<i>username</i>	The username with which to log into the database.
<i>password</i>	The password with which to log into the database.

##### Returns

A pointer to the database implementation.

#### 8.2.2.2 std::string glldb::get\_database\_type ( )

Returns the name of the compiled-in database type.

##### Returns

The name of the compiled-in database type.

#### 8.2.2.3 static TableRow get\_field\_names ( MySQLResult & result ) [static]

Gets field names from a MySQL result structure.

##### Parameters

<i>result</i>	The MySQL result structure.
---------------	-----------------------------

##### Returns

A TableRow containing the field names.

#### 8.2.2.4 static TableRow get\_row ( MySQLResult & result, MYSQL\_ROW row ) [static]

Creates a TableRow from a MySQL result row.

**Parameters**

<i>result</i>	The MySQL result structure.
<i>row</i>	The MySQL row structure.

**Returns**

A TableRow containing the row data.

## 8.3 SQL statements module

### Classes

- class [genleg::DBSQLDummy](#)  
*Dummy SQL statements class.*
- class [genleg::DBSQLMySQL](#)  
*MySQL SQL statements class.*
- class [genleg::DBSQLStatements](#)  
*SQL statements class.*

### 8.3.1 Detailed Description

Module for producing SQL statements used by program.

## 8.4 General Ledger database module.

### Classes

- class `genleg::GLDatabase`  
*General ledger database class.*
- class `genleg::GLDBException`  
*Base general ledger database exception class.*
- class `genleg::GLReport`  
*General ledger report class.*
- class `genleg::GLUser`  
*General ledger user class.*

### Functions

- static `std::vector< size_t > max_column_widths` (const `gldb::Table` &table)  
*Calculates the maximum required column widths for a table.*
- static void `grow_widths` (std::vector< size\_t > &widths, const `TableRow` &row)  
*Increments a vector of required column widths.*
- static `std::string separator_row` (const std::vector< size\_t > &widths)  
*Returns a decorated separator row for a table.*
- static `std::string plain_row` (const `TableRow` &row, const std::vector< size\_t > &widths)  
*Returns a row for a plain report.*
- static `std::string decorated_row` (const `TableRow` &row, const std::vector< size\_t > &widths)  
*Returns a row for a decorated report.*
- `std::string genleg::plain_report_from_table` (const `gldb::Table` &table)  
*Creates a plain report from a table.*
- `std::string genleg::decorated_report_from_table` (const `gldb::Table` &table)  
*Creates a decorated report from a table.*

### 8.4.1 Detailed Description

Module for interacting with the general ledger database model.

### 8.4.2 Function Documentation

#### 8.4.2.1 `std::string genleg::decorated_report_from_table ( const gldb::Table & table )`

Creates a decorated report from a table.

A "decorated report" presents the table surrounding with ASCII-art style lines consisting of '+' , '-' and '|' characters.

#### Parameters

<i>table</i>	The table from which to create the report.
--------------	--

#### Returns

A string containing the report.

8.4.2.2 `static std::string decorated_row ( const TableRow & row, const std::vector< size_t > & widths ) [static]`

Returns a row for a decorated report.

#### Parameters

<i>row</i>	The row for which to create the report row.
<i>widths</i>	A vector of required widths.

#### Returns

A string containing the decorated row.

8.4.2.3 `static void grow_widths ( std::vector< size_t > & widths, const TableRow & row ) [static]`

Increments a vector of required column widths.

Each element of the vector is increased to fit the width of each file in the row, if the existing width is not large enough to contain it.

#### Parameters

<i>widths</i>	An existing vector of widths.
<i>row</i>	The row against which to check and potentially increase the vector.

8.4.2.4 `static std::vector< size_t > max_column_widths ( const gldb::Table & table ) [static]`

Calculates the maximum required column widths for a table.

#### Parameters

<i>table</i>	The table.
--------------	------------

#### Returns

A vector of `size_t` containing the maximum required width for each column, without padding.

8.4.2.5 `std::string genleg::plain_report_from_table ( const gldb::Table & table )`

Creates a plain report from a table.

A "plain report" separates each column with a space.

#### Parameters

<i>table</i>	The table from which to create the report.
--------------	--

#### Returns

A string containing the report.

8.4.2.6 `static std::string plain_row ( const TableRow & row, const std::vector< size_t > & widths ) [static]`

Returns a row for a plain report.

## Parameters

<i>row</i>	The row for which to create the report row.
<i>widths</i>	A vector of required widths.

## Returns

A string containing the plain row.

**8.4.2.7** `static std::string separator_row ( const std::vector< size_t > & widths )` `[static]`

Returns a decorated separator row for a table.

The "separator row" is of the format "+&mdash;+&mdash;+&mdash;+" where each column is separated by a ' +' character, and consists of enough ' - ' characters to fit the respective width in the vector plus two additional characters for spacing.

## Parameters

<i>widths</i>	A vector of required widths.
---------------	------------------------------

## Returns

A string containing the separator row.

## 8.5 General purpose utilities.

### Functions

- `std::string & pgutils::trim_front (std::string &s)`  
*Trims leading whitespace from a string.*
- `std::string & pgutils::trim_back (std::string &s)`  
*Trims trailing whitespace from a string.*
- `std::string & pgutils::trim (std::string &s)`  
*Trims leading and trailing whitespace from a string.*
- `std::vector< std::string > pgutils::split (const std::string &s, const char delim)`  
*Splits a delimited string into tokens.*
- `std::vector< std::string > & pgutils::split (std::vector< std::string > &vec, const std::string &s, const char delim)`  
*Splits a delimited string into tokens.*
- `bool pgutils::next_content_line (std::istream &ifs, std::string &s)`  
*Gets the next content line from a stream.*
- `std::vector< std::string > & pgutils::content_lines (std::vector< std::string > &vec, std::istream &ifs)`  
*Populates a vector of content lines from a stream.*
- `std::vector< std::vector< std::string > > & pgutils::split_lines (std::vector< std::vector< std::string > > &vec, std::istream &ifs, const char delim)`  
*Populates a vector of vectors of fields from a stream.*
- `std::string & pgutils::join (const std::vector< std::string > &vec, std::string &s, const char delim)`  
*Joins a vector of strings into a delimited line.*
- `bool pgutils::replace (std::string &str, const std::string &from, const std::string &to)`  
*Replaces a substring with another string.*

### 8.5.1 Detailed Description

General purpose utility classes and functions.

### 8.5.2 Function Documentation

#### 8.5.2.1 `std::vector< std::string > & pgutils::content_lines ( std::vector< std::string > & vec, std::istream & ifs )`

Populates a vector of content lines from a stream.

##### Parameters

<code>vec</code>	The vector to populate.
<code>ifs</code>	The input stream.

##### Returns

A reference to `vec`.

#### 8.5.2.2 `std::string & pgutils::join ( const std::vector< std::string > & vec, std::string & s, const char delim )`

Joins a vector of strings into a delimited line.

The function is the opposite of `split`.

**Parameters**

<i>vec</i>	The vector containing the strings.
<i>s</i>	The string in which to store the line.
<i>delim</i>	The delimiter character to be used to delimit the fields.

**Returns**

A reference to *s*.

**8.5.2.3 bool pgutils::next\_content\_line ( std::istream & ifs, std::string & s )**

Gets the next content line from a stream.

A "content line" is defined as a line which, after being trimmed of trailing and leading whitespace, is not empty, and does not start with a '#' character (indicating a comment line).

**Parameters**

<i>ifs</i>	The input stream.
<i>s</i>	The string in which to store the line.

**Returns**

`true` if there is a next content line, `false` otherwise.

**8.5.2.4 bool pgutils::replace ( std::string & str, const std::string & from, const std::string & to )**

Replaces a substring with another string.

**Parameters**

<i>str</i>	The string containing the substring to replace.
<i>from</i>	The substring to replace.
<i>to</i>	The string with which to replace the substring.

**Returns**

`true` if a replacement was made, `false` otherwise.

**8.5.2.5 std::vector< std::string > pgutils::split ( const std::string & s, const char delim )**

Splits a delimited string into tokens.

**Parameters**

<i>s</i>	The string to split.
<i>delim</i>	The delimiter character on which to split.

**Returns**

A vector of tokens.



**8.5.2.6** `std::vector< std::string > & pgutils::split ( std::vector< std::string > & vec, const std::string & s, const char delim )`

Splits a delimited string into tokens.

#### Parameters

<i>vec</i>	The vector into which to add the tokens.
<i>s</i>	The string to split.
<i>delim</i>	The delimiter character on which to split.

#### Returns

A reference to `vec`.

**8.5.2.7** `std::vector< std::vector< std::string > > & pgutils::split_lines ( std::vector< std::vector< std::string > > & vec, std::istream & ifs, const char delim )`

Populates a vector of vectors of fields from a stream.

#### Parameters

<i>vec</i>	The vector to populate.
<i>ifs</i>	The input stream.
<i>delim</i>	The delimiter character to split each content line.

#### Returns

A reference to `vec`.

**8.5.2.8** `std::string & pgutils::trim ( std::string & s )`

Trims leading and trailing whitespace from a string.

#### Parameters

<i>s</i>	The string to trim.
----------	---------------------

#### Returns

The trimmed string.

**8.5.2.9** `std::string & pgutils::trim_back ( std::string & s )`

Trims trailing whitespace from a string.

#### Parameters

<i>s</i>	The string to trim.
----------	---------------------

#### Returns

The trimmed string.

#### 8.5.2.10 `std::string & pgutils::trim_front ( std::string & s )`

Trims leading whitespace from a string.

##### Parameters

<code>s</code>	The string to trim.
----------------	---------------------

##### Returns

The trimmed string.

## 8.6 Database program.

### Functions

- static void `set_configuration` (`Config` &config, int argc, char \*argv[])  
*Sets program configuration options.*
- static bool `check_help_and_version` (const `Config` &config)  
*Prints help or version messages if requested.*
- static bool `check_db_parameters` (const `Config` &config)  
*Checks if database, hostname and username were provided.*
- static void `print_usage_message` ()  
*Prints a program usage message.*
- static void `print_version_message` ()  
*Prints a program version message.*
- static void `print_help_message` ()  
*Prints a program help message.*
- static std::string `login` (void)  
*Gets a password from the terminal.*
- int `main` (int argc, char \*argv[])  
*Main function.*

### Variables

- static const char \* `programe` = "gl\_db"  
*Static variable for program name.*

### 8.6.1 Detailed Description

Administrative database management program.

### 8.6.2 Function Documentation

#### 8.6.2.1 static bool `check_db_parameters` ( const `Config` & `config` ) [static]

Checks if database, hostname and username were provided.

#### Parameters

<code>config</code>	Reference to a <code>Config</code> object.
---------------------	--

#### Returns

`true` if the information was provided, `false` otherwise.

#### 8.6.2.2 static bool `check_help_and_version` ( const `Config` & `config` ) [static]

Prints help or version messages if requested.

#### Parameters

<code>config</code>	Reference to a <code>Config</code> object.
---------------------	--

**Returns**

`true` if the help or version message was requested, `false` otherwise.

**8.6.2.3 static std::string login ( void ) [static]**

Gets a password from the terminal.

**Returns**

The password.

**8.6.2.4 int main ( int argc, char \* argv[] )**

Main function.

**Parameters**

<i>argc</i>	Number of command line arguments.
<i>argv</i>	Command line arguments.

**Returns**

Exit status code.

**8.6.2.5 static void set\_configuration ( Config & config, int argc, char \* argv[] ) [static]**

Sets program configuration options.

**Parameters**

<i>config</i>	Reference to a Config object.
<i>argc</i>	<code>argc</code> passed to <code>main()</code> .
<i>argv</i>	<code>argv</code> passed to <code>main()</code> .

## 8.7 Reporting program.

### Functions

- static void `set_configuration` (`Config` &config, int argc, char \*argv[])  
*Sets program configuration options.*
- static bool `check_help_and_version` (const `Config` &config)  
*Prints help or version messages if requested.*
- static bool `check_db_parameters` (const `Config` &config)  
*Checks if database, hostname and username were provided.*
- static void `print_usage_message` ()  
*Prints a program usage message.*
- static void `print_version_message` ()  
*Prints a program version message.*
- static void `print_help_message` ()  
*Prints a program help message.*
- static std::string `login` (void)  
*Gets a password from the terminal.*
- int `main` (int argc, char \*argv[])  
*Main function.*

### Variables

- static const char \* `programe` = "gl\_report"  
*Static variable for program name.*

#### 8.7.1 Detailed Description

Administrative reporting program.

#### 8.7.2 Function Documentation

**8.7.2.1** static bool `check_db_parameters` ( const `Config` & *config* ) [static]

Checks if database, hostname and username were provided.

##### Parameters

<i>config</i>	Reference to a Config object.
---------------	-------------------------------

##### Returns

`true` if the information was provided, `false` otherwise.

**8.7.2.2** static bool `check_help_and_version` ( const `Config` & *config* ) [static]

Prints help or version messages if requested.

##### Parameters

<i>config</i>	Reference to a Config object.
---------------	-------------------------------

**Returns**

`true` if the help or version message was requested, `false` otherwise.

**8.7.2.3 static std::string login ( void ) [static]**

Gets a password from the terminal.

**Returns**

The password.

**8.7.2.4 int main ( int argc, char \* argv[] )**

Main function.

**Parameters**

<i>argc</i>	Number of command line arguments.
<i>argv</i>	Command line arguments.

**Returns**

Exit status code.

**8.7.2.5 static void set\_configuration ( Config & config, int argc, char \* argv[] ) [static]**

Sets program configuration options.

**Parameters**

<i>config</i>	Reference to a Config object.
<i>argc</i>	<code>argc</code> passed to <code>main()</code> .
<i>argv</i>	<code>argv</code> passed to <code>main()</code> .

## 8.8 User administration program.

### Functions

- static void `set_configuration` (`Config` &config, int argc, char \*argv[])  
*Sets program configuration options.*
- static bool `check_help_and_version` (const `Config` &config)  
*Prints help or version messages if requested.*
- static bool `check_db_parameters` (const `Config` &config)  
*Checks if database, hostname and username were provided.*
- `GLUser` `get_user` (`Config` &config, `GLDatabase` &gdb)  
*Returns a user from either an ID or a name.*
- static void `show_user_details` (const `GLUser` &user)  
*Outputs details for a user.*
- static void `enable_user` (`GLUser` &user, `Config` &config, `GLDatabase` &gdb)  
*Enables or disables a user.*
- static void `set_user_password` (`GLUser` &user, `Config` &config, `GLDatabase` &gdb)  
*Sets a user's password.*
- static void `check_user_password` (`GLUser` &user, `Config` &config)  
*Checks a user's password.*
- static void `print_usage_message` ()  
*Prints a program usage message.*
- static void `print_version_message` ()  
*Prints a program version message.*
- static void `print_help_message` ()  
*Prints a program help message.*
- static std::string `login` (void)  
*Gets a password from the terminal.*
- int `main` (int argc, char \*argv[])  
*Main function.*

### Variables

- static const char \* `progrname` = "gl\_user"  
*Static variable for program name.*

#### 8.8.1 Detailed Description

User administration program.

#### 8.8.2 Function Documentation

##### 8.8.2.1 static bool `check_db_parameters` ( const `Config` & *config* ) [static]

Checks if database, hostname and username were provided.

##### Parameters

<i>config</i>	Reference to a Config object.
---------------	-------------------------------

**Returns**

`true` if the information was provided, `false` otherwise.

**8.8.2.2 static bool check\_help\_and\_version ( const Config & config ) [static]**

Prints help or version messages if requested.

**Parameters**

<i>config</i>	Reference to a Config object.
---------------	-------------------------------

**Returns**

`true` if the help or version message was requested, `false` otherwise.

**8.8.2.3 static void check\_user\_password ( GLUser & user, Config & config ) [static]**

Checks a user's password.

**Parameters**

<i>user</i>	Reference to user.
<i>config</i>	Reference to program configuration options.

**8.8.2.4 static void enable\_user ( GLUser & user, Config & config, GLDatabase & gdb ) [static]**

Enables or disables a user.

**Parameters**

<i>user</i>	Reference to user.
<i>config</i>	Reference to program configuration.
<i>gdb</i>	Reference to database object.

**8.8.2.5 GLUser get\_user ( Config & config, GLDatabase & gdb )**

Returns a user from either an ID or a name.

**Parameters**

<i>config</i>	Program configurations object.
<i>gdb</i>	Database object.

**Returns**

The user.

**8.8.2.6 static std::string login ( void ) [static]**

Gets a password from the terminal.



## Returns

The password.

8.8.2.7 `int main ( int argc, char * argv[] )`

Main function.

## Parameters

<i>argc</i>	Number of command line arguments.
<i>argv</i>	Command line arguments.

## Returns

Exit status code.

8.8.2.8 `static void set_configuration ( Config & config, int argc, char * argv[] ) [static]`

Sets program configuration options.

## Parameters

<i>config</i>	Reference to a Config object.
<i>argc</i>	argc passed to <code>main()</code> .
<i>argv</i>	argv passed to <code>main()</code> .

8.8.2.9 `static void set_user_password ( GLUser & user, Config & config, GLDatabase & gdb ) [static]`

Sets a user's password.

## Parameters

<i>user</i>	Reference to user.
<i>config</i>	Reference to program configuration.
<i>gdb</i>	Reference to database object.

8.8.2.10 `static void show_user_details ( const GLUser & user ) [static]`

Outputs details for a user.

## Parameters

<i>user</i>	Reference to user.
-------------	--------------------



## Chapter 9

# Class Documentation

### 9.1 genleg::Config Class Reference

Configuration options class.

```
#include <config.h>
```

#### Public Member Functions

- [Config](#) ()
- [~Config](#) ()
- void [add\\_cmdline\\_option](#) (const std::string option, const enum [Argument](#) arg)  
*Adds a supported command line option.*
- void [populate\\_from\\_cmdline](#) (const int argc, char \*const \*argv)  
*Populates options from the command line.*
- void [populate\\_from\\_file](#) (const std::string filename)  
*Populates options from a configuration file.*
- bool [is\\_set](#) (const std::string option) const  
*Checks if an option is set.*
- const std::string & [operator\[\]](#) (const std::string &option) const  
*operator[] overload.*

#### Private Attributes

- std::map< std::string,  
std::string > [m\\_opts\\_set](#)
- std::list< std::pair  
< std::string, enum [Argument](#) > > [m\\_opts\\_supp](#)

#### 9.1.1 Detailed Description

Configuration options class.

#### 9.1.2 Constructor & Destructor Documentation

##### 9.1.2.1 Config::Config ( )

Constructor

### 9.1.2.2 Config::~Config ( )

Destructor

## 9.1.3 Member Function Documentation

### 9.1.3.1 void Config::add\_cmdline\_option ( const std::string *option*, const enum Argument *arg* )

Adds a supported command line option.

#### Parameters

<i>option</i>	The name of the option.
<i>arg</i>	The argument specification for the option.

### 9.1.3.2 bool Config::is\_set ( const std::string *option* ) const

Checks is an option is set.

#### Parameters

<i>option</i>	The name of the option to check.
---------------	----------------------------------

#### Returns

`true` if the option has been set, `false` if it has not.

### 9.1.3.3 const std::string & Config::operator[] ( const std::string & *option* ) const

operator[] overload.

Retrieves the value of a set option.

#### Parameters

<i>option</i>	The name of the option.
---------------	-------------------------

#### Returns

The value of the option.

#### Exceptions

<a href="#"><i>ConfigOptionNotSet</i></a>	If the named option has not been set.
---	---------------------------------------

### 9.1.3.4 void Config::populate\_from\_cmdline ( const int *argc*, char \*const \* *argv* )

Populates options from the command line.

#### Parameters

<i>argc</i>	<i>argc</i> supplied to <code>main()</code> .
<i>argv</i>	<i>argv</i> supplied to <code>main()</code> .

## Exceptions

<a href="#"><i>ConfigBadOption</i></a>	If an unsupported option is specified, or if a required argument is missing, or if an unexpected argument is found.
--	---

## 9.1.3.5 void Config::populate\_from\_file ( const std::string filename )

Populates options from a configuration file.

## Parameters

<i>filename</i>	The name of the configuration file.
-----------------	-------------------------------------

## Exceptions

<a href="#"><i>ConfigCouldNotOpenFile</i></a>	If the configuration file cannot be opened.
<a href="#"><i>ConfigBadConfigFile</i></a>	If the configuration file is badly formed.

## 9.1.4 Member Data Documentation

## 9.1.4.1 std::map&lt;std::string, std::string&gt; genleg::Config::m\_opts\_set [private]

Map of options which have been set

## 9.1.4.2 std::list&lt;std::pair&lt;std::string, enum Argument&gt; &gt; genleg::Config::m\_opts\_supp [private]

List of options which are supported

The documentation for this class was generated from the following files:

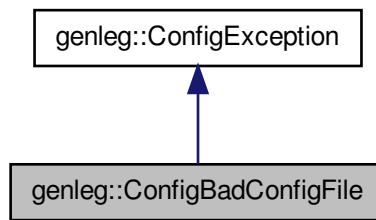
- lib/config/[config.h](#)
- lib/config/[config.cpp](#)
- lib/config/[config\\_getopt.cpp](#)

## 9.2 genleg::ConfigBadConfigFile Class Reference

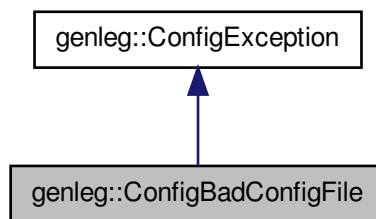
Exception class for badly formed configuration file.

```
#include <config.h>
```

Inheritance diagram for `genleg::ConfigBadConfigFile`:



Collaboration diagram for `genleg::ConfigBadConfigFile`:



## Public Member Functions

- [ConfigBadConfigFile](#) (const std::string &msg)  
*Constructor.*

### 9.2.1 Detailed Description

Exception class for badly formed configuration file.

### 9.2.2 Constructor & Destructor Documentation

9.2.2.1 `genleg::ConfigBadConfigFile::ConfigBadConfigFile ( const std::string & msg ) [inline], [explicit]`

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

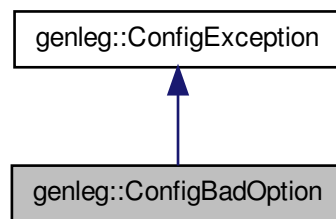
- [lib/config/config.h](#)

## 9.3 genleg::ConfigBadOption Class Reference

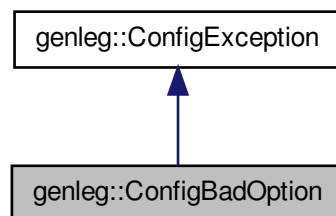
Exception class for bad provided option.

```
#include <config.h>
```

Inheritance diagram for genleg::ConfigBadOption:



Collaboration diagram for genleg::ConfigBadOption:



### Public Member Functions

- [ConfigBadOption](#) (const std::string &msg)  
*Constructor.*

#### 9.3.1 Detailed Description

Exception class for bad provided option.

#### 9.3.2 Constructor & Destructor Documentation

### 9.3.2.1 `genleg::ConfigBadOption::ConfigBadOption ( const std::string & msg ) [inline], [explicit]`

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

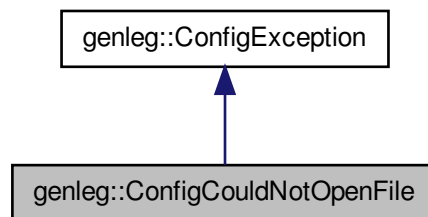
- `lib/config/config.h`

## 9.4 `genleg::ConfigCouldNotOpenFile` Class Reference

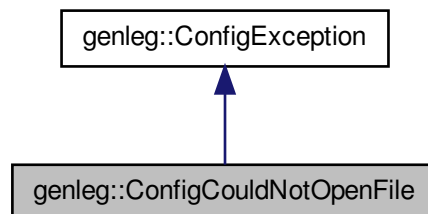
Exception class for when conf file cannot be opened.

```
#include <config.h>
```

Inheritance diagram for `genleg::ConfigCouldNotOpenFile`:



Collaboration diagram for `genleg::ConfigCouldNotOpenFile`:



### Public Member Functions

- `ConfigCouldNotOpenFile` (const std::string &msg)  
*Constructor.*



### 9.4.1 Detailed Description

Exception class for when conf file cannot be opened.

### 9.4.2 Constructor & Destructor Documentation

9.4.2.1 `genleg::ConfigCouldNotOpenFile::ConfigCouldNotOpenFile ( const std::string & msg ) [inline], [explicit]`

Constructor.

Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

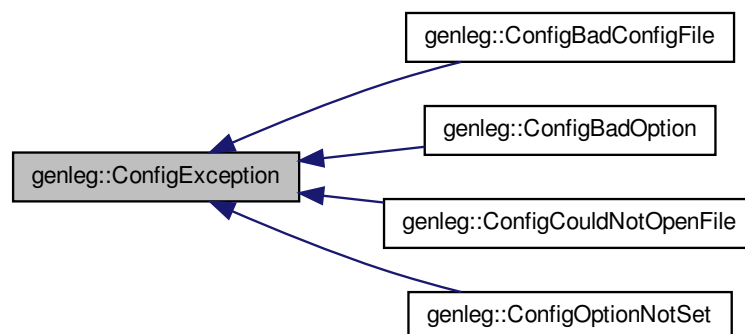
- [lib/config/config.h](#)

## 9.5 genleg::ConfigException Class Reference

Configuration module exception base class.

```
#include <config.h>
```

Inheritance diagram for genleg::ConfigException:



### Public Member Functions

- [ConfigException](#) (const std::string &msg)  
*Constructor.*

### 9.5.1 Detailed Description

Configuration module exception base class.

## 9.5.2 Constructor & Destructor Documentation

### 9.5.2.1 `genleg::ConfigException::ConfigException ( const std::string & msg ) [inline],[explicit]`

Constructor.

Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

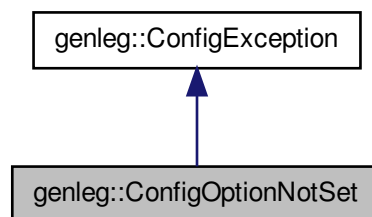
- `lib/config/config.h`

## 9.6 `genleg::ConfigOptionNotSet` Class Reference

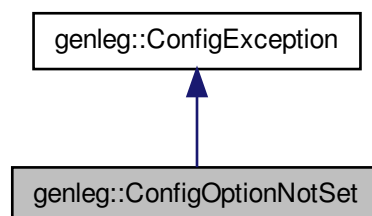
Exception class for option not set.

```
#include <config.h>
```

Inheritance diagram for `genleg::ConfigOptionNotSet`:



Collaboration diagram for `genleg::ConfigOptionNotSet`:



## Public Member Functions

- [`ConfigOptionNotSet`](#) (const std::string &msg)

*Constructor.*

### 9.6.1 Detailed Description

Exception class for option not set.

### 9.6.2 Constructor & Destructor Documentation

#### 9.6.2.1 genleg::ConfigOptionNotSet::ConfigOptionNotSet ( const std::string & msg ) [inline],[explicit]

Constructor.

Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

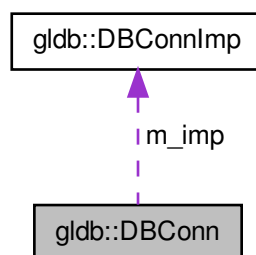
- lib/config/[config.h](#)

## 9.7 glldb::DBConn Class Reference

Database connection class.

```
#include <dbconn.h>
```

Collaboration diagram for glldb::DBConn:



### Public Member Functions

- [DBConn](#) ([DBConnImp](#) \*imp)  
*Constructor.*
- [~DBConn](#) ()  
*Destructor..*
- void [query](#) (const std::string &sql\_query)  
*Runs an SQL query.*
- [Table select](#) (const std::string &query)

*Runs an SQL SELECT query.*

- [DBConn](#) (const [DBConn](#) &)
- [DBConn](#) (const [DBConn](#) &&)
- [DBConn](#) & [operator=](#) (const [DBConn](#) &)
- [DBConn](#) & [operator=](#) (const [DBConn](#) &&)

## Private Attributes

- [DBConnImp](#) \* [m\\_imp](#)

## 9.7.1 Detailed Description

Database connection class.

## 9.7.2 Constructor & Destructor Documentation

### 9.7.2.1 [DBConn::DBConn \( \[DBConnImp\]\(#\) \\* \*imp\* \)](#) `[explicit]`

Constructor.

#### Parameters

<i>imp</i>	Pointer to database implementation object.
------------	--

### 9.7.2.2 [gldb::DBConn::DBConn \( const \[DBConn\]\(#\) & \)](#)

Deleted copy constructor

### 9.7.2.3 [gldb::DBConn::DBConn \( const \[DBConn\]\(#\) && \)](#)

Deleted move constructor

## 9.7.3 Member Function Documentation

### 9.7.3.1 [DBConn& gldb::DBConn::operator= \( const \[DBConn\]\(#\) & \)](#)

Deleted copy assignment operator

### 9.7.3.2 [DBConn& gldb::DBConn::operator= \( const \[DBConn\]\(#\) && \)](#)

Deleted move assignment operator

### 9.7.3.3 [void DBConn::query \( const std::string & \*sql\\_query\* \)](#)

Runs an SQL query.

#### Parameters

<i>sql_query</i>	The query.
------------------	------------

## Returns

A [Table](#) object containing the results.

9.7.3.4 Table DBConn::select ( const std::string & *query* )

Runs an SQL SELECT query.

## Parameters

<i>query</i>	The query.
--------------	------------

## Returns

A [Table](#) object containing the results.

## 9.7.4 Member Data Documentation

## 9.7.4.1 DBConnImp\* glldb::DBConn::m\_imp [private]

Pointer to database implementation object.

The documentation for this class was generated from the following files:

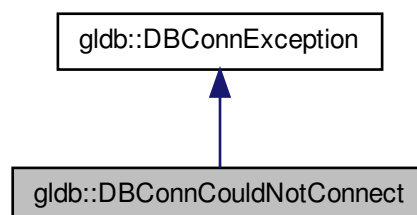
- lib/database/[dbconn.h](#)
- lib/database/[dbconn.cpp](#)

## 9.8 glldb::DBConnCouldNotConnect Class Reference

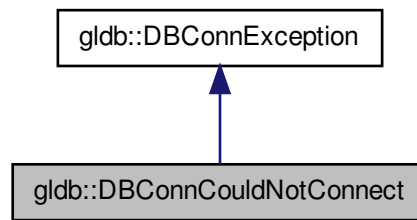
Could not connect to database exception class.

```
#include <dbconn.h>
```

Inheritance diagram for glldb::DBConnCouldNotConnect:



Collaboration diagram for `gldb::DBConnCouldNotConnect`:



## Public Member Functions

- [DBConnCouldNotConnect](#) (const std::string &msg)  
*Constructor.*

### 9.8.1 Detailed Description

Could not connect to database exception class.

### 9.8.2 Constructor & Destructor Documentation

9.8.2.1 `gldb::DBConnCouldNotConnect::DBConnCouldNotConnect ( const std::string & msg ) [inline], [explicit]`

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

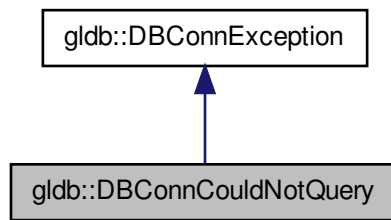
- lib/database/[dbconn.h](#)

## 9.9 gldb::DBConnCouldNotQuery Class Reference

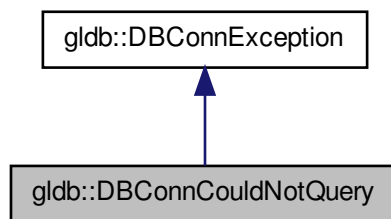
Could not execute database query exception class.

```
#include <dbconn.h>
```

Inheritance diagram for glldb::DBConnCouldNotQuery:



Collaboration diagram for glldb::DBConnCouldNotQuery:



## Public Member Functions

- [`DBConnCouldNotQuery`](#) (const std::string &msg)  
*Constructor.*

### 9.9.1 Detailed Description

Could not execute database query exception class.

### 9.9.2 Constructor & Destructor Documentation

9.9.2.1 `glldb::DBConnCouldNotQuery::DBConnCouldNotQuery ( const std::string & msg ) [inline], [explicit]`

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

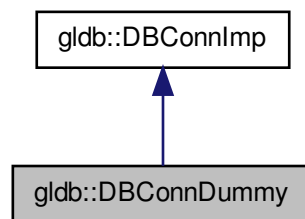
- [lib/database/dbconn.h](#)

## 9.10 gldb::DBConnDummy Class Reference

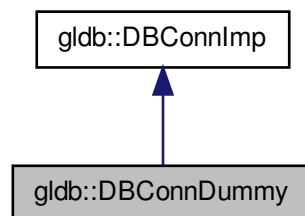
Dummy database implementation class.

```
#include <dbconn_dummy_imp.h>
```

Inheritance diagram for gldb::DBConnDummy:



Collaboration diagram for gldb::DBConnDummy:



### Public Member Functions

- [DBConnDummy](#) (const std::string database, const std::string hostname, const std::string username, const std::string password)  
*Constructor.*
- [DBConnDummy](#) (const [DBConnDummy](#) &)
- virtual [~DBConnDummy](#) ()
- [DBConnDummy](#) & [operator=](#) (const [DBConnDummy](#) &)
- virtual void [query](#) (const std::string &sql\_query)  
*Runs an SQL query.*
- [Table select](#) (const std::string &[query](#))  
*Fakes running of an SQL SELECT query.*



### 9.10.1 Detailed Description

Dummy database implementation class.

### 9.10.2 Constructor & Destructor Documentation

#### 9.10.2.1 DBConnDummy::DBConnDummy ( const std::string *database*, const std::string *hostname*, const std::string *username*, const std::string *password* )

Constructor.

##### Parameters

<i>database</i>	The name of the Dummy database.
<i>hostname</i>	The hostname of the server.
<i>username</i>	The username to log into the database.
<i>password</i>	The password to log into the database.

#### 9.10.2.2 glldb::DBConnDummy::DBConnDummy ( const DBConnDummy & )

Deleted copy constructor

#### 9.10.2.3 DBConnDummy::~DBConnDummy ( ) [virtual]

Destructor

### 9.10.3 Member Function Documentation

#### 9.10.3.1 DBConnDummy& glldb::DBConnDummy::operator= ( const DBConnDummy & )

Deleted assignment operator

#### 9.10.3.2 void DBConnDummy::query ( const std::string & *sql\_query* ) [virtual]

Runs an SQL query.

##### Parameters

<i>sql_query</i>	The query.
------------------	------------

##### Exceptions

<a href="#"><i>DBConnCouldNotQuery</i></a>	If could not successfully execute query.
--	--

Implements [`glldb::DBConnImp`](#).

#### 9.10.3.3 Table DBConnDummy::select ( const std::string & *query* ) [virtual]

Fakes running of an SQL SELECT query.

## Parameters

<i>query</i>	Any query.
--------------	------------

## Returns

A [Table](#) object containing dummy results.

Implements [gldb::DBConnImp](#).

The documentation for this class was generated from the following files:

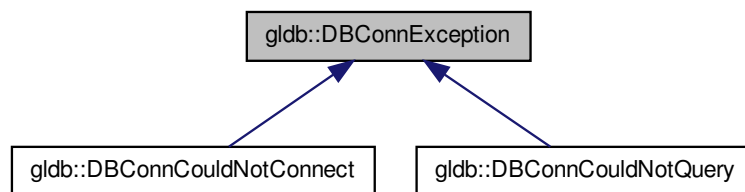
- [lib/database\\_imp/dummy/dbconn\\_dummy\\_imp.h](#)
- [lib/database\\_imp/dummy/dbconn\\_dummy\\_imp.cpp](#)

## 9.11 gldb::DBConnException Class Reference

Base database connection exception class.

```
#include <dbconn.h>
```

Inheritance diagram for gldb::DBConnException:



### Public Member Functions

- [DBConnException](#) (const std::string &msg)  
*Constructor.*

#### 9.11.1 Detailed Description

Base database connection exception class.

#### 9.11.2 Constructor & Destructor Documentation

9.11.2.1 `gldb::DBConnException::DBConnException ( const std::string & msg ) [inline], [explicit]`

Constructor.

## Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

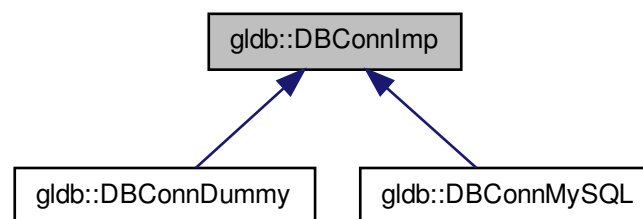
- lib/database/[dbconn.h](#)

## 9.12 glldb::DBConnImp Class Reference

Abstract database implementation base class.

```
#include <dbconnimp.h>
```

Inheritance diagram for glldb::DBConnImp:



### Public Member Functions

- [DBConnImp](#) ()
- virtual [~DBConnImp](#) ()
- virtual void [query](#) (const std::string &sql\_query)=0  
*Runs an SQL query.*
- virtual [Table select](#) (const std::string &query)=0  
*Runs an SQL SELECT query.*

### 9.12.1 Detailed Description

Abstract database implementation base class.

### 9.12.2 Constructor & Destructor Documentation

#### 9.12.2.1 glldb::DBConnImp::DBConnImp ( ) [inline]

Constructor

#### 9.12.2.2 virtual glldb::DBConnImp::~~DBConnImp ( ) [inline], [virtual]

Destructor

### 9.12.3 Member Function Documentation

9.12.3.1 `virtual void gldb::DBConnImp::query ( const std::string & sql_query )` [pure virtual]

Runs an SQL query.

#### Parameters

<i>sql_query</i>	The query.
------------------	------------

Implemented in [gldb::DBConnMySQL](#), and [gldb::DBConnDummy](#).

9.12.3.2 `virtual Table gldb::DBConnImp::select ( const std::string & query )` [pure virtual]

Runs an SQL SELECT query.

#### Parameters

<i>query</i>	The query.
--------------	------------

#### Returns

A [Table](#) object containing the results.

Implemented in [gldb::DBConnMySQL](#), and [gldb::DBConnDummy](#).

The documentation for this class was generated from the following file:

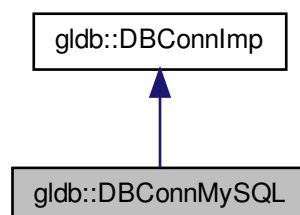
- `lib/database/dbconnimp.h`

## 9.13 gldb::DBConnMySQL Class Reference

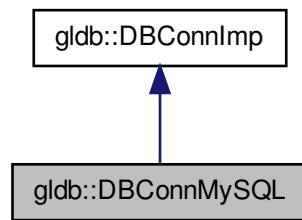
MySQL database implementation class.

```
#include <dbconn_mysql_imp.h>
```

Inheritance diagram for `gldb::DBConnMySQL`:



Collaboration diagram for glldb::DBConnMySQL:



## Public Member Functions

- [DBConnMySQL](#) (const std::string &database, const std::string &hostname, const std::string &username, const std::string &password)  
*Constructor.*
- [DBConnMySQL](#) (const [DBConnMySQL](#) &)
- [DBConnMySQL](#) (const [DBConnMySQL](#) &&)
- virtual [~DBConnMySQL](#) ()
- [DBConnMySQL](#) & [operator=](#) (const [DBConnMySQL](#) &)
- [DBConnMySQL](#) & [operator=](#) (const [DBConnMySQL](#) &&)
- virtual void [query](#) (const std::string &sql\_query)  
*Runs an SQL query.*
- virtual [Table select](#) (const std::string &sql\_query)  
*Runs an SQL SELECT query.*

## Private Attributes

- MYSQL \* [m\\_conn](#)

### 9.13.1 Detailed Description

MySQL database implementation class.

### 9.13.2 Constructor & Destructor Documentation

#### 9.13.2.1 [DBConnMySQL::DBConnMySQL](#) ( const std::string & *database*, const std::string & *hostname*, const std::string & *username*, const std::string & *password* )

Constructor.

#### Parameters

<i>database</i>	The name of the MySQL database.
<i>hostname</i>	The hostname of the server.
<i>username</i>	The username to log into the database.
<i>password</i>	The password to log into the database.

## Exceptions

<a href="#"><i>DBConnCouldNotConnect</i></a>	If could not connect to database.
--	-----------------------------------

9.13.2.2 `gldb::DBConnMySQL::DBConnMySQL ( const DBConnMySQL & )`

Deleted copy constructor

9.13.2.3 `gldb::DBConnMySQL::DBConnMySQL ( const DBConnMySQL && )`

Delete move constructor

9.13.2.4 `virtual gldb::DBConnMySQL::~DBConnMySQL ( ) [virtual]`

Destructor

### 9.13.3 Member Function Documentation

9.13.3.1 `DBConnMySQL& gldb::DBConnMySQL::operator= ( const DBConnMySQL & )`

Deleted assignment operator

9.13.3.2 `DBConnMySQL& gldb::DBConnMySQL::operator= ( const DBConnMySQL && )`

Deleted move assignment operator

9.13.3.3 `virtual void gldb::DBConnMySQL::query ( const std::string & sql_query ) [virtual]`

Runs an SQL query.

## Parameters

<i>sql_query</i>	The SQL query.
------------------	----------------

## Exceptions

<a href="#"><i>DBConnCouldNotQuery</i></a>	If could not successfully execute query.
--	--

Implements [`gldb::DBConnImp`](#).

9.13.3.4 `virtual Table gldb::DBConnMySQL::select ( const std::string & sql_query ) [virtual]`

Runs an SQL SELECT query.

## Parameters

<i>sql_query</i>	The SQL query.
------------------	----------------

## Returns

A [`Table`](#) object containing the results.

## Exceptions

<a href="#">DBConnCouldNotQuery</a>	If could not successfully execute query.
-------------------------------------	--

Implements [gldb::DBConnImp](#).

### 9.13.4 Member Data Documentation

#### 9.13.4.1 MySQL\* gldb::DBConnMySQL::m\_conn [private]

The initialized MySQL handle.

The documentation for this class was generated from the following files:

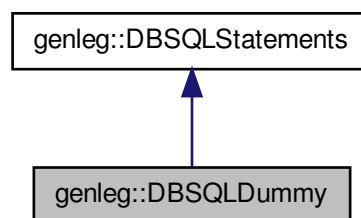
- lib/database\_imp/mysql/[dbconn\\_mysql\\_imp.h](#)
- lib/database\_imp/mysql/[dbconn\\_mysql\\_imp.cpp](#)

## 9.14 genleg::DBSQLDummy Class Reference

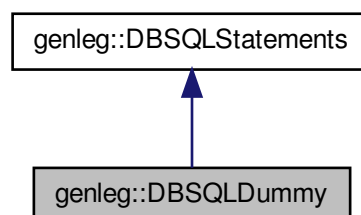
Dummy SQL statements class.

```
#include <dbsql_dummy.h>
```

Inheritance diagram for genleg::DBSQLDummy:



Collaboration diagram for genleg::DBSQLDummy:



## Additional Inherited Members

### 9.14.1 Detailed Description

Dummy SQL statements class.

The documentation for this class was generated from the following file:

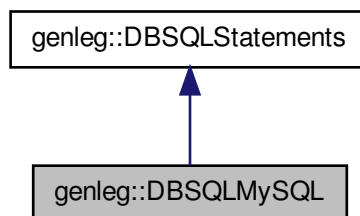
- lib/dbsql/[dbsql\\_dummy.h](#)

## 9.15 genleg::DBSQLMySQL Class Reference

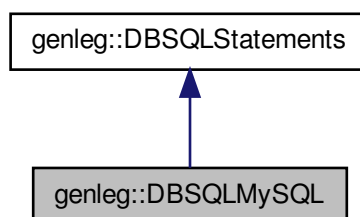
MySQL SQL statements class.

```
#include <dbsql_mysql.h>
```

Inheritance diagram for genleg::DBSQLMySQL:



Collaboration diagram for genleg::DBSQLMySQL:



## Additional Inherited Members

### 9.15.1 Detailed Description

MySQL SQL statements class.



The documentation for this class was generated from the following file:

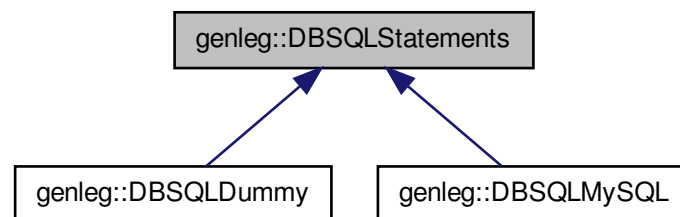
- [lib/dbsql/dbsql\\_mysql.h](#)

## 9.16 genleg::DBSQLStatements Class Reference

SQL statements class.

```
#include <dbsqlstatements.h>
```

Inheritance diagram for genleg::DBSQLStatements:



### Public Member Functions

- [DBSQLStatements](#) ()
- virtual [~DBSQLStatements](#) ()
- virtual std::string [create\\_table](#) (const std::string &table\_name) const  
*Returns a SQL statement for creating a table.*
- virtual std::string [drop\\_table](#) (const std::string &table\_name) const  
*Returns a SQL statement for dropping a table.*
- virtual std::string [create\\_view](#) (const std::string &view\_name) const  
*Returns a SQL statement for creating a view.*
- virtual std::string [drop\\_view](#) (const std::string &view\_name) const  
*Returns a SQL statement for dropping a view.*
- virtual std::string [user\\_by\\_id](#) (const std::string &user\_id) const  
*Returns a SQL statement to select a user by ID.*
- virtual std::string [user\\_by\\_username](#) (const std::string &user\_name) const  
*Returns a SQL statement to select a user by username.*
- virtual std::string [update\\_user](#) (const [GLUser](#) &user) const  
*Returns a SQL UPDATE statement to update a user.*
- virtual std::string [grant](#) (const std::string &user\_id, const std::string &perm) const  
*Returns a SQL statement to grant a user a permission.*
- virtual std::string [revoke](#) (const std::string &user\_id, const std::string &perm) const  
*Returns a SQL UPDATE statement to revoke a permission from a user.*
- virtual std::string [get\\_perms](#) (const std::string &user\_id) const  
*Returns a SQL UPDATE statement to list a user's permissions.*
- virtual std::string [currenttb](#) () const  
*Returns a SQL statement to run the current trial balance report.*
- virtual std::string [currenttb\\_by\\_entity](#) (const std::string &entity) const  
*Returns a SQL statement to run the current trial balance report by entity.*

### 9.16.1 Detailed Description

SQL statements class.

### 9.16.2 Constructor & Destructor Documentation

#### 9.16.2.1 DBSQLStatements::DBSQLStatements ( )

Constructor

#### 9.16.2.2 DBSQLStatements::~~DBSQLStatements ( ) [virtual]

Destructor

### 9.16.3 Member Function Documentation

#### 9.16.3.1 std::string DBSQLStatements::create\_table ( const std::string & *table\_name* ) const [virtual]

Returns a SQL statement for creating a table.

Parameters

<i>table_name</i>	The table to create.
-------------------	----------------------

Returns

The SQL statement to create the table.

#### 9.16.3.2 std::string DBSQLStatements::create\_view ( const std::string & *view\_name* ) const [virtual]

Returns a SQL statement for creating a view.

Parameters

<i>view_name</i>	The view to create.
------------------	---------------------

Returns

The SQL statement to create the view.

#### 9.16.3.3 std::string DBSQLStatements::currenttb ( ) const [virtual]

Returns a SQL statement to run the current trial balance report.

Returns

The SQL statement.

#### 9.16.3.4 std::string DBSQLStatements::currenttb\_by\_entity ( const std::string & *entity* ) const [virtual]

Returns a SQL statement to run the current trial balance report by entity.

## Parameters

<i>entity</i>	The entity number for which to run the report.
---------------	--

## Returns

The SQL statement.

**9.16.3.5** `std::string DBSQLStatements::drop_table ( const std::string & table_name ) const` [virtual]

Returns a SQL statement for dropping a table.

## Parameters

<i>table_name</i>	The table to drop.
-------------------	--------------------

## Returns

The SQL statement to drop the table.

**9.16.3.6** `std::string DBSQLStatements::drop_view ( const std::string & view_name ) const` [virtual]

Returns a SQL statement for dropping a view.

## Parameters

<i>view_name</i>	The view to drop.
------------------	-------------------

## Returns

The SQL statement to drop the view.

**9.16.3.7** `std::string DBSQLStatements::get_perms ( const std::string & user_id ) const` [virtual]

Returns a SQL UPDATE statement to list a user's permissions.

## Parameters

<i>user_id</i>	The user ID for which to list.
----------------	--------------------------------

## Returns

The SQL statement.

**9.16.3.8** `std::string DBSQLStatements::grant ( const std::string & user_id, const std::string & perm ) const` [virtual]

Returns a SQL statement to grant a user a permission.

## Attention

This function always sets the user granting the permission to user 1. This will need to be updated to support the recording of which user has granted the permission, when support for others to be able to do so is implemented.

**Parameters**

<i>user_id</i>	The user ID for which to grant the permission.
<i>perm</i>	A string containing the name of the permission.

**Returns**

The SQL statement.

**9.16.3.9** `std::string DBSQLStatements::revoke ( const std::string & user_id, const std::string & perm ) const` [virtual]

Returns a SQL UPDATE statement to revoke a permission from a user.

**Parameters**

<i>user_id</i>	The user ID from which to revoke.
<i>perm</i>	The permission to revoke.

**Returns**

The SQL statement.

**9.16.3.10** `std::string DBSQLStatements::update_user ( const GLUser & user ) const` [virtual]

Returns a SQL UPDATE statement to update a user.

**Parameters**

<i>user</i>	A user object.
-------------	----------------

**Returns**

The SQL statement.

**9.16.3.11** `std::string DBSQLStatements::user_by_id ( const std::string & user_id ) const` [virtual]

Returns a SQL statement to select a user by ID.

**Parameters**

<i>user_id</i>	The user_id
----------------	-------------

**Returns**

The SQL statement.

**9.16.3.12** `std::string DBSQLStatements::user_by_username ( const std::string & user_name ) const` [virtual]

Returns a SQL statement to select a user by username.

**Parameters**

<i>user_name</i>	The username.
------------------	---------------

## Returns

The SQL statement.

The documentation for this class was generated from the following files:

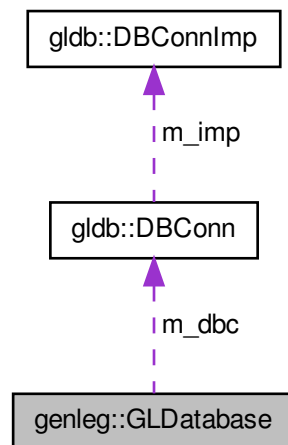
- lib/dbsql/dbsqlstatements.h
- lib/dbsql/dbsqlstatements.cpp

## 9.17 genleg::GLDatabase Class Reference

General ledger database class.

```
#include <gldatabase.h>
```

Collaboration diagram for genleg::GLDatabase:



### Public Member Functions

- `GLDatabase` (const std::string &database, const std::string &hostname, const std::string &username, const std::string &password)  
*Constructor.*
- `~GLDatabase` ()
- void `create_structure` ()  
*Creates the database structure.*
- void `destroy_structure` ()  
*Destroys the database structure.*
- void `load_sample_data` (const std::string &dir)  
*Loads sample data into the database.*
- `GLUser` `get_user_by_id` (const std::string &user\_id)  
*Returns a user from an ID.*
- `GLUser` `get_user_by_username` (const std::string &user\_name)  
*Returns a user from a user name.*

- void [update\\_user](#) (const [GLUser](#) &user)  
*Updates a user's details.*
- void [grant](#) (const [GLUser](#) &user, const std::string &perm)  
*Grants a user a permission.*
- void [revoke](#) (const [GLUser](#) &user, const std::string &perm)  
*Revokes a permission from a user.*
- [GLReport report](#) (const std::string &report\_name, const std::string &arg="")  
*Runs a report.*

### Static Public Member Functions

- static std::string [backend](#) ()  
*Returns the backend database implementation.*

### Private Member Functions

- [GLUser create\\_user](#) ([gldb::Table](#) &table)  
*Creates a user from a query table.*
- [GLReport current\\_trial\\_balance\\_report](#) (const std::string &entity)  
*Returns a current trial balance report.*

### Private Attributes

- [gldb::DBConn m\\_dbc](#)
- const std::shared\_ptr< const [DBSQLStatements](#) > [m\\_sql](#)
- const std::vector< std::string > [m\\_tables](#)
- const std::vector< std::string > [m\\_views](#)

## 9.17.1 Detailed Description

General ledger database class.

## 9.17.2 Constructor & Destructor Documentation

- 9.17.2.1 [GLDatabase::GLDatabase](#) ( const std::string & *database*, const std::string & *hostname*, const std::string & *username*, const std::string & *password* )

Constructor.

#### Parameters

<i>database</i>	Database name.
<i>hostname</i>	Hostname of database machine.
<i>username</i>	Username to log into database.
<i>password</i>	Password to log into database.

#### Exceptions

<a href="#">GLDBException</a>	on error.
-------------------------------	-----------

## 9.17.2.2 GLDatabase::~~GLDatabase ( )

Destructor

## 9.17.3 Member Function Documentation

## 9.17.3.1 std::string GLDatabase::backend ( ) [static]

Returns the backend database implementation.

This may be called to discover which database platform support has been compiled into the application.

Returns

A string containing the database platform name.

## 9.17.3.2 void GLDatabase::create\_structure ( )

Creates the database structure.

Exceptions

<a href="#">GLDBException</a>	on error.
-------------------------------	-----------

## 9.17.3.3 GLUser GLDatabase::create\_user ( glldb::Table &amp; table ) [private]

Creates a user from a query table.

Provided because the public functions can get a user either from an ID or a name, this function contains the common functionality.

Parameters

<i>table</i>	A table from the appropriate query.
--------------	-------------------------------------

Returns

The new user.

## 9.17.3.4 GLReport GLDatabase::current\_trial\_balance\_report ( const std::string &amp; entity ) [private]

Returns a current trial balance report.

Parameters

<i>entity</i>	The entity for which to run the report, or an empty string for all entities.
---------------	--

Returns

A [GLReport](#) object with the report.

## 9.17.3.5 void GLDatabase::destroy\_structure ( )

Destroys the database structure.

## Exceptions

<a href="#"><i>GLDBException</i></a>	on error.
--------------------------------------	-----------

9.17.3.6 **GLUser** **GLDatabase::get\_user\_by\_id** ( const std::string & *user\_id* )

Returns a user from an ID.

## Parameters

<i>user_id</i>	The user ID.
----------------	--------------

## Returns

The user.

## Exceptions

<a href="#"><i>GLDBException</i></a>	if the user cannot be found.
--------------------------------------	------------------------------

9.17.3.7 **GLUser** **GLDatabase::get\_user\_by\_username** ( const std::string & *user\_name* )

Returns a user from a user name.

## Parameters

<i>user_name</i>	The user name.
------------------	----------------

## Returns

The user.

## Exceptions

<a href="#"><i>GLDBException</i></a>	if the user cannot be found.
--------------------------------------	------------------------------

9.17.3.8 void **GLDatabase::grant** ( const **GLUser** & *user*, const std::string & *perm* )

Grants a user a permission.

## Parameters

<i>user</i>	The user for which to grant.
<i>perm</i>	A string containing the permission to grant.

9.17.3.9 void **GLDatabase::load\_sample\_data** ( const std::string & *dir* )

Loads sample data into the database.

## Parameters

<i>dir</i>	The directory containing the sample data. Individual files in that directory should be named after the table they are intended to populate.
------------	---



## Exceptions

<a href="#"><i>GLDBException</i></a>	on error.
--------------------------------------	-----------

9.17.3.10 GLReport GLDatabase::report ( const std::string & *report\_name*, const std::string & *arg* = " " )

Runs a report.

## Parameters

<i>report_name</i>	The name of the report.
<i>arg</i>	An optional argument.

## Returns

A report object.

9.17.3.11 void GLDatabase::revoke ( const GLUser & *user*, const std::string & *perm* )

Revokes a permission from a user.

## Parameters

<i>user</i>	The user for which to revoke.
<i>perm</i>	A string containing the permission to revoke.

9.17.3.12 void GLDatabase::update\_user ( const GLUser & *user* )

Updates a user's details.

## Parameters

<i>user</i>	The user object.
-------------	------------------

## 9.17.4 Member Data Documentation

## 9.17.4.1 glldb::DBConn genleg::GLDatabase::m\_dbc [private]

Database connection

## 9.17.4.2 const std::shared\_ptr&lt;const DBSQLStatements&gt; genleg::GLDatabase::m\_sql [private]

SQL statements object

## 9.17.4.3 const std::vector&lt;std::string&gt; genleg::GLDatabase::m\_tables [private]

Vector containing database table names

## 9.17.4.4 const std::vector&lt;std::string&gt; genleg::GLDatabase::m\_views [private]

Vector containing database view names

The documentation for this class was generated from the following files:

- [lib/gldb/gldatabase.h](#)
- [lib/gldb/gldatabase.cpp](#)

## 9.18 genleg::GLDBException Class Reference

Base general ledger database exceptionc class.

```
#include <glexception.h>
```

### Public Member Functions

- [GLDBException](#) (const std::string &msg)  
*Constructor.*

### 9.18.1 Detailed Description

Base general ledger database exceptionc class.

### 9.18.2 Constructor & Destructor Documentation

9.18.2.1 `genleg::GLDBException::GLDBException ( const std::string & msg ) [inline],[explicit]`

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

- [lib/gldb/glexception.h](#)

## 9.19 genleg::GLReport Class Reference

General ledger report class.

```
#include <glreport.h>
```

### Public Member Functions

- [GLReport](#) (const std::string &report)
- [~GLReport](#) ()

### Private Attributes

- const std::string [m\\_report\\_text](#)

## Friends

- `std::ostream & operator<< (std::ostream &out, const GLReport &report)`  
*Overridden << operator for printing a report.*

### 9.19.1 Detailed Description

General ledger report class.

### 9.19.2 Constructor & Destructor Documentation

9.19.2.1 `genleg::GLReport::GLReport ( const std::string &report )` `[inline]`

Constructor

9.19.2.2 `genleg::GLReport::~GLReport ( )` `[inline]`

Destructor

### 9.19.3 Friends And Related Function Documentation

9.19.3.1 `std::ostream& operator<< ( std::ostream &out, const GLReport &report )` `[friend]`

Overridden << operator for printing a report.

#### Parameters

<i>out</i>	The ostream to which to print.
<i>report</i>	A reference to the report.

#### Returns

A reference to `out`.

### 9.19.4 Member Data Documentation

9.19.4.1 `const std::string genleg::GLReport::m_report_text` `[private]`

The main report text

The documentation for this class was generated from the following file:

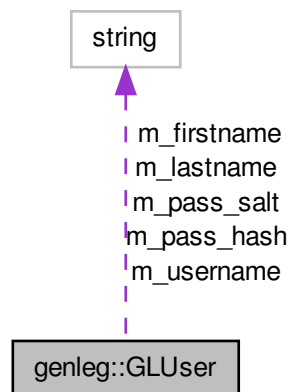
- `lib/gldb/glreport.h`

## 9.20 genleg::GLUser Class Reference

General ledger user class.

```
#include <gluser.h>
```

Collaboration diagram for genleg::GLUser:



## Public Member Functions

- [GLUser](#) (const std::string &[id](#), const std::string &[username](#), const std::string &[firstname](#), const std::string &[lastname](#), const std::string &[pass\\_hash](#), const std::string &[pass\\_salt](#), std::vector< std::string > &&perms, const bool [enabled](#))  
*Constructor.*
- [~GLUser](#) ()
- const std::string & [id](#) () const  
*Returns the user ID.*
- const std::string & [username](#) () const  
*Returns the username.*
- const std::string & [firstname](#) () const  
*Returns the user's first name.*
- const std::string & [lastname](#) () const  
*Returns the user's last name.*
- const std::string & [pass\\_hash](#) () const  
*Returns the user's hashed password.*
- const std::string & [pass\\_salt](#) () const  
*Returns the user's password salt.*
- const std::vector< std::string > & [permissions](#) () const  
*Returns the permissions for a user.*
- bool [enabled](#) () const  
*Returns the user's enabled status.*
- void [set\\_username](#) (const std::string &new\_username)  
*Sets a user's username.*
- void [set\\_firstname](#) (const std::string &new\_firstname)  
*Sets a user's first name.*
- void [set\\_lastname](#) (const std::string &new\_lastname)  
*Sets a user's last name.*
- void [set\\_enabled](#) (const bool new\_enabled)  
*Sets a user's enabled status.*

- void [set\\_password](#) (const std::string &new\_pass)  
*Sets a user's password hash and salt.*
- bool [check\\_password](#) (const std::string &check\_pass)  
*Checks a password against the user's hash.*

### Private Attributes

- const std::string [m\\_id](#)
- std::string [m\\_username](#)
- std::string [m\\_firstname](#)
- std::string [m\\_lastname](#)
- std::string [m\\_pass\\_hash](#)
- std::string [m\\_pass\\_salt](#)
- const std::vector< std::string > [m\\_perms](#)
- bool [m\\_enabled](#)

## 9.20.1 Detailed Description

General ledger user class.

## 9.20.2 Constructor & Destructor Documentation

9.20.2.1 `GLUser::GLUser ( const std::string & id, const std::string & username, const std::string & firstname, const std::string & lastname, const std::string & pass_hash, const std::string & pass_salt, std::vector< std::string > && perms, const bool enabled )`

Constructor.

### Parameters

<i>id</i>	User ID
<i>username</i>	Username
<i>firstname</i>	First name
<i>lastname</i>	Last name
<i>pass_hash</i>	The hashed password
<i>pass_salt</i>	The salt for the hashed password
<i>perms</i>	Vector of user permissions
<i>enabled</i>	true if user is enabled, false otherwise.

9.20.2.2 `GLUser::~~GLUser ( )`

Destructor

## 9.20.3 Member Function Documentation

9.20.3.1 `bool GLUser::check_password ( const std::string & check_pass )`

Checks a password against the user's hash.

### Parameters

<i>check_pass</i>	The password to check, must be > 8 characters.
-------------------	--

**Returns**

`true` is the password matches, `false` otherwise.

**9.20.3.2 `bool GLUser::enabled ( ) const`**

Returns the user's enabled status.

**Returns**

The user's enabled status.

**9.20.3.3 `const std::string & GLUser::firstname ( ) const`**

Returns the user's first name.

**Returns**

The user's first name.

**9.20.3.4 `const std::string & GLUser::id ( ) const`**

Returns the user ID.

**Returns**

The user ID.

**9.20.3.5 `const std::string & GLUser::lastname ( ) const`**

Returns the user's last name.

**Returns**

The user's last name.

**9.20.3.6 `const std::string & GLUser::pass_hash ( ) const`**

Returns the user's hashed password.

**Returns**

The user's hashed password.

**9.20.3.7 `const std::string & GLUser::pass_salt ( ) const`**

Returns the user's password salt.

**Returns**

The user's password salt.

9.20.3.8 `const std::vector< std::string > & GLUser::permissions ( ) const`

Returns the permissions for a user.

Returns

A vector of strings containing the names of the permissions held by the user.

9.20.3.9 `void GLUser::set_enabled ( const bool new_enabled )`

Sets a user's enabled status.

Parameters

<i>new_enabled</i>	The user's new enabled status.
--------------------	--------------------------------

9.20.3.10 `void GLUser::set_firstname ( const std::string & new_firstname )`

Sets a user's first name.

Parameters

<i>new_firstname</i>	The user's new first name.
----------------------	----------------------------

9.20.3.11 `void GLUser::set_lastname ( const std::string & new_lastname )`

Sets a user's last name.

Parameters

<i>new_lastname</i>	The user's new last name.
---------------------	---------------------------

9.20.3.12 `void GLUser::set_password ( const std::string & new_pass )`

Sets a user's password hash and salt.

Parameters

<i>new_pass</i>	The new password, must be > 8 characters.
-----------------	---

9.20.3.13 `void GLUser::set_username ( const std::string & new_username )`

Sets a user's username.

Parameters

<i>new_username</i>	The user's new username.
---------------------	--------------------------

9.20.3.14 `const std::string & GLUser::username ( ) const`

Returns the username.

**Returns**

The username.

**9.20.4 Member Data Documentation**

**9.20.4.1** `bool genleg::GLUser::m_enabled` [private]

User's enabled status

**9.20.4.2** `std::string genleg::GLUser::m_firstname` [private]

User's first name

**9.20.4.3** `const std::string genleg::GLUser::m_id` [private]

User ID

**9.20.4.4** `std::string genleg::GLUser::m_lastname` [private]

User's last name

**9.20.4.5** `std::string genleg::GLUser::m_pass_hash` [private]

User's hashed password

**9.20.4.6** `std::string genleg::GLUser::m_pass_salt` [private]

User's password salt

**9.20.4.7** `const std::vector<std::string> genleg::GLUser::m_perms` [private]

List of permissions

**9.20.4.8** `std::string genleg::GLUser::m_username` [private]

Username

The documentation for this class was generated from the following files:

- [lib/gldb/gluser.h](#)
- [lib/gldb/gluser.cpp](#)
- [lib/gldb/gluser\\_pass.cpp](#)

**9.21 gldb::MySQLResult Class Reference**

MySQL result structure class.

```
#include <dbconn_mysql_result.h>
```



## Public Member Functions

- [MySQLResult](#) (MySQL \*conn)  
*Constructor.*
- [~MySQLResult](#) ()
- [MySQLResult](#) (const [MySQLResult](#) &result)
- [MySQLResult](#) ([MySQLResult](#) &&result)
- [MySQLResult](#) & operator= (const [MySQLResult](#) &result)
- [MySQLResult](#) & operator= ([MySQLResult](#) &&result)
- MySQL\_RES \* [result](#) ()  
*Returns the MYSQL\_RES pointer.*
- unsigned int [num\\_fields](#) () const  
*Returns the number of fields in the result set.*

## Private Attributes

- MySQL\_RES \* [m\\_result](#)
- unsigned int [m\\_num\\_fields](#)

### 9.21.1 Detailed Description

MySQL result structure class.

### 9.21.2 Constructor & Destructor Documentation

#### 9.21.2.1 MySQLResult::MySQLResult ( MySQL \* conn ) [explicit]

Constructor.

##### Parameters

<i>conn</i>	MySQL connection
-------------	------------------

##### Exceptions

<a href="#">DBConnCouldNotQuery</a>	on failure
-------------------------------------	------------

#### 9.21.2.2 gldb::MySQLResult::~~MySQLResult ( )

Destructor

#### 9.21.2.3 gldb::MySQLResult::MySQLResult ( const MySQLResult & result )

Deleted copy constructor

#### 9.21.2.4 gldb::MySQLResult::MySQLResult ( MySQLResult && result )

Deleted move constructor

### 9.21.3 Member Function Documentation

**9.21.3.1** `unsigned int glldb::MySQLResult::num_fields ( ) const [inline]`

Returns the number of fields in the result set.

#### Returns

The number of fields in the result set.

**9.21.3.2** `MySQLResult& glldb::MySQLResult::operator= ( const MySQLResult & result )`

Deleted copy assignment operator

**9.21.3.3** `MySQLResult& glldb::MySQLResult::operator= ( MySQLResult && result )`

Deleted move assignment operator

**9.21.3.4** `MYSQL_RES* glldb::MySQLResult::result ( ) [inline]`

Returns the MYSQL\_RES pointer.

#### Returns

The MYSQL\_RES pointer.

## 9.21.4 Member Data Documentation

**9.21.4.1** `unsigned int glldb::MySQLResult::m_num_fields [private]`

The number of fields in the result set

**9.21.4.2** `MYSQL_RES* glldb::MySQLResult::m_result [private]`

The MYSQL\_RES pointer

The documentation for this class was generated from the following files:

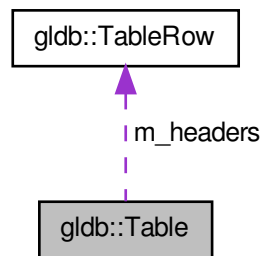
- [lib/database\\_imp/mysql/dbconn\\_mysql\\_result.h](#)
- [lib/database\\_imp/mysql/dbconn\\_mysql\\_result.cpp](#)

## 9.22 glldb::Table Class Reference

Database table class.

```
#include <table.h>
```

Collaboration diagram for glldb::Table:



## Public Member Functions

- [Table](#) (const [TableRow](#) &headers)  
*Constructor.*
- [Table](#) ([TableRow](#) &&headers)  
*Constructor with move semantics.*
- [Table](#) (const [Table](#) &table)  
*Copy constructor.*
- [Table](#) ([Table](#) &&table)  
*Move constructor.*
- [Table](#) & [operator=](#) (const [Table](#) &table)  
*Copy assignment operator.*
- [Table](#) & [operator=](#) ([Table](#) &&table)  
*Move assignment operator.*
- [~Table](#) ()
- size\_t [num\\_fields](#) () const  
*Returns the number of fields in each row.*
- size\_t [num\\_records](#) () const  
*Returns the number of record in the table.*
- iterator [begin](#) ()  
*Returns iterator for beginning.*
- iterator [end](#) ()  
*Returns iterator for end plus one.*
- const\_iterator [begin](#) () const  
*Returns const iterator for beginning.*
- const\_iterator [end](#) () const  
*Returns const iterator for end plus one.*
- void [set\\_quoted](#) (const std::vector< bool > &vec)  
*Sets the quote flags for the records.*
- void [set\\_quoted](#) (std::vector< bool > &&vec)  
*Sets the quote flags for the records with move semantics.*
- const [TableRow](#) & [get\\_headers](#) () const  
*Returns the field names.*
- const [TableRow](#) & [operator\[\]](#) (const size\_t idx) const

*Overloaded index operator.*

- void [append\\_record](#) (const [TableRow](#) &new\_record)  
*Appends a record to the table.*
- void [append\\_record](#) ([TableRow](#) &&new\_record)  
*Appends a record to the table with move semantics.*
- std::string [insert\\_query](#) (const std::string &table\_name, const size\_t idx)  
*Creates an SQL INSERT query from a table record.*
- std::string [get\\_field](#) (const std::string &field\_name, const size\_t row\_index)  
*Gets a field from a record by field name.*

## Static Public Member Functions

- static [Table](#) [create\\_from\\_file](#) (const std::string &filename, const char delim)  
*Creates a table from an input file.*

## Private Attributes

- [TableRow](#) [m\\_headers](#)
- std::vector< [TableRow](#) > [m\\_records](#)
- std::vector< bool > [m\\_quoted](#)

### 9.22.1 Detailed Description

Database table class.

### 9.22.2 Constructor & Destructor Documentation

#### 9.22.2.1 [Table::Table](#) ( const [TableRow](#) & *headers* ) [explicit]

Constructor.

Parameters

<i>headers</i>	<a href="#">Table</a> row containing field names.
----------------	---

#### 9.22.2.2 [Table::Table](#) ( [TableRow](#) && *headers* ) [explicit]

Constructor with move semantics.

Parameters

<i>headers</i>	<a href="#">Table</a> row containing field names.
----------------	---

#### 9.22.2.3 [Table::Table](#) ( const [Table](#) & *table* )

Copy constructor.

**Bug** 'explicit' removed from here after failure to compile at end of MySQL query function.

## Parameters

<i>table</i>	<a href="#">Table</a> to copy.
--------------	--------------------------------

9.22.2.4 Table::Table ( Table && *table* )

Move constructor.

## Parameters

<i>table</i>	<a href="#">Table</a> to move.
--------------	--------------------------------

## 9.22.2.5 Table::~~Table ( )

Destructor

## 9.22.3 Member Function Documentation

9.22.3.1 void Table::append\_record ( const TableRow & *new\_record* )

Appends a record to the table.

## Parameters

<i>new_record</i>	The record to append.
-------------------	-----------------------

9.22.3.2 void Table::append\_record ( TableRow && *new\_record* )

Appends a record to the table with move semantics.

## Parameters

<i>new_record</i>	The record to append.
-------------------	-----------------------

## 9.22.3.3 iterator glldb::Table::begin ( ) [inline]

Returns iterator for beginning.

## Returns

Iterator for beginning.

## 9.22.3.4 const\_iterator glldb::Table::begin ( ) const [inline]

Returns const iterator for beginning.

## Returns

Const iterator for beginning.

### 9.22.3.5 Table Table::create\_from\_file ( const std::string & *filename*, const char *delim* ) [static]

Creates a table from an input file.

#### Parameters

<i>filename</i>	The name of the input file.
<i>delim</i>	The delimiting character.

#### Returns

The table.

#### Exceptions

<a href="#"><i>TableBadInputFile</i></a>	on badly formed input file.
<a href="#"><i>TableCouldNotOpenInputFile</i></a>	on bad filename.

### 9.22.3.6 iterator glldb::Table::end ( ) [inline]

Returns iterator for end plus one.

#### Returns

Iterator for end plus one.

### 9.22.3.7 const\_iterator glldb::Table::end ( ) const [inline]

Returns const iterator for end plus one.

#### Returns

Const iterator for end plus one.

### 9.22.3.8 std::string Table::get\_field ( const std::string & *field\_name*, const size\_t *row\_index* )

Gets a field from a record by field name.

#### Parameters

<i>field_name</i>	The name of the field.
<i>row_index</i>	The index of the row.

#### Returns

The contents of the field.

#### Exceptions

<a href="#"><i>TableNoSuchField</i></a>	if <i>field_name</i> is not a valid field name.
<a href="#"><i>TableNoSuchRecord</i></a>	if there is no record at index <i>row_index</i> .

**9.22.3.9** `const TableRow& glldb::Table::get_headers ( ) const [inline]`

Returns the field names.

**Returns**

The field names.

**9.22.3.10** `std::string Table::insert_query ( const std::string & table_name, const size_t idx )`

Creates an SQL INSERT query from a table record.

**Parameters**

<i>table_name</i>	The name of the table into which to INSERT.
<i>idx</i>	The index of the record.

**Returns**

A string containing the query.

**9.22.3.11** `size_t glldb::Table::num_fields ( ) const [inline]`

Returns the number of fields in each row.

**Returns**

The number of fields in each row.

**9.22.3.12** `size_t glldb::Table::num_records ( ) const [inline]`

Returns the number of record in the table.

**Returns**

The number of records in the table.

**9.22.3.13** `Table & Table::operator= ( const Table & table )`

Copy assignment operator.

**Parameters**

<i>table</i>	<a href="#">Table</a> to copy.
--------------	--------------------------------

**Returns**

Reference to the assigned-to table.

**9.22.3.14** `Table & Table::operator= ( Table && table )`

Move assignment operator.

## Parameters

<i>table</i>	<a href="#">Table</a> to move.
--------------	--------------------------------

## Returns

Reference to the assigned-to table.

#### 9.22.3.15 `const TableRow & Table::operator[] ( const size_t idx ) const`

Overloaded index operator.

## Parameters

<i>idx</i>	The zero-based index of the record.
------------	-------------------------------------

## Returns

The selected record.

#### 9.22.3.16 `void Table::set_quoted ( const std::vector< bool > & vec )`

Sets the quote flags for the records.

## Parameters

<i>vec</i>	A vector of bools. The size must match the size of the records.
------------	---

#### 9.22.3.17 `void Table::set_quoted ( std::vector< bool > && vec )`

Sets the quote flags for the records with move semantics.

## Parameters

<i>vec</i>	A vector of bools. The size must match the size of the records.
------------	---

### 9.22.4 Member Data Documentation

#### 9.22.4.1 `TableRow glldb::Table::m_headers` `[private]`

The names of the fields

#### 9.22.4.2 `std::vector<bool> glldb::Table::m_quoted` `[private]`

A vector to show if fields should be quoted for INSERT

#### 9.22.4.3 `std::vector<TableRow> glldb::Table::m_records` `[private]`

A vector of the records

The documentation for this class was generated from the following files:

- lib/database/[table.h](#)



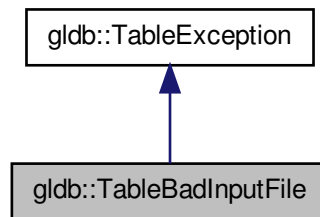
- [lib/database/table.cpp](#)

## 9.23 glldb::TableBadInputFile Class Reference

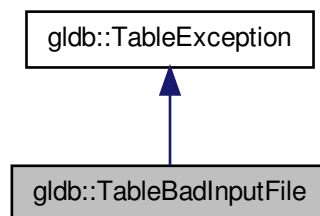
Could not connect to database exception class.

```
#include <table.h>
```

Inheritance diagram for glldb::TableBadInputFile:



Collaboration diagram for glldb::TableBadInputFile:



### Public Member Functions

- [TableBadInputFile](#) (const std::string &msg)  
*Constructor.*

#### 9.23.1 Detailed Description

Could not connect to database exception class.

#### 9.23.2 Constructor & Destructor Documentation

### 9.23.2.1 glldb::TableBadInputFile::TableBadInputFile ( const std::string & msg ) [inline], [explicit]

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

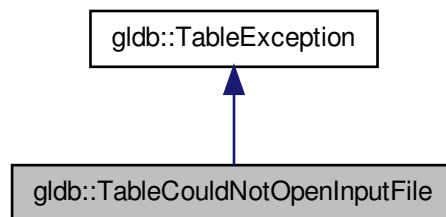
- lib/database/[table.h](#)

## 9.24 glldb::TableCouldNotOpenInputFile Class Reference

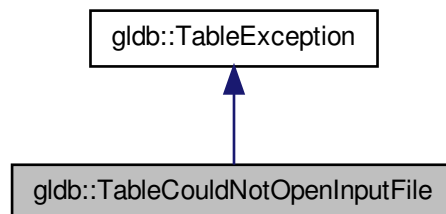
Could not connect to database exception class.

```
#include <table.h>
```

Inheritance diagram for glldb::TableCouldNotOpenInputFile:



Collaboration diagram for glldb::TableCouldNotOpenInputFile:



## Public Member Functions

- [TableCouldNotOpenInputFile](#) (const std::string &msg)  
*Constructor.*

### 9.24.1 Detailed Description

Could not connect to database exception class.

### 9.24.2 Constructor & Destructor Documentation

9.24.2.1 `gldb::TableCouldNotOpenInputFile::TableCouldNotOpenInputFile ( const std::string & msg ) [inline], [explicit]`

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

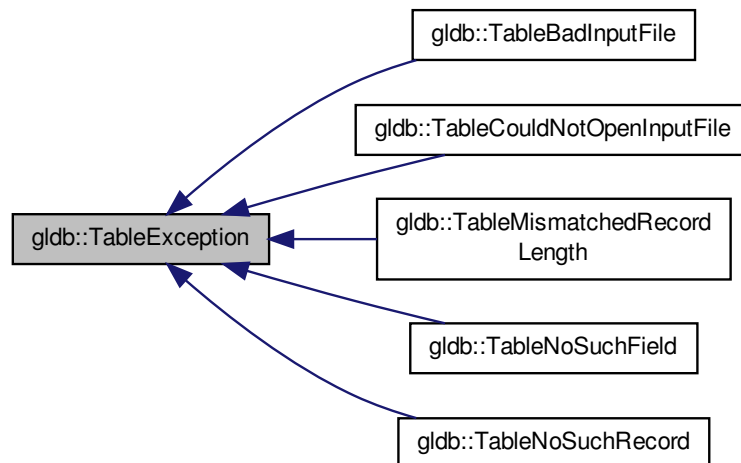
- `lib/database/table.h`

## 9.25 gldb::TableException Class Reference

Base database connection exception class.

```
#include <table.h>
```

Inheritance diagram for `gldb::TableException`:



### Public Member Functions

- `TableException (const std::string &msg)`

*Constructor.*

### 9.25.1 Detailed Description

Base database connection exception class.

### 9.25.2 Constructor & Destructor Documentation

#### 9.25.2.1 `gldb::TableException::TableException ( const std::string & msg ) [inline],[explicit]`

Constructor.

Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

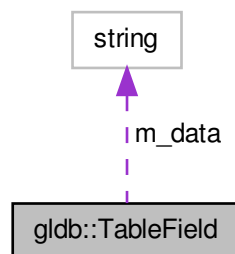
- [lib/database/table.h](#)

## 9.26 `gldb::TableField` Class Reference

Database table field class.

```
#include <tablefield.h>
```

Collaboration diagram for `gldb::TableField`:



### Public Member Functions

- [`TableField`](#) (`const char *data`)  
*Constructor accepting `const char * data`.*
- [`TableField`](#) (`const std::string &data`)  
*Constructor accepting `std::string data`.*
- [`TableField`](#) (`std::string &&data`)  
*Constructor accepting `std::string data` with move semantics.*
- [`TableField`](#) (`const TableField &field`)  
*Copy constructor.*
- [`TableField`](#) (`TableField &&field`)  
*Move constructor.*

- `~TableField ()`
- `size_t length () const`  
*Returns the length of the field.*
- `operator std::string () const`  
*Overridden conversion operator.*
- `TableField & operator= (const char *data)`  
*Overridden assignment operator for `const char *`.*
- `TableField & operator= (const std::string &data)`  
*Overridden assignment operator for `std::string`.*
- `TableField & operator= (std::string &&data)`  
*Overridden assignment operator for `std::string` with move semantics.*
- `TableField & operator= (const TableField &field)`  
*Overridden copy assignment operator.*
- `TableField & operator= (TableField &&field)`  
*Overridden move assignment operator.*
- `char & operator[] (const size_t idx)`  
*Overridden index operator.*
- `const char & operator[] (const size_t idx) const`  
*Overridden index operator.*
- `TableField & operator+= (const char c)`  
*Overridden compound assignment operator.*
- `TableField & operator+= (const std::string &data)`  
*Overridden compound assignment operator.*

### Private Attributes

- `std::string m_data`

### Friends

- `std::ostream & operator<< (std::ostream &out, const TableField &field)`  
*Overridden << operator for printing a field.*

## 9.26.1 Detailed Description

Database table field class.

## 9.26.2 Constructor & Destructor Documentation

### 9.26.2.1 TableField::TableField ( const char \* data ) [explicit]

Constructor accepting `const char * data`.

#### Parameters

<code>data</code>	The initial contents of the field.
-------------------	------------------------------------

### 9.26.2.2 TableField::TableField ( const std::string & data )

Constructor accepting `std::string data`.

## Parameters

<i>data</i>	The initial contents of the field.
-------------	------------------------------------

9.26.2.3 `TableField::TableField ( std::string && data )`

Constructor accepting `std::string` data with move semantics.

## Parameters

<i>data</i>	The initial contents of the field.
-------------	------------------------------------

9.26.2.4 `TableField::TableField ( const TableField & field )`

Copy constructor.

## Parameters

<i>field</i>	The field from which to copy.
--------------	-------------------------------

9.26.2.5 `TableField::TableField ( TableField && field )`

Move constructor.

## Parameters

<i>field</i>	The field from which to move.
--------------	-------------------------------

9.26.2.6 `TableField::~~TableField ( )`

Destructor

## 9.26.3 Member Function Documentation

9.26.3.1 `size_t glldb::TableField::length ( ) const` `[inline]`

Returns the length of the field.

## Returns

The length of the field.

9.26.3.2 `glldb::TableField::operator std::string ( ) const` `[inline]`

Overridden conversion operator.

Returns the field contents as a string.

9.26.3.3 `TableField & TableField::operator+= ( const char c )`

Overridden compound assignment operator.

## Parameters

<i>c</i>	The character to append to the field.
----------	---------------------------------------

## Returns

A reference to the same field.

**9.26.3.4 TableField & TableField::operator+= ( const std::string & *data* )**

Overridden compound assignment operator.

## Parameters

<i>data</i>	The string to append to the field.
-------------	------------------------------------

## Returns

A reference to the same field.

**9.26.3.5 TableField & TableField::operator= ( const char \* *data* )**

Overridden assignment operator for `const char *`.

## Parameters

<i>data</i>	The new contents of the field.
-------------	--------------------------------

## Returns

A reference to the same field.

**9.26.3.6 TableField & TableField::operator= ( const std::string & *data* )**

Overridden assignment operator for `std::string`.

## Parameters

<i>data</i>	The new contents of the field.
-------------	--------------------------------

## Returns

A reference to the same field.

**9.26.3.7 TableField & TableField::operator= ( std::string && *data* )**

Overridden assignment operator for `std::string` with move semantics.

## Parameters

<i>data</i>	The new contents of the field.
-------------	--------------------------------

**Returns**

A reference to the same field.

**9.26.3.8   `TableField & TableField::operator= ( const TableField & field )`**

Overridden copy assignment operator.

**Parameters**

<i>field</i>	The field to copy.
--------------	--------------------

**Returns**

A reference to the same field.

**9.26.3.9   `TableField & TableField::operator= ( TableField && field )`**

Overridden move assignment operator.

**Parameters**

<i>field</i>	The field to move.
--------------	--------------------

**Returns**

A reference to the same field.

**9.26.3.10   `char& glldb::TableField::operator[] ( const size_t idx )   [inline]`**

Overridden index operator.

**Parameters**

<i>idx</i>	The desired index.
------------	--------------------

**Returns**

A reference to the character at the specified index.

**9.26.3.11   `const char& glldb::TableField::operator[] ( const size_t idx ) const   [inline]`**

Overridden index operator.

**Parameters**

<i>idx</i>	The desired index.
------------	--------------------



**Returns**

A const reference to the character at the specified index.

**9.26.4 Friends And Related Function Documentation**

9.26.4.1 `std::ostream& operator<< ( std::ostream & out, const TableField & field )` [*friend*]

Overridden << operator for printing a field.

**Parameters**

<i>out</i>	The ostream to which to print.
<i>field</i>	A reference to the field.

**Returns**

A reference to `out`.

**9.26.5 Member Data Documentation**

9.26.5.1 `std::string glldb::TableField::m_data` [*private*]

The field contents

The documentation for this class was generated from the following files:

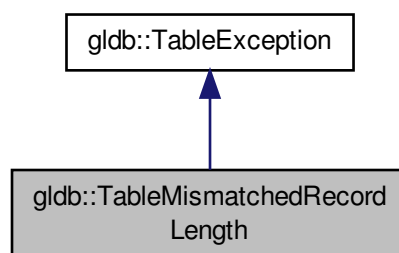
- [lib/database/tablefield.h](#)
- [lib/database/tablefield.cpp](#)

**9.27 glldb::TableMismatchedRecordLength Class Reference**

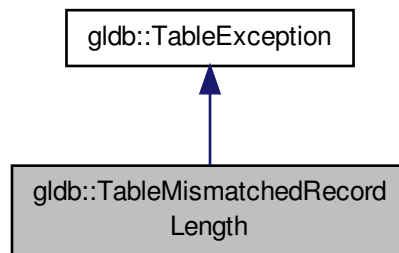
Mismatched record length exception class.

```
#include <table.h>
```

Inheritance diagram for `glldb::TableMismatchedRecordLength`:



Collaboration diagram for `gldb::TableMismatchedRecordLength`:



## Public Member Functions

- [TableMismatchedRecordLength](#) (const std::string &msg)  
*Constructor.*

### 9.27.1 Detailed Description

Mismatched record length exception class.

### 9.27.2 Constructor & Destructor Documentation

**9.27.2.1** `gldb::TableMismatchedRecordLength::TableMismatchedRecordLength ( const std::string & msg ) [inline], [explicit]`

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

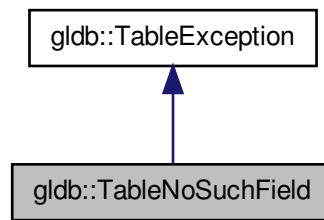
- lib/database/[table.h](#)

## 9.28 gldb::TableNoSuchField Class Reference

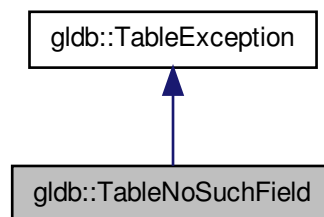
No such field exception class.

```
#include <table.h>
```

Inheritance diagram for gldb::TableNoSuchField:



Collaboration diagram for gldb::TableNoSuchField:



## Public Member Functions

- [TableNoSuchField](#) (const std::string &msg)  
*Constructor.*

### 9.28.1 Detailed Description

No such field exception class.

### 9.28.2 Constructor & Destructor Documentation

9.28.2.1 `gldb::TableNoSuchField::TableNoSuchField ( const std::string & msg ) [inline], [explicit]`

Constructor.

#### Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

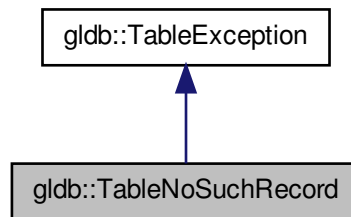
- [lib/database/table.h](#)

## 9.29 glDb::TableNoSuchRecord Class Reference

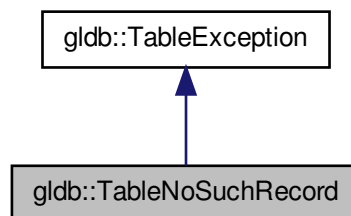
No such record exception class.

```
#include <table.h>
```

Inheritance diagram for glDb::TableNoSuchRecord:



Collaboration diagram for glDb::TableNoSuchRecord:



### Public Member Functions

- [TableNoSuchRecord](#) (const std::string &msg)  
*Constructor.*

#### 9.29.1 Detailed Description

No such record exception class.

#### 9.29.2 Constructor & Destructor Documentation

9.29.2.1 glldb::TableNoSuchRecord::TableNoSuchRecord ( const std::string & msg ) [inline],[explicit]

Constructor.

Parameters

<i>msg</i>	Database error message
------------	------------------------

The documentation for this class was generated from the following file:

- lib/database/[table.h](#)

## 9.30 glldb::TableRow Class Reference

Database table row class.

```
#include <tablerow.h>
```

### Public Member Functions

- [TableRow](#) ()
- [TableRow](#) (const size\_t size)  
*Constructor with initial number of fields.*
- [TableRow](#) (const std::vector< std::string > &vec)  
*Constructor with string vector.*
- [TableRow](#) (std::vector< std::string > &&vec)  
*Constructor with string vector and move semantics.*
- [TableRow](#) (std::initializer\_list< std::string > i)  
*Constructor with std::string initializer list.*
- [TableRow](#) (const [TableRow](#) &row)  
*Copy constructor.*
- [TableRow](#) ([TableRow](#) &&row)  
*Move constructor.*
- [TableRow](#) & operator= (const [TableRow](#) &row)  
*Copy assignment operator.*
- [TableRow](#) & operator= ([TableRow](#) &&row)  
*Move assignment operator.*
- [~TableRow](#) ()
- size\_t size () const  
*Returns the number of fields.*
- iterator [begin](#) ()  
*Returns iterator for beginning.*
- iterator [end](#) ()  
*Returns iterator for end plus one.*
- const\_iterator [begin](#) () const  
*Returns const iterator for beginning.*
- const\_iterator [end](#) () const  
*Returns const iterator for end plus one.*
- [TableField](#) & operator[] (const size\_t idx)  
*Overridden index operator.*
- const [TableField](#) & operator[] (const size\_t idx) const  
*Overridden index operator.*

- void `append_field` (const char \*new\_field)  
*Appends a field to the row.*
- void `append_field` (const std::string &new\_field)  
*Appends a field to the row.*
- void `append_field` (std::string &&new\_field)  
*Appends a field to the row with move semantics.*
- void `append_field` (const `TableField` &new\_field)  
*Appends a field to the row.*
- void `append_field` (`TableField` &&new\_field)  
*Appends a field to the row with move semantics.*
- void `print` (std::ostream &stream) const  
*Prints a row.*
- std::string `record_string` (const std::vector< bool > &quoted) const  
*Creates a comma separated string of fields.*
- std::string `record_string` () const  
*Creates an unquoted comma separated string of fields.*

## Private Attributes

- std::vector< `TableField` > `m_fields`

### 9.30.1 Detailed Description

Database table row class.

### 9.30.2 Constructor & Destructor Documentation

#### 9.30.2.1 `TableRow::TableRow ( )`

Default constructor

#### 9.30.2.2 `TableRow::TableRow ( const size_t size )` `[explicit]`

Constructor with initial number of fields.

##### Parameters

<code>size</code>	The initial number of fields.
-------------------	-------------------------------

#### 9.30.2.3 `TableRow::TableRow ( const std::vector< std::string > & vec )` `[explicit]`

Constructor with string vector.

##### Parameters

<code>vec</code>	The vector.
------------------	-------------

#### 9.30.2.4 `TableRow::TableRow ( std::vector< std::string > && vec )` `[explicit]`

Constructor with string vector and move semantics.

## Parameters

<i>vec</i>	The vector.
------------	-------------

9.30.2.5 TableRow::TableRow ( std::initializer\_list< std::string > *i* ) [explicit]

Constructor with std::string initializer list.

## Parameters

<i>i</i>	The initializer list.
----------	-----------------------

9.30.2.6 TableRow::TableRow ( const TableRow & *row* )

Copy constructor.

## Parameters

<i>row</i>	The row to copy.
------------	------------------

9.30.2.7 TableRow::TableRow ( TableRow && *row* )

Move constructor.

## Parameters

<i>row</i>	The row to move.
------------	------------------

## 9.30.2.8 TableRow::~~TableRow ( )

Destructor

## 9.30.3 Member Function Documentation

9.30.3.1 void TableRow::append\_field ( const char \* *new\_field* )

Appends a field to the row.

## Parameters

<i>new_field</i>	The contents of the new field.
------------------	--------------------------------

9.30.3.2 void TableRow::append\_field ( const std::string & *new\_field* )

Appends a field to the row.

## Parameters

<i>new_field</i>	The contents of the new field.
------------------	--------------------------------

### 9.30.3.3 void TableRow::append\_field ( std::string && *new\_field* )

Appends a field to the row with move semantics.

#### Parameters

<i>new_field</i>	The contents of the new field.
------------------	--------------------------------

### 9.30.3.4 void TableRow::append\_field ( const TableField & *new\_field* )

Appends a field to the row.

#### Parameters

<i>new_field</i>	A field from which to copy.
------------------	-----------------------------

### 9.30.3.5 void TableRow::append\_field ( TableField && *new\_field* )

Appends a field to the row with move semantics.

#### Parameters

<i>new_field</i>	A field from which to copy.
------------------	-----------------------------

### 9.30.3.6 iterator glldb::TableRow::begin ( ) [inline]

Returns iterator for beginning.

#### Returns

Iterator for beginning.

### 9.30.3.7 const\_iterator glldb::TableRow::begin ( ) const [inline]

Returns const iterator for beginning.

#### Returns

Const iterator for beginning.

### 9.30.3.8 iterator glldb::TableRow::end ( ) [inline]

Returns iterator for end plus one.

#### Returns

Iterator for end plus one.

### 9.30.3.9 const\_iterator glldb::TableRow::end ( ) const [inline]

Returns const iterator for end plus one.



**Returns**

Const iterator for end plus one.

**9.30.3.10 TableRow & TableRow::operator= ( const TableRow & row )**

Copy assignment operator.

**Parameters**

<i>row</i>	The row to copy.
------------	------------------

**Returns**

A reference to the assigned-to row.

**9.30.3.11 TableRow & TableRow::operator= ( TableRow && row )**

Move assignment operator.

**Parameters**

<i>row</i>	The row to move.
------------	------------------

**Returns**

A reference to the assigned-to row.

**9.30.3.12 TableField& glldb::TableRow::operator[] ( const size\_t idx ) [inline]**

Overridden index operator.

**Parameters**

<i>idx</i>	The zero-based index of the field.
------------	------------------------------------

**Returns**

A reference to the field at the specified index.

**9.30.3.13 const TableField& glldb::TableRow::operator[] ( const size\_t idx ) const [inline]**

Overridden index operator.

**Parameters**

<i>idx</i>	The zero-based index of the field.
------------	------------------------------------

**Returns**

A const reference to the field at the specified index.

#### 9.30.3.14 void TableRow::print ( std::ostream & *stream* ) const

Prints a row.

##### Parameters

<i>stream</i>	The ostream to which to print.
---------------	--------------------------------

#### 9.30.3.15 std::string TableRow::record\_string ( const std::vector< bool > & *quoted* ) const

Creates a comma separated string of fields.

##### Parameters

<i>quoted</i>	A vector of <code>bool</code> , for each field <code>true</code> means that field will be enclosed in single quotes in the comma separated string, <code>false</code> means it will not be.
---------------	---

##### Returns

The comma separated string.

#### 9.30.3.16 std::string TableRow::record\_string ( ) const

Creates an unquoted comma separated string of fields.

##### Returns

The unquoted comma separated string.

#### 9.30.3.17 size\_t glldb::TableRow::size ( ) const [inline]

Returns the number of fields.

##### Returns

The number of fields.

### 9.30.4 Member Data Documentation

#### 9.30.4.1 std::vector<TableField> glldb::TableRow::m\_fields [private]

A vector of fields

The documentation for this class was generated from the following files:

- lib/database/[tablerow.h](#)
- lib/database/[tablerow.cpp](#)

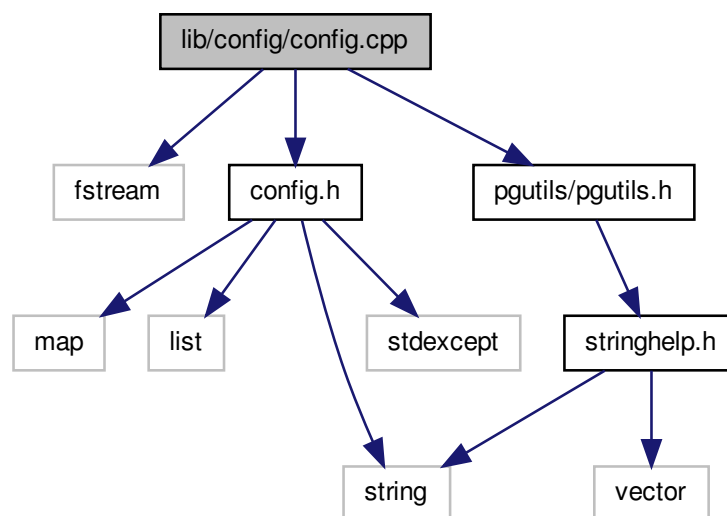
## Chapter 10

# File Documentation

### 10.1 lib/config/config.cpp File Reference

Implementation of program configurations class.

```
#include <fstream>
#include "config.h"
#include "pgutils/pgutils.h"
Include dependency graph for config.cpp:
```



#### 10.1.1 Detailed Description

Implementation of program configurations class.

Author

Paul Griffiths

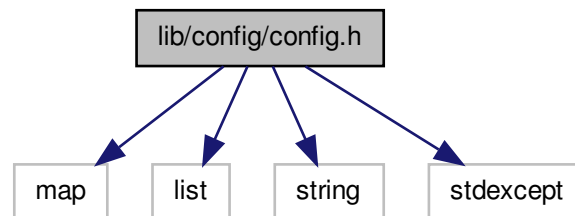
## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

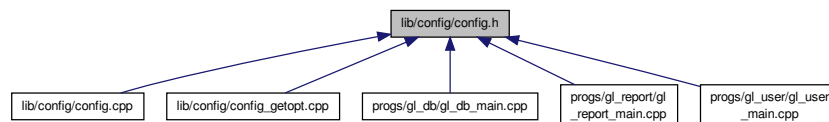
## 10.2 lib/config/config.h File Reference

Interface to program configurations class.

```
#include <map>
#include <list>
#include <string>
#include <stdexcept>
Include dependency graph for config.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class `genleg::ConfigException`  
*Configuration module exception base class.*
- class `genleg::ConfigOptionNotSet`  
*Exception class for option not set.*
- class `genleg::ConfigBadOption`  
*Exception class for bad provided option.*
- class `genleg::ConfigCouldNotOpenFile`  
*Exception class for when conf file cannot be opened.*
- class `genleg::ConfigBadConfigFile`  
*Exception class for badly formed configuration file.*
- class `genleg::Config`  
*Configuration options class.*

## Enumerations

- enum [genleg::Argument](#)

*Enumeration class for option argument specifications.*

### 10.2.1 Detailed Description

Interface to program configurations class.

#### Author

Paul Griffiths

#### Copyright

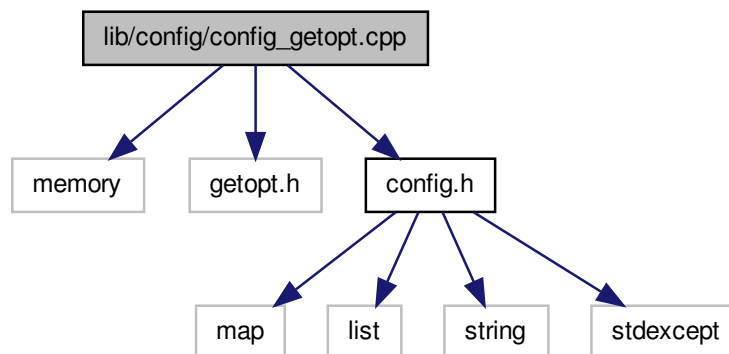
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.3 lib/config/config\_getopt.cpp File Reference

Implementation of command line functionality.

```
#include <memory>
#include <getopt.h>
#include "config.h"
```

Include dependency graph for config\_getopt.cpp:



## Macros

- `#define \_XOPEN\_SOURCE 600`

### 10.3.1 Detailed Description

Implementation of command line functionality. Included in separate file to isolate usage of non-standard getopt library.

**Author**

Paul Griffiths

**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

**10.3.2 Macro Definition Documentation****10.3.2.1 #define \_XOPEN\_SOURCE 600**

UNIX feature test macro for getopt library

**10.4 lib/database/data\_structures.h File Reference**

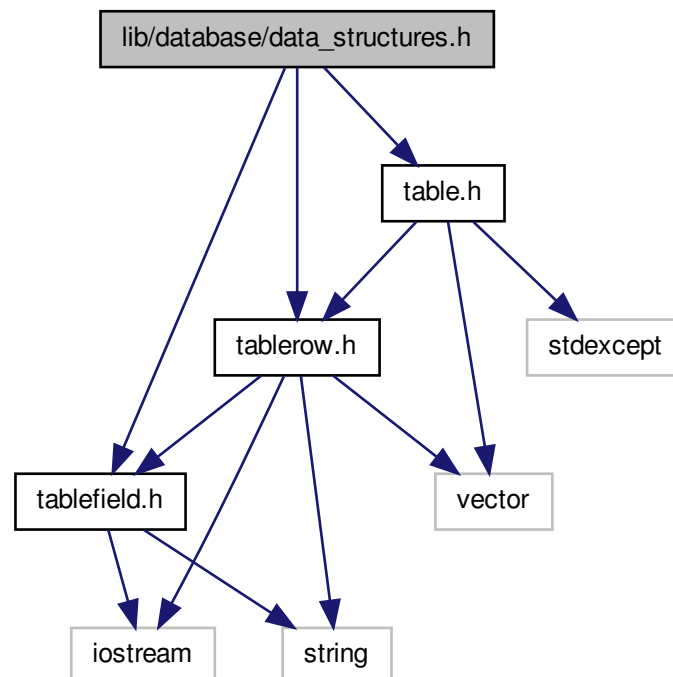
Main interface to database data structures.

```
#include "tablefield.h"
```

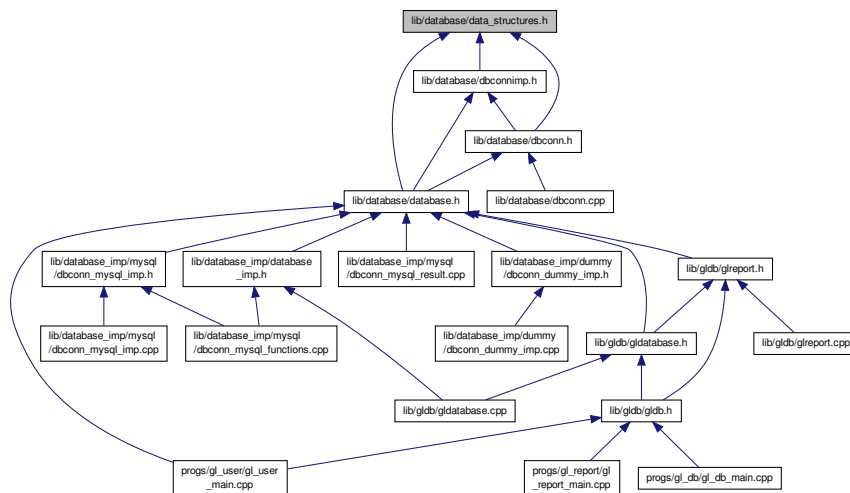
```
#include "tablerow.h"
```

```
#include "table.h"
```

Include dependency graph for data\_structures.h:



This graph shows which files directly or indirectly include this file:



### 10.4.1 Detailed Description

Main interface to database data structures.

#### Author

Paul Griffiths

#### Copyright

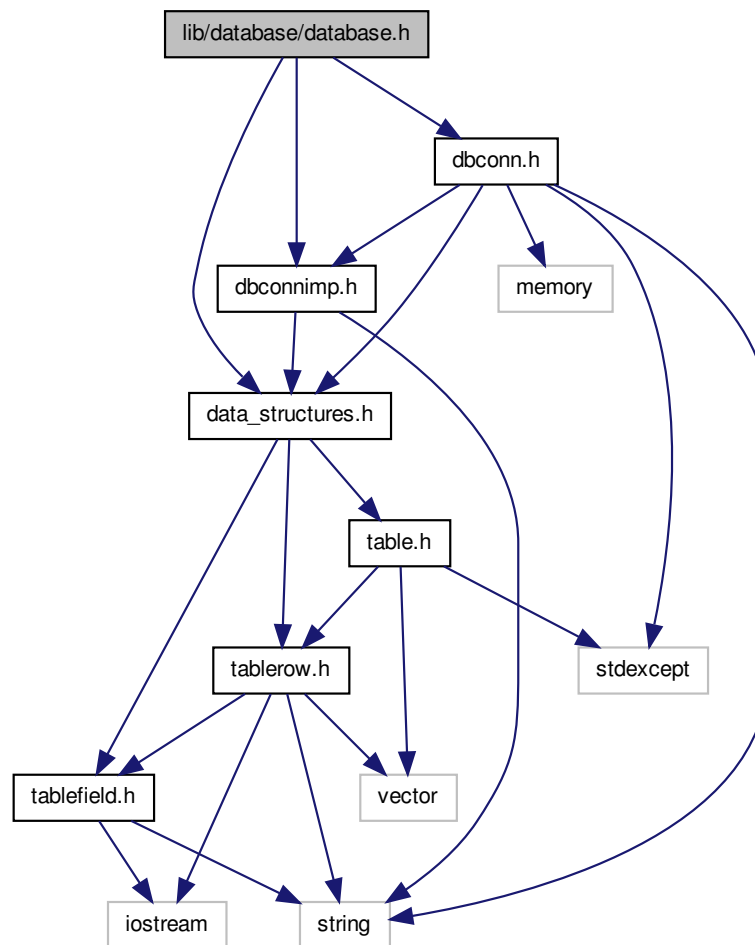
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.5 lib/database/database.h File Reference

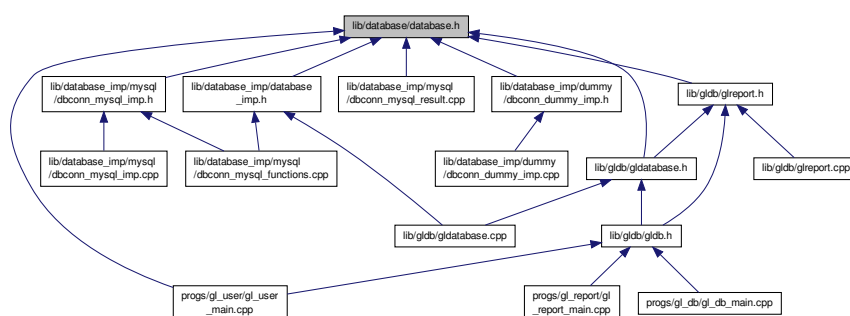
User interface to database functionality.

```
#include "data_structures.h"
#include "dbconnimp.h"
#include "dbconn.h"
```

Include dependency graph for database.h:



This graph shows which files directly or indirectly include this file:





### 10.5.1 Detailed Description

User interface to database functionality.

#### Author

Paul Griffiths

#### Copyright

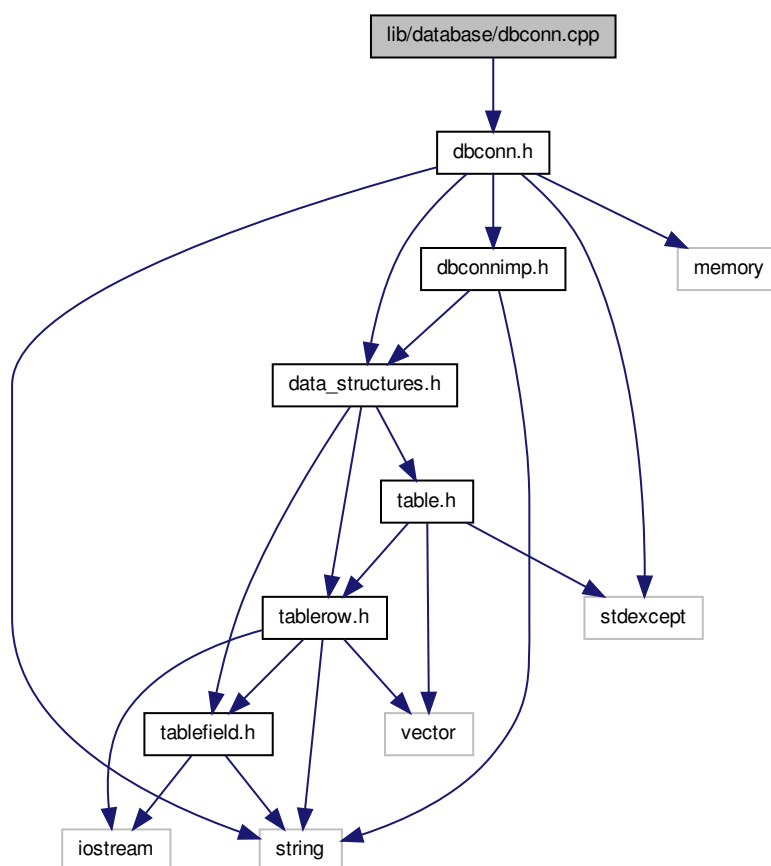
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.6 lib/database/dbconn.cpp File Reference

Implementation of database connection class.

```
#include "dbconn.h"
```

Include dependency graph for dbconn.cpp:



### 10.6.1 Detailed Description

Implementation of database connection class.

## Author

Paul Griffiths

## Copyright

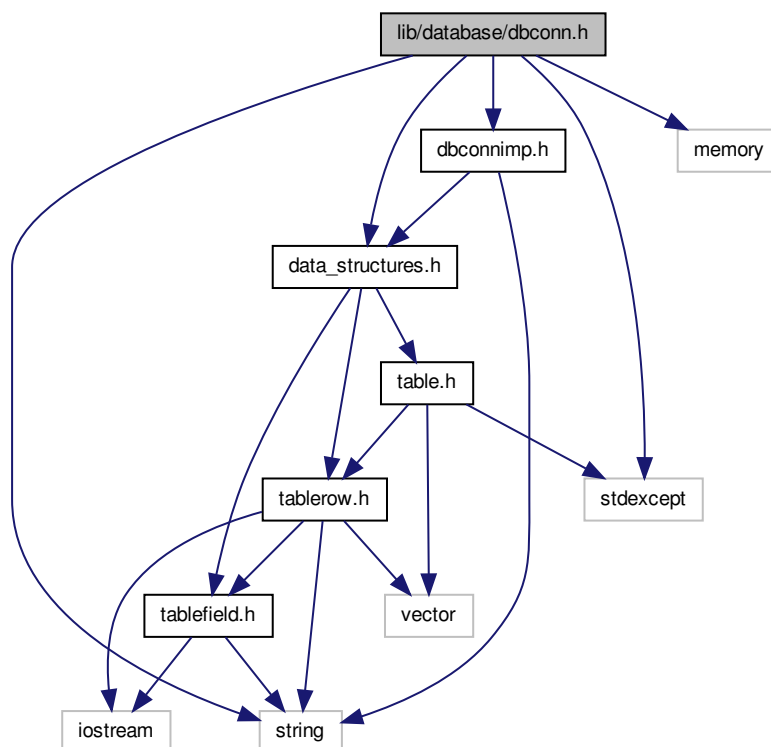
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.7 lib/database/dbconn.h File Reference

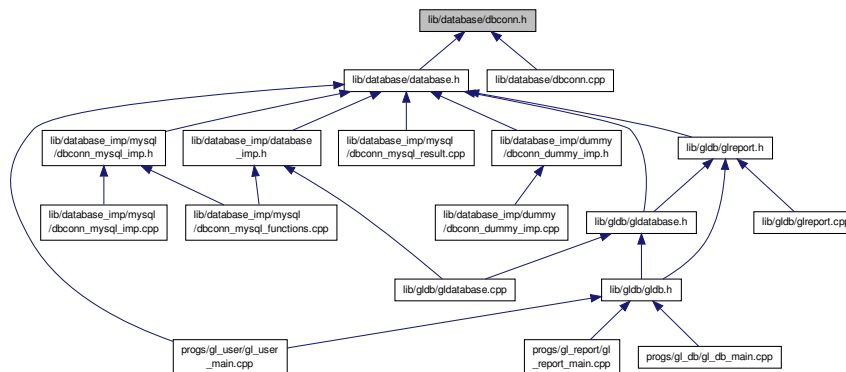
Interface to database connection base class.

```
#include <string>
#include <memory>
#include <stdexcept>
#include "data_structures.h"
#include "dbconnimp.h"
```

Include dependency graph for dbconn.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [gldb::DBConnException](#)  
*Base database connection exception class.*
- class [gldb::DBConnCouldNotConnect](#)  
*Could not connect to database exception class.*
- class [gldb::DBConnCouldNotQuery](#)  
*Could not execute database query exception class.*
- class [gldb::DBConn](#)  
*Database connection class.*

### 10.7.1 Detailed Description

Interface to database connection base class.

#### Author

Paul Griffiths

#### Copyright

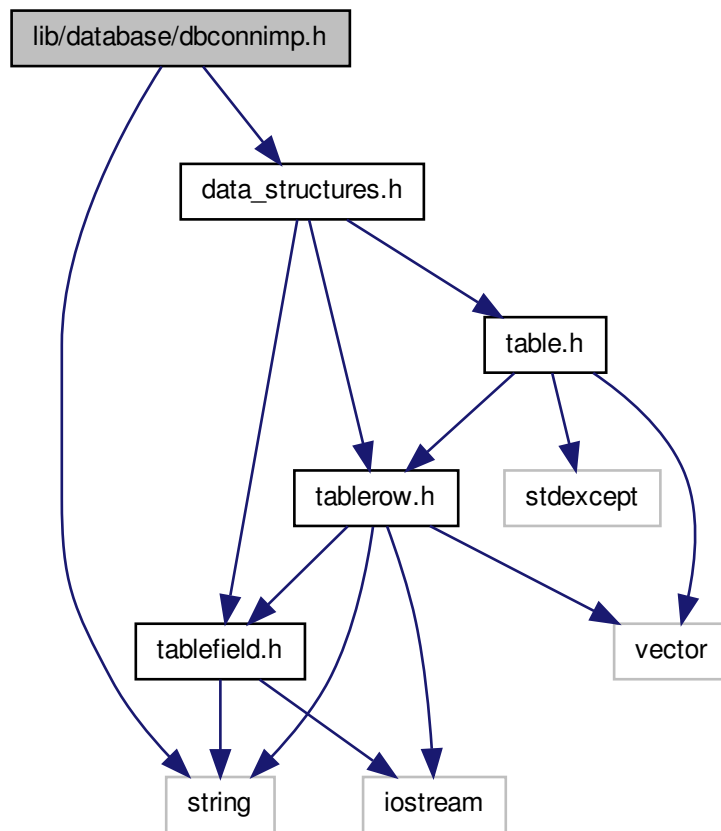
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.8 lib/database/dbconnimp.h File Reference

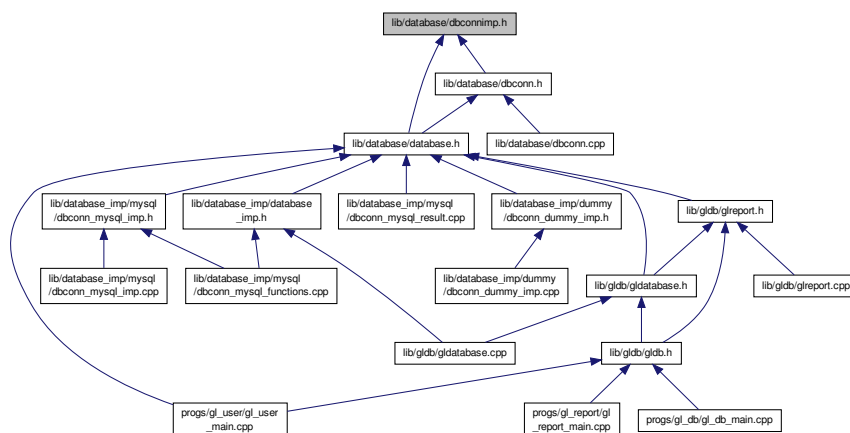
Interface to abstract database implementation base class.

```
#include <string>
#include "data_structures.h"
```

Include dependency graph for dbconnimp.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [gldb::DBConnImp](#)  
*Abstract database implementation base class.*

### 10.8.1 Detailed Description

Interface to abstract database implementation base class.

#### Author

Paul Griffiths

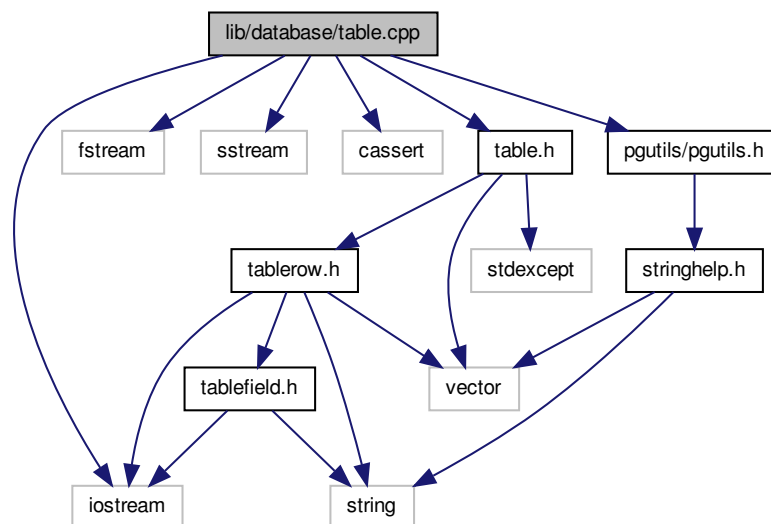
#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.9 lib/database/table.cpp File Reference

Implementation of database table data structure.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <cassert>
#include "table.h"
#include "pgutils/pgutils.h"
Include dependency graph for table.cpp:
```



### 10.9.1 Detailed Description

Implementation of database table data structure.

## Author

Paul Griffiths

## Copyright

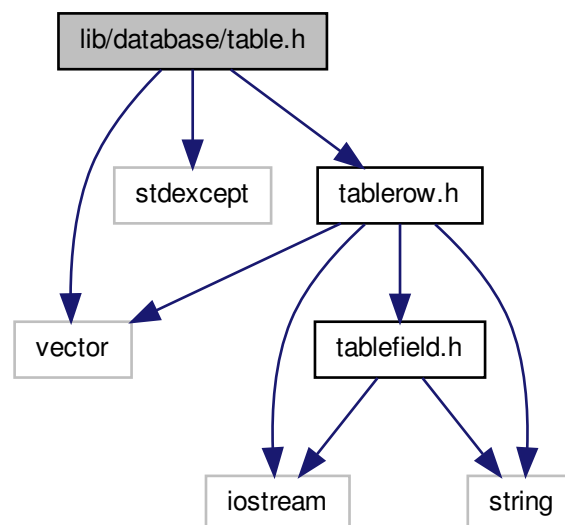
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.10 lib/database/table.h File Reference

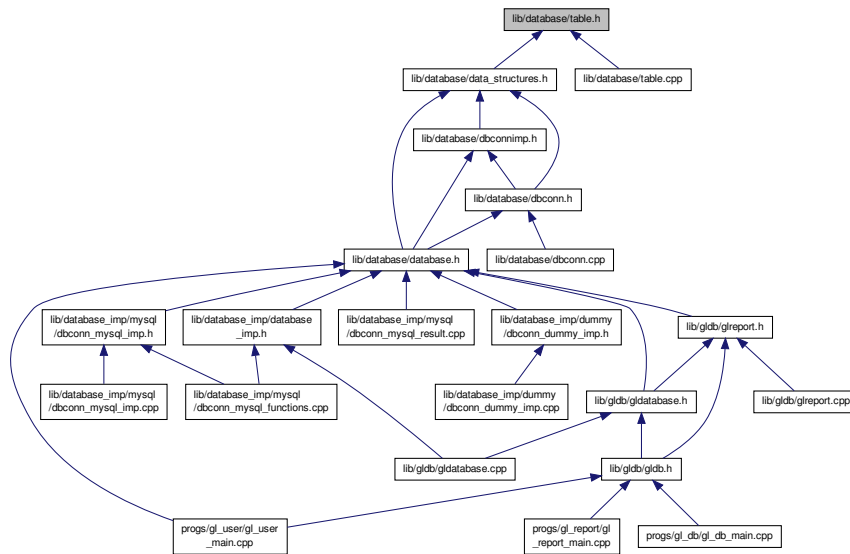
Interface to database table data structure.

```
#include <vector>
#include <stdexcept>
#include "tablerow.h"
```

Include dependency graph for table.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `gldb::TableException`  
*Base database connection exception class.*
- class `gldb::TableNoSuchField`  
*No such field exception class.*
- class `gldb::TableNoSuchRecord`  
*No such record exception class.*
- class `gldb::TableMismatchedRecordLength`  
*Mismatched record length exception class.*
- class `gldb::TableBadInputFile`  
*Could not connect to database exception class.*
- class `gldb::TableCouldNotOpenInputFile`  
*Could not connect to database exception class.*
- class `gldb::Table`  
*Database table class.*

### 10.10.1 Detailed Description

Interface to database table data structure.

**Author**

Paul Griffiths

**Copyright**

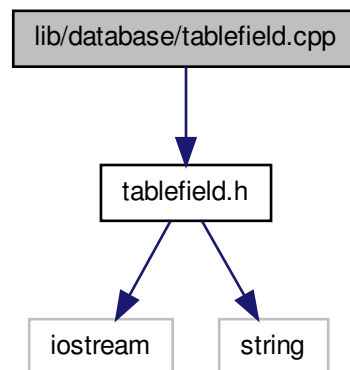
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.11 lib/database/tablefield.cpp File Reference

Implementation of database table field class.

```
#include "tablefield.h"
```

Include dependency graph for tablefield.cpp:



### 10.11.1 Detailed Description

Implementation of database table field class.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

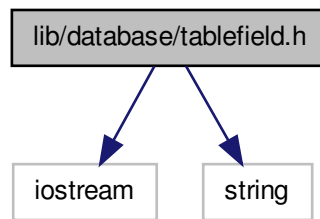
## 10.12 lib/database/tablefield.h File Reference

Interface to database table field class.

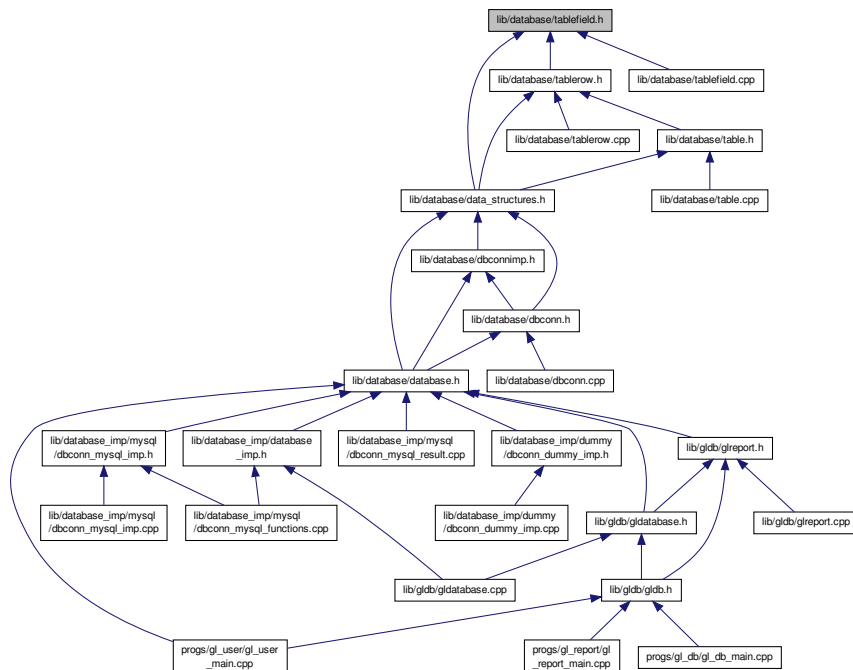
```
#include <iostream>
#include <string>
```



Include dependency graph for tablefield.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [gldb::TableField](#)  
*Database table field class.*

## Functions

- `std::ostream & gldb::operator<< (std::ostream &out, const TableField &field)`  
*Overridden << operator for printing a field.*

### 10.12.1 Detailed Description

Interface to database table field class.

**Author**

Paul Griffiths

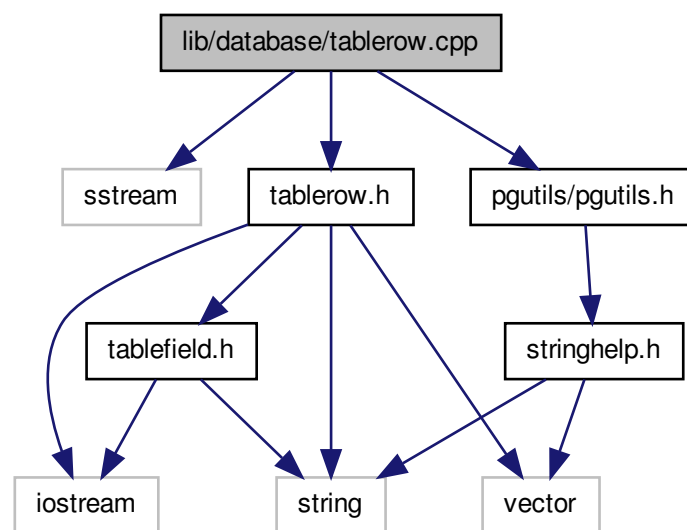
**Copyright**

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.13 lib/database/tablerow.cpp File Reference

Implementation of database table row data structure.

```
#include <sstream>
#include "tablerow.h"
#include "pgutils/pgutils.h"
Include dependency graph for tablerow.cpp:
```



### 10.13.1 Detailed Description

Implementation of database table row data structure.

**Author**

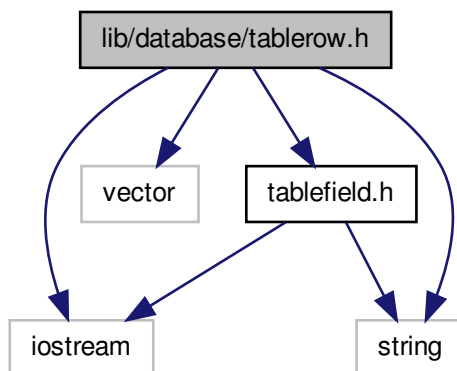
Paul Griffiths

**Copyright**

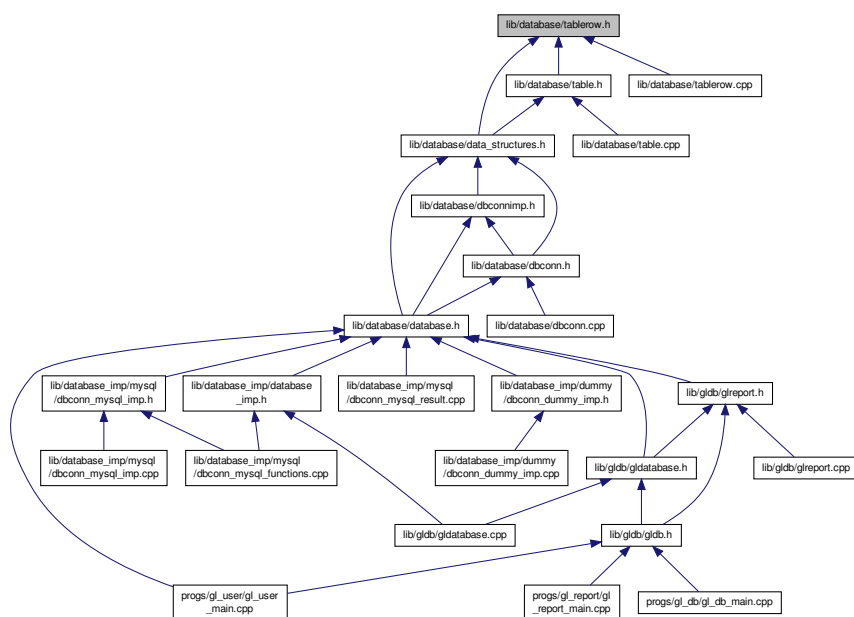
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

Interface to database table row data structure.

```
#include <iostream>
#include <vector>
#include <string>
#include "tablefield.h"
Include dependency graph for tablerow.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class `gldb::TableRow`

*Database table row class.*

### 10.14.1 Detailed Description

Interface to database table row data structure.

#### Author

Paul Griffiths

#### Copyright

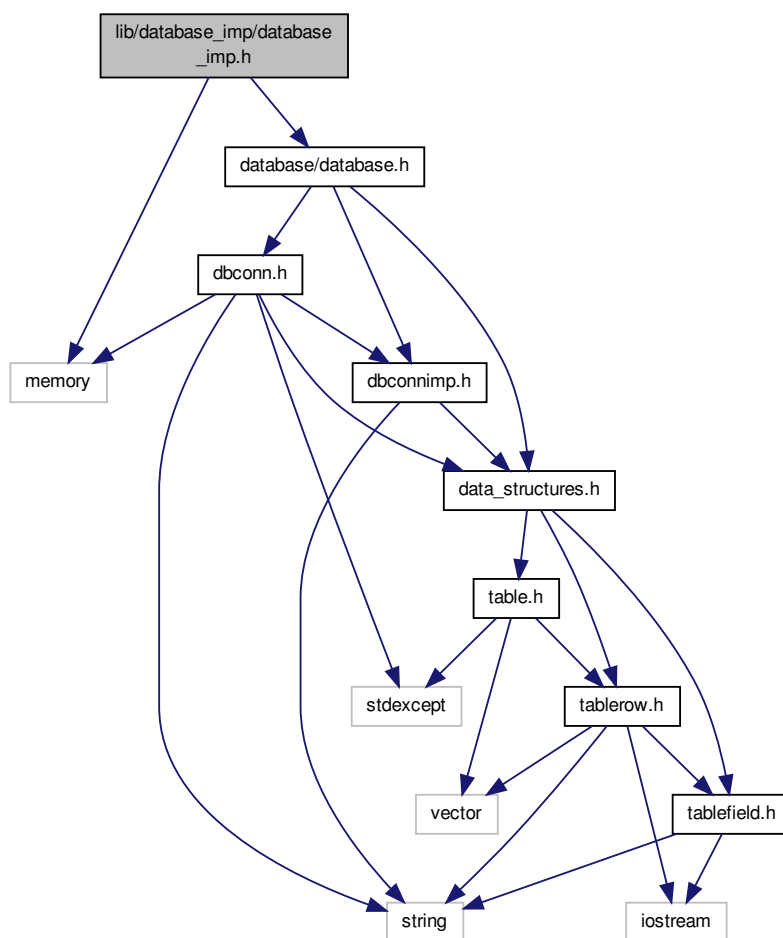
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.15 lib/database\_imp/database\_imp.h File Reference

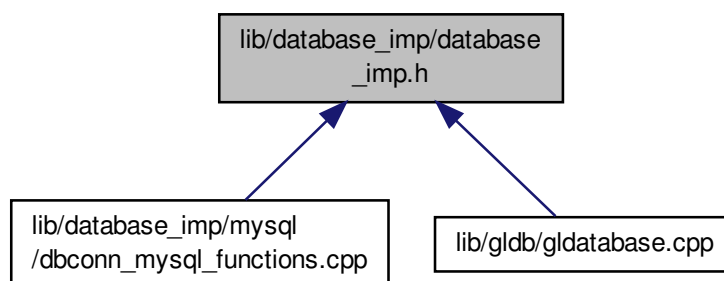
Interface to database implementation factory function.

```
#include <memory>
#include "database/database.h"
```

Include dependency graph for database\_imp.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `DBConnImp * glldb::get_connection` (const std::string &database, const std::string &hostname, const std::string &username, const std::string &password)

*Creates and returns a pointer to a database implementation.*

- `std::string glldb::get_database_type` ()

*Returns the name of the compiled-in database type.*

### 10.15.1 Detailed Description

Interface to database implementation factory function.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.16 lib/database\_imp/dummy/dbconn\_dummy\_imp.cpp File Reference

Implementation of Dummy database connection implementation class.

```
#include <sstream>
#include "dbconn_dummy_imp.h"
```

```
graph TD; Root["lib/database_imp/dummy/dbconn_dummy_imp.cpp"] --> sstream; Root --> dbconn_dummy_imp_h["dbconn_dummy_imp.h"]; dbconn_dummy_imp_h --> database_database_h["database/database.h"]; database_database_h --> dbconn_h["dbconn.h"]; database_database_h --> dbconnimp_h["dbconnimp.h"]; database_database_h --> data_structures_h["data_structures.h"]; dbconn_h --> memory; dbconn_h --> stdexcept; dbconn_h --> string; dbconnimp_h --> data_structures_h; data_structures_h --> table_h["table.h"]; data_structures_h --> tablerow_h["tablerow.h"]; data_structures_h --> tablefield_h["tablefield.h"]; table_h --> vector; table_h --> string; tablerow_h --> vector; tablerow_h --> string; tablefield_h --> vector; tablefield_h --> string; string --> iostream;
```

### Implementation of Dummy database connection implementation class.

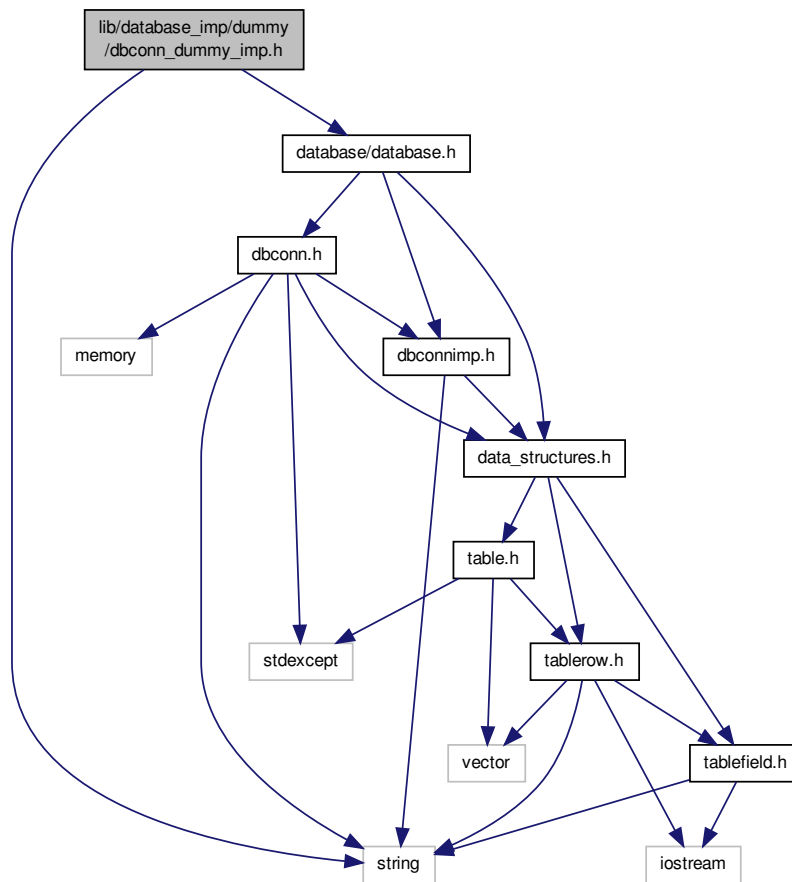
Paul Griffiths

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

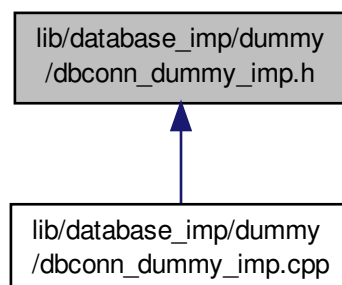
Interface to dummy database connection implementation class.

```
#include <string>
#include "database/database.h"
```

Include dependency graph for `dbconn_dummy_imp.h`:



This graph shows which files directly or indirectly include this file:





## Classes

- class `gldb::DBConnDummy`

*Dummy database implementation class.*

### 10.17.1 Detailed Description

Interface to dummy database connection implementation class.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

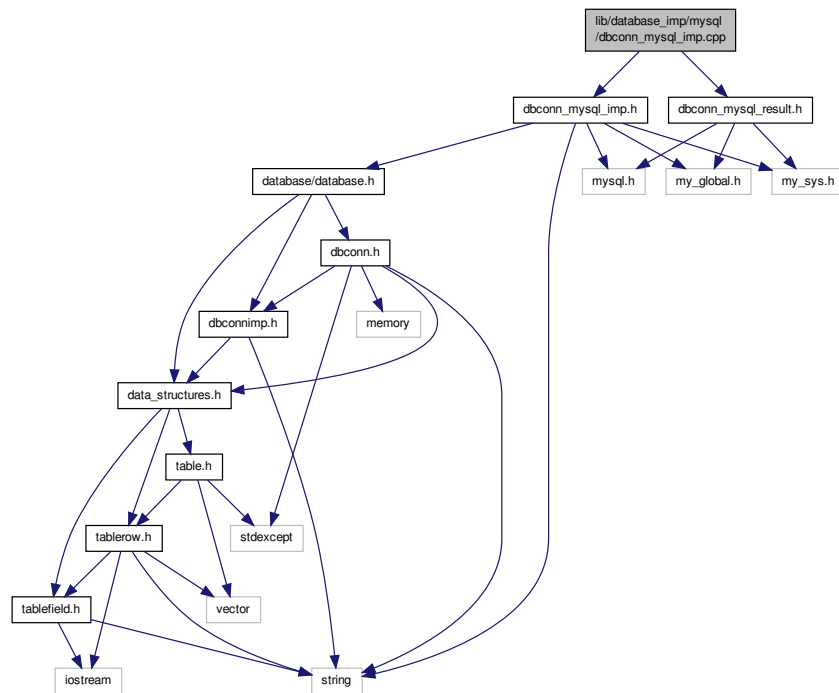
## 10.18 lib/database\_imp/mysql/dbconn\_mysql\_functions.cpp File Reference

Implementation of MySQL implementation factory function.

```
#include "../database_imp.h"
#include "dbconn_mysql_imp.h"
```



```
#include "dbconn_mysql_imp.h"
#include "dbconn_mysql_result.h"
Include dependency graph for dbconn_mysql_imp.cpp:
```



## Functions

- static [TableRow get\\_field\\_names](#) ([MySQLResult](#) &result)  
*Gets field names from a MySQL result structure.*
- static [TableRow get\\_row](#) ([MySQLResult](#) &result, [MYSQL\\_ROW](#) row)  
*Creates a TableRow from a MySQL result row.*

### 10.19.1 Detailed Description

Implementation of MySQL database connection implementation class.

#### Author

Paul Griffiths

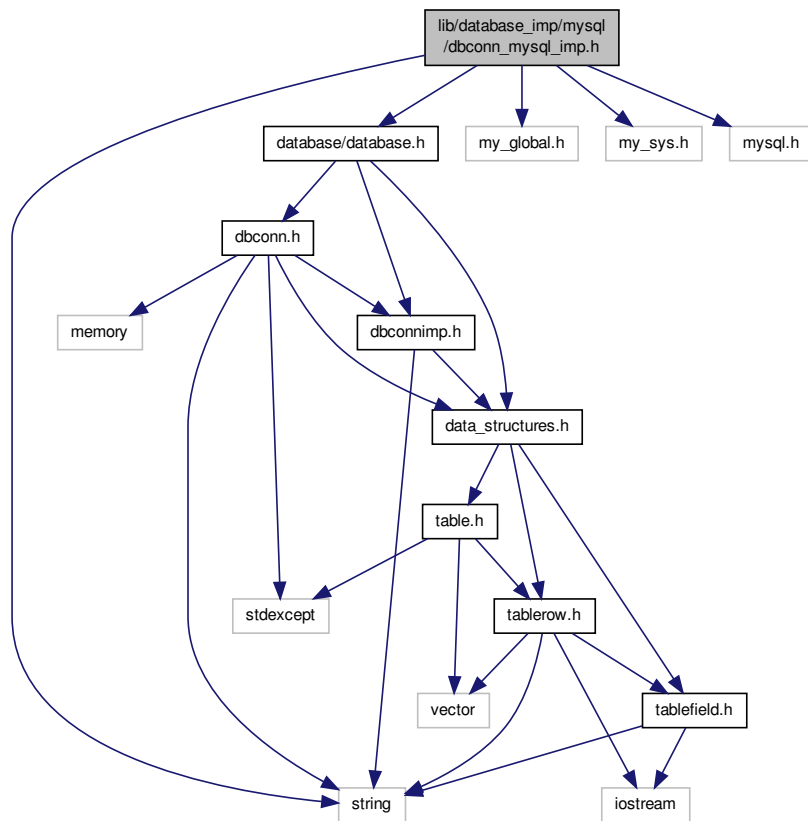
#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

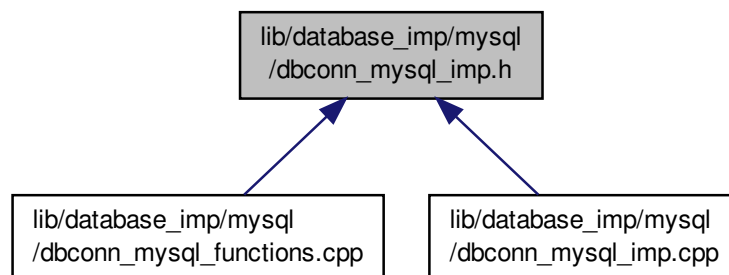
## 10.20 lib/database\_imp/mysql/dbconn\_mysql\_imp.h File Reference

Interface to MySQL database connection implementation class.

```
#include <string>
#include "database/database.h"
#include <my_global.h>
#include <my_sys.h>
#include <mysql.h>
Include dependency graph for dbconn_mysql_imp.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class [gldb::DBConnMySQL](#)

*MySQL database implementation class.*

### 10.20.1 Detailed Description

Interface to MySQL database connection implementation class.

#### Author

Paul Griffiths

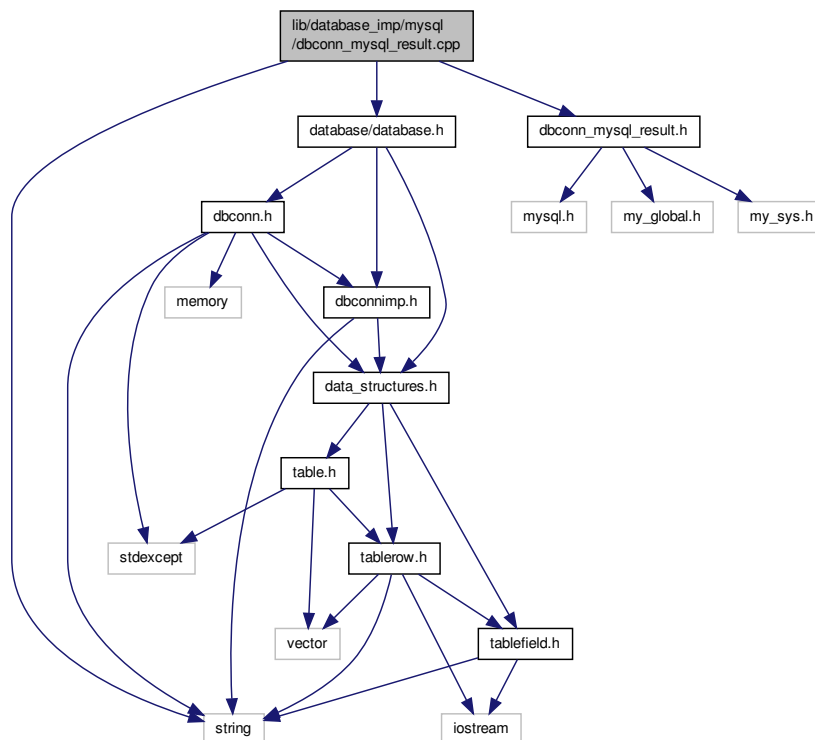
#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.21 lib/database\_imp/mysql/dbconn\_mysql\_result.cpp File Reference

Implementation of MySQL result structure resource handle class.

```
#include <string>
#include "database/database.h"
#include "dbconn_mysql_result.h"
Include dependency graph for dbconn_mysql_result.cpp:
```



### 10.21.1 Detailed Description

Implementation of MySQL result structure resource handle class.

#### Author

Paul Griffiths

#### Copyright

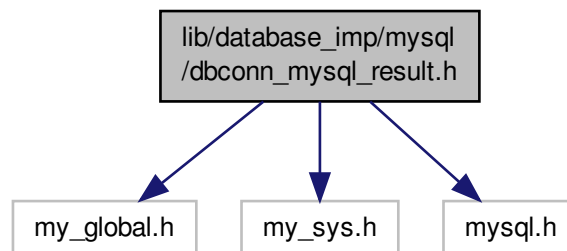
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.22 lib/database\_imp/mysql/dbconn\_mysql\_result.h File Reference

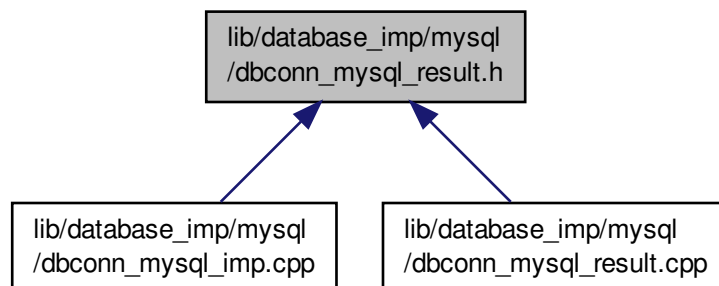
Interface to MySQL result structure resource handle class.

```
#include <my_global.h>
#include <my_sys.h>
#include <mysql.h>
```

Include dependency graph for dbconn\_mysql\_result.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `gldb::MySQLResult`  
*MySQL result structure class.*

### 10.22.1 Detailed Description

Interface to MySQL result structure resource handle class.

#### Author

Paul Griffiths

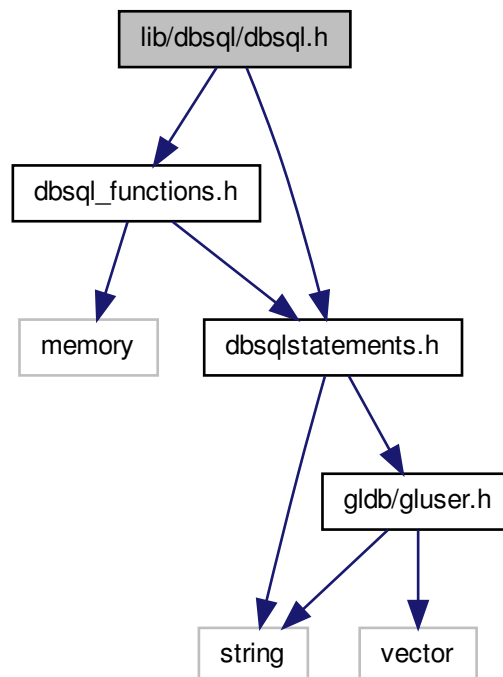
#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

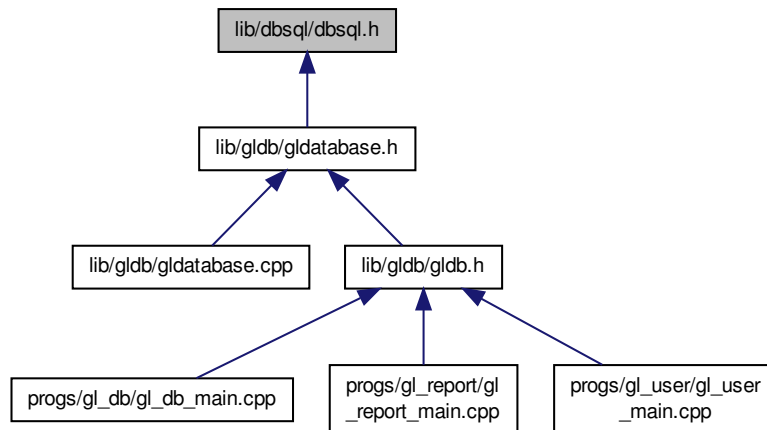
## 10.23 lib/dbsql/dbsql.h File Reference

User interface to DBSQL module.

```
#include "dbsql_functions.h"
#include "dbsqlstatements.h"
Include dependency graph for dbsql.h:
```



This graph shows which files directly or indirectly include this file:



### 10.23.1 Detailed Description

User interface to DBSQL module.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

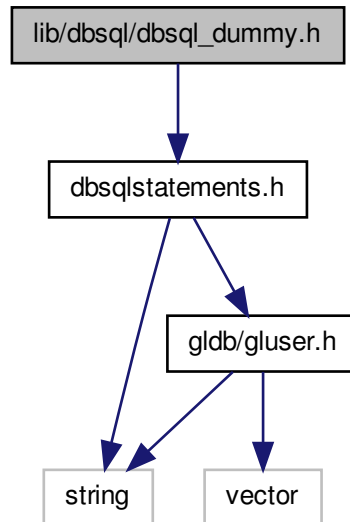
## 10.24 lib/dbsql/dbsql\_dummy.h File Reference

Interface to dummy SQL statement class.

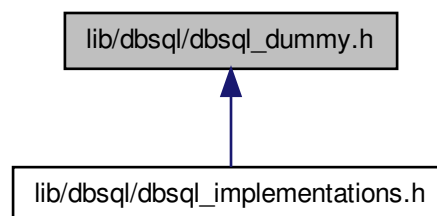


```
#include "dbsqlstatements.h"
```

Include dependency graph for `dbsql_dummy.h`:



This graph shows which files directly or indirectly include this file:



## Classes

- class [genleg::DBSQLDummy](#)  
*Dummy SQL statements class.*

### 10.24.1 Detailed Description

Interface to dummy SQL statement class.

**Author**

Paul Griffiths

**Copyright**

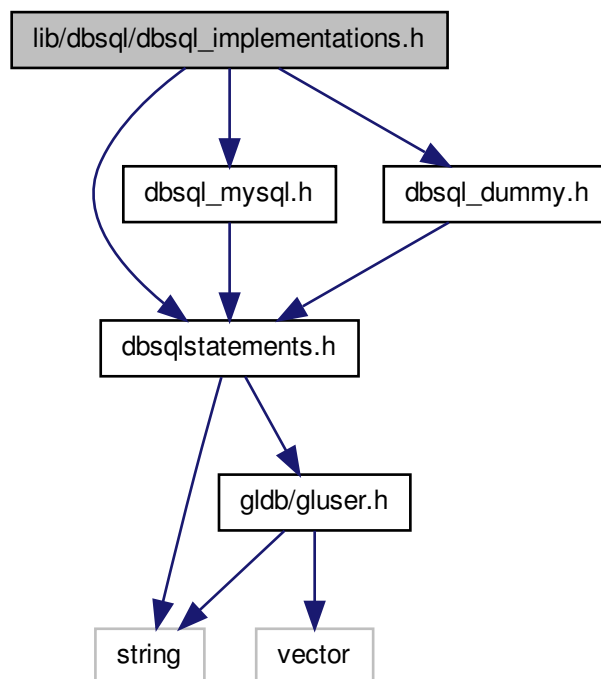
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.25 lib/dbsql/dbsql\_implementations.h File Reference

Aggregation header for DBSqlStatements implementations.

```
#include "dbsqlstatements.h"  
#include "dbsql_mysql.h"  
#include "dbsql_dummy.h"
```

Include dependency graph for dbsql\_implementations.h:



### 10.25.1 Detailed Description

Aggregation header for DBSqlStatements implementations.

**Author**

Paul Griffiths

### Copyright

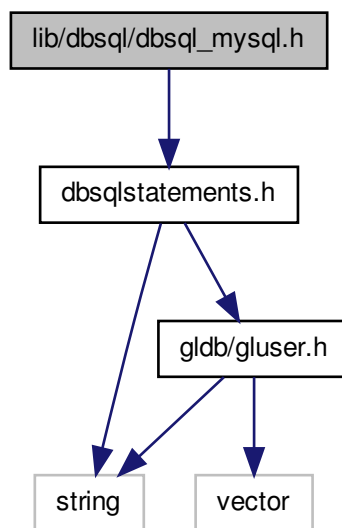
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.26 lib/dbsql/dbsql\_mysql.h File Reference

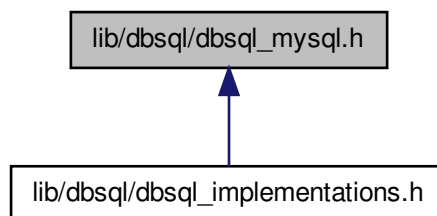
Interface to MySQL SQL statement class.

```
#include "dbsqlstatements.h"
```

Include dependency graph for dbsql\_mysql.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [genleg::DBSQLMySQL](#)

*MySQL SQL statements class.*

### 10.26.1 Detailed Description

Interface to MySQL SQL statement class.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

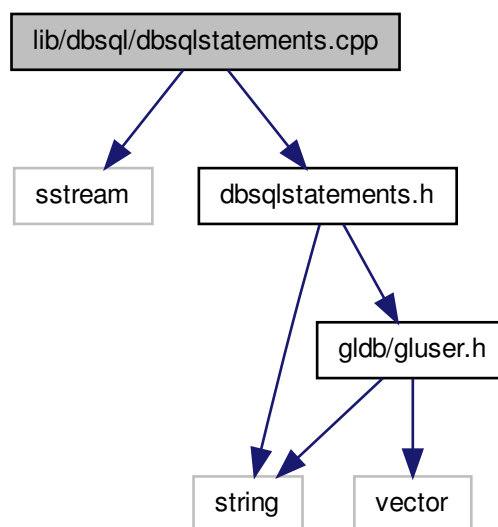
## 10.27 lib/dbsql/dbsqlstatements.cpp File Reference

Implementation of SQL statement class.

```
#include <sstream>
```

```
#include "dbsqlstatements.h"
```

Include dependency graph for dbsqlstatements.cpp:



### 10.27.1 Detailed Description

Implementation of SQL statement class.

#### Author

Paul Griffiths

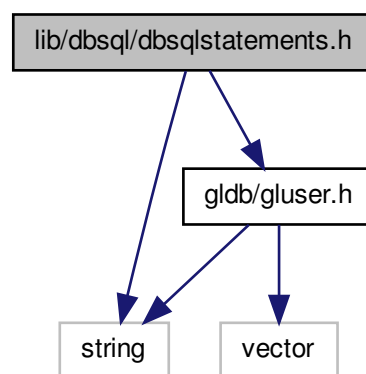
## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

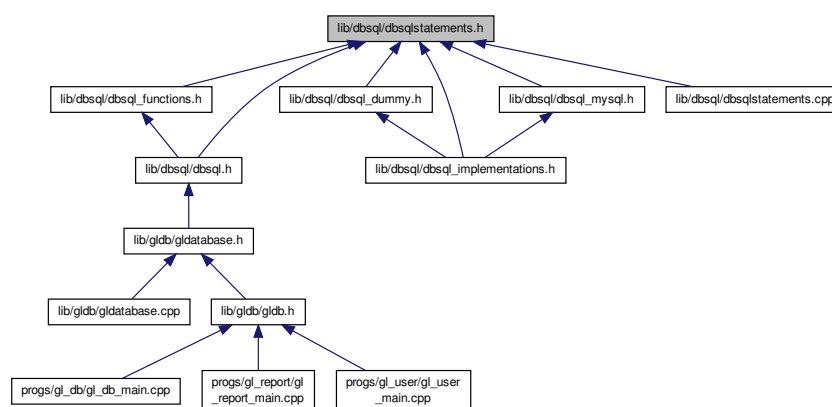
## 10.28 lib/dbsql/dbsqlstatements.h File Reference

Implementation of SQL module standalone functions.

```
#include <string>
#include "gldb/gluser.h"
Include dependency graph for dbsqlstatements.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class `genleg::DBSQLStatements`  
*SQL statements class.*



### 10.29.1 Detailed Description

Implementation of General Ledger database class.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 10.29.2 Function Documentation

10.29.2.1 `static bool boolstring_to_bool ( const std::string & bs ) [static]`

Converts a string representation of a bool to a bool.

#### Parameters

<i>bs</i>	The bool string.
-----------	------------------

#### Returns

true if *bs* contains "1" or "TRUE", false if *bs* contains "0" or "FALSE".

#### Exceptions

<i>GLDBException</i>	if <i>bs</i> contains any other value.
----------------------	--

## 10.30 lib/gldb/gldatabase.h File Reference

Interface to General Ledger database class.

```
#include <vector>
#include <string>
#include "database/database.h"
#include "dbsql/dbsql.h"
#include "gluser.h"
#include "glreport.h"
```

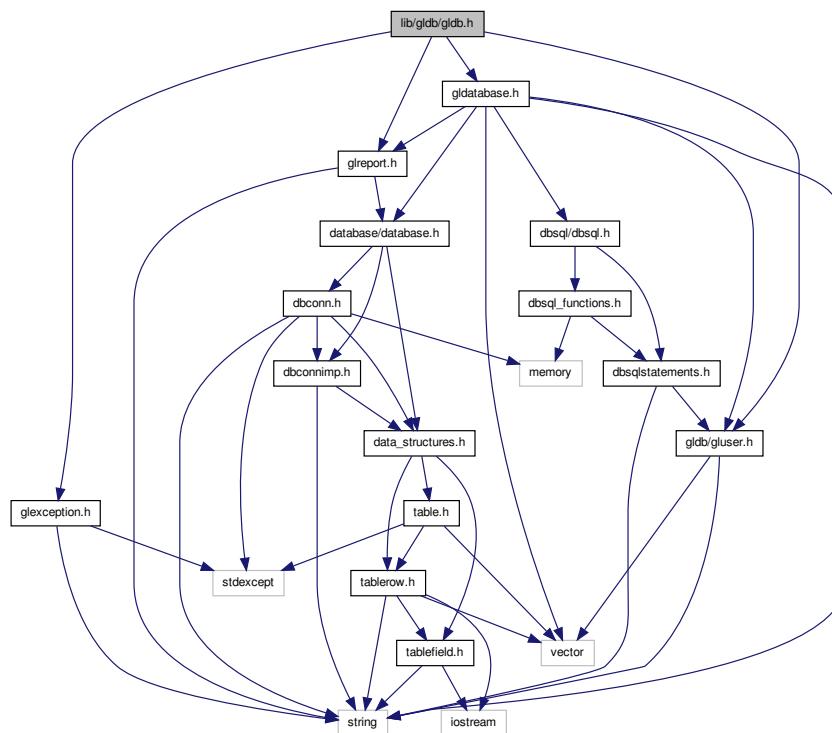




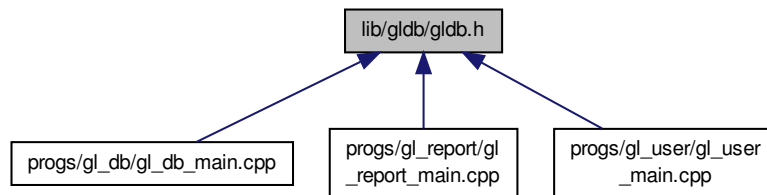
Paul Griffiths

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

Include dependency graph for glldb.h:



This graph shows which files directly or indirectly include this file:



### 10.31.1 Detailed Description

User interface to General Ledger database module.

#### Author

Paul Griffiths

#### Copyright

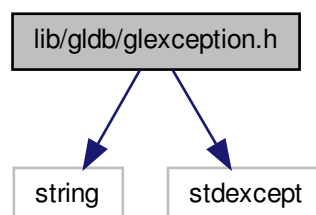
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.32 lib/gldb/glexception.h File Reference

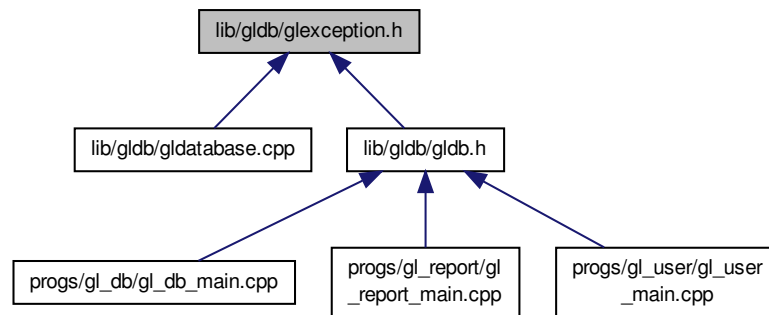
Interface to General Ledger base exception class.

```
#include <string>
#include <stdexcept>
```

Include dependency graph for glexception.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [genleg::GLDBException](#)

*Base general ledger database exceptionc class.*

### 10.32.1 Detailed Description

Interface to General Ledger base exception class.

#### Author

Paul Griffiths

#### Copyright

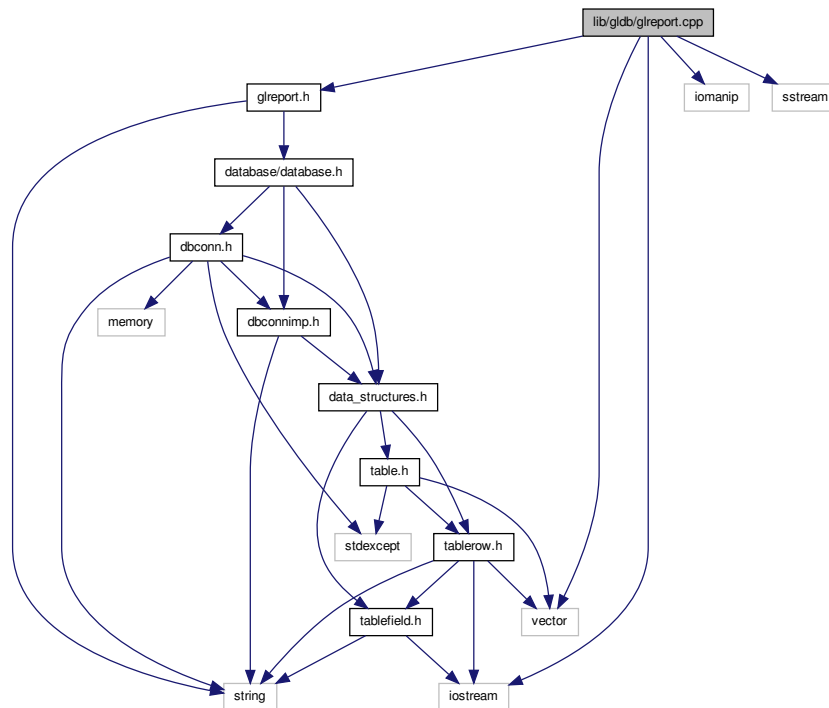
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.33 lib/gldb/glreport.cpp File Reference

Implementation of report class.

```
#include <vector>
#include <iomanip>
#include <iostream>
#include <sstream>
#include "glreport.h"
```

Include dependency graph for glreport.cpp:



## Functions

- static `std::vector< size_t > max_column_widths` (const `gldb::Table` &table)  
*Calculates the maximum required column widths for a table.*
- static void `grow_widths` (std::vector< size\_t > &widths, const `TableRow` &row)  
*Increments a vector of required column widths.*
- static `std::string separator_row` (const std::vector< size\_t > &widths)  
*Returns a decorated separator row for a table.*
- static `std::string plain_row` (const `TableRow` &row, const std::vector< size\_t > &widths)  
*Returns a row for a plain report.*
- static `std::string decorated_row` (const `TableRow` &row, const std::vector< size\_t > &widths)  
*Returns a row for a decorated report.*

### 10.33.1 Detailed Description

Implementation of report class.

#### Author

Paul Griffiths

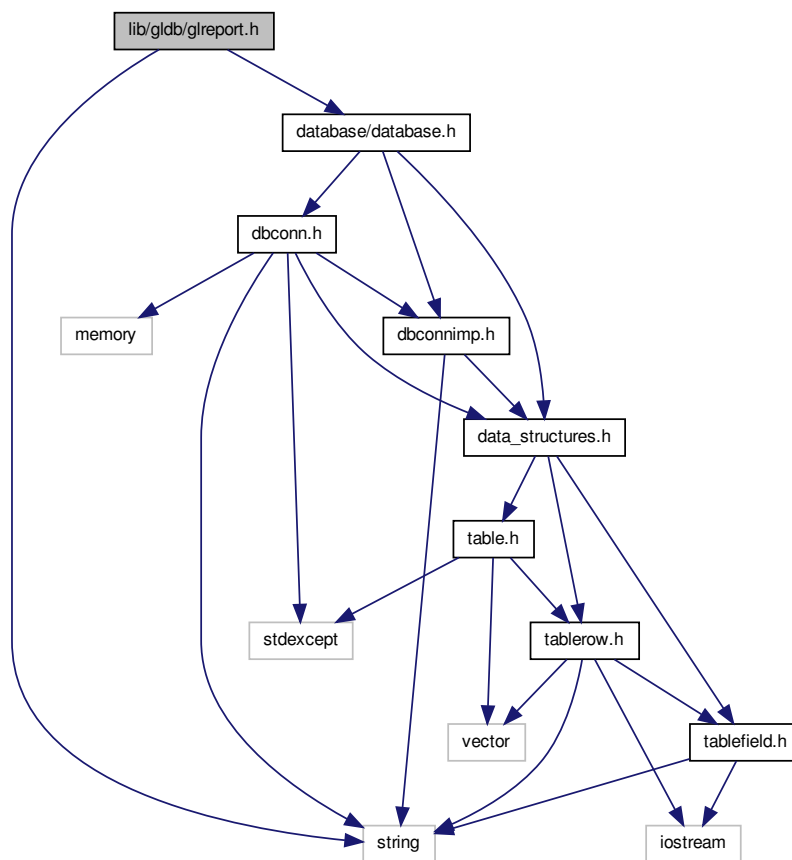
#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.34 lib/gldb/glreport.h File Reference

Interface to report class.

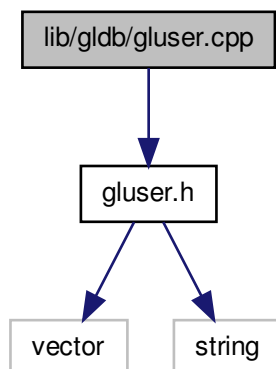
```
#include <string>
#include <database/database.h>
Include dependency graph for glreport.h:
```





```
#include "gluser.h"
```

Include dependency graph for gluser.cpp:



### 10.35.1 Detailed Description

Implementation of user class.

#### Author

Paul Griffiths

#### Copyright

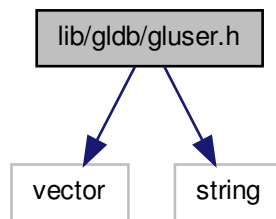
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.36 lib/gdb/gluser.h File Reference

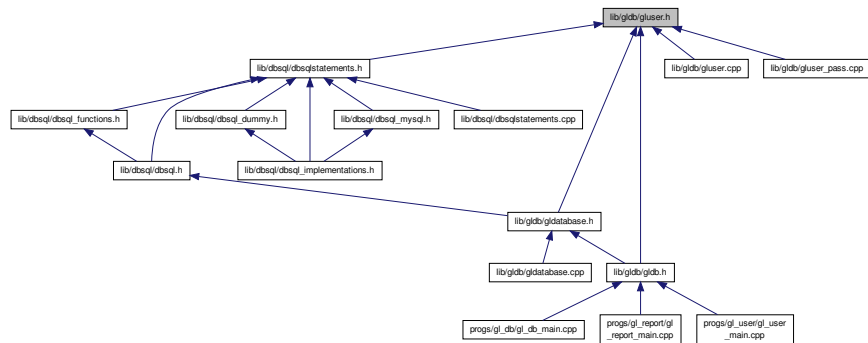
Interface to user class.

```
#include <vector>
#include <string>
```

Include dependency graph for gluser.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `genleg::GLUser`  
*General ledger user class.*

### 10.36.1 Detailed Description

Interface to user class.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

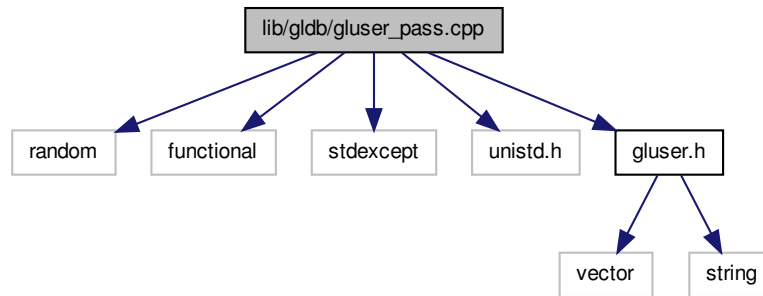
### 10.37 lib/gldb/gluser\_pass.cpp File Reference

Implementation of password functions for user class.



```
#include <random>
#include <functional>
#include <stdexcept>
#include <unistd.h>
#include "gluser.h"
```

Include dependency graph for gluser\_pass.cpp:



## Macros

- `#define _XOPEN_SOURCE 600`

## Functions

- static `std::string generate_salt ()`  
*Generates a random two-character salt for crypt()*

### 10.37.1 Detailed Description

Implementation of password functions for user class.

**Todo** Implement a better form of password encryption. In particular, these functions are not re-entrant, and only use the first 8 characters of the password.

## Author

Paul Griffiths

## Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

### 10.37.2 Macro Definition Documentation

#### 10.37.2.1 `#define _XOPEN_SOURCE 600`

UNIX feature test macro

### 10.37.3 Function Documentation

#### 10.37.3.1 `static std::string generate_salt ( ) [static]`

Generates a random two-character salt for crypt()

#### Returns

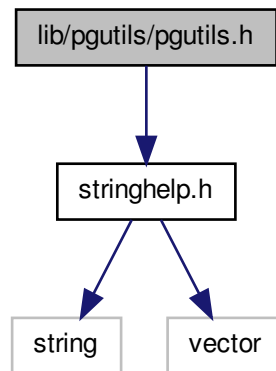
The two-character salt.

## 10.38 lib/pgutils/pgutils.h File Reference

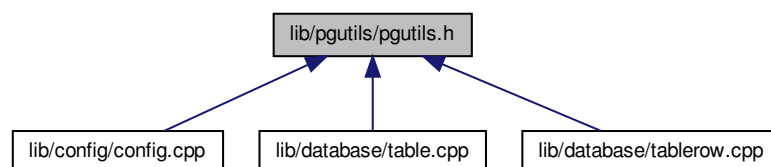
Aggregate interface to general utility functions.

```
#include "stringhelp.h"
```

Include dependency graph for pgutils.h:



This graph shows which files directly or indirectly include this file:



### 10.38.1 Detailed Description

Aggregate interface to general utility functions.

**Author**

Paul Griffiths

**Copyright**

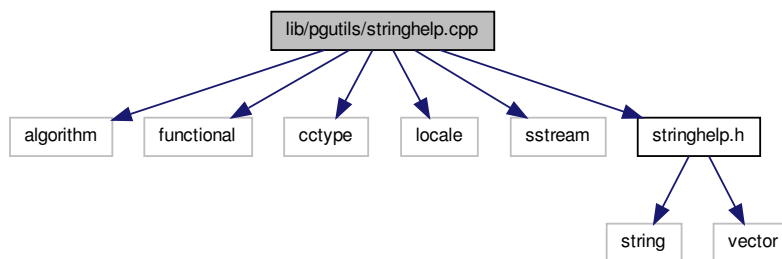
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.39 lib/pgutils/stringhelp.cpp File Reference

Implementation of string helper functions.

```
#include <algorithm>
#include <functional>
#include <cctype>
#include <locale>
#include <sstream>
#include "stringhelp.h"
```

Include dependency graph for stringhelp.cpp:



### 10.39.1 Detailed Description

Implementation of string helper functions.

**Author**

Paul Griffiths

**Copyright**

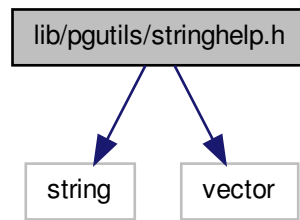
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.40 lib/pgutils/stringhelp.h File Reference

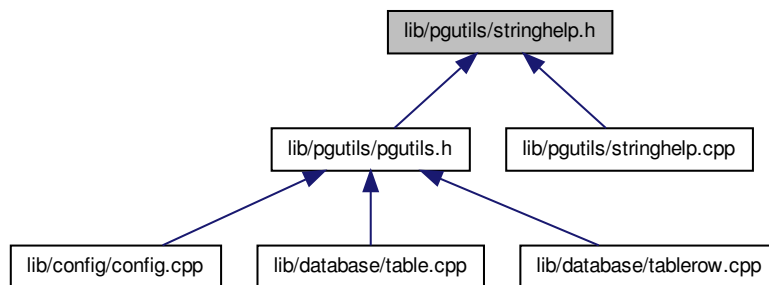
Interface to string helper functions.

```
#include <string>
#include <vector>
```

Include dependency graph for stringhelp.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `std::string & pgutils::trim_front (std::string &s)`  
*Trims leading whitespace from a string.*
- `std::string & pgutils::trim_back (std::string &s)`  
*Trims trailing whitespace from a string.*
- `std::string & pgutils::trim (std::string &s)`  
*Trims leading and trailing whitespace from a string.*
- `std::vector< std::string > pgutils::split (const std::string &s, const char delim)`  
*Splits a delimited string into tokens.*
- `std::vector< std::string > & pgutils::split (std::vector< std::string > &vec, const std::string &s, const char delim)`  
*Splits a delimited string into tokens.*
- `bool pgutils::next_content_line (std::istream &if, std::string &s)`  
*Gets the next content line from a stream.*
- `std::vector< std::string > & pgutils::content_lines (std::vector< std::string > &vec, std::istream &if)`  
*Populates a vector of content lines from a stream.*
- `std::vector< std::vector< std::string > > & pgutils::split_lines (std::vector< std::vector< std::string > > &vec, std::istream &if, const char delim)`

*Populates a vector of vectors of fields from a stream.*

- `std::string & pgutils::join` (`const std::vector< std::string > &vec`, `std::string &s`, `const char delim`)

*Joins a vector of strings into a delimited line.*

- `bool pgutils::replace` (`std::string &str`, `const std::string &from`, `const std::string &to`)

*Replaces a substring with another string.*

### 10.40.1 Detailed Description

Interface to string helper functions.

#### Author

Paul Griffiths

#### Copyright

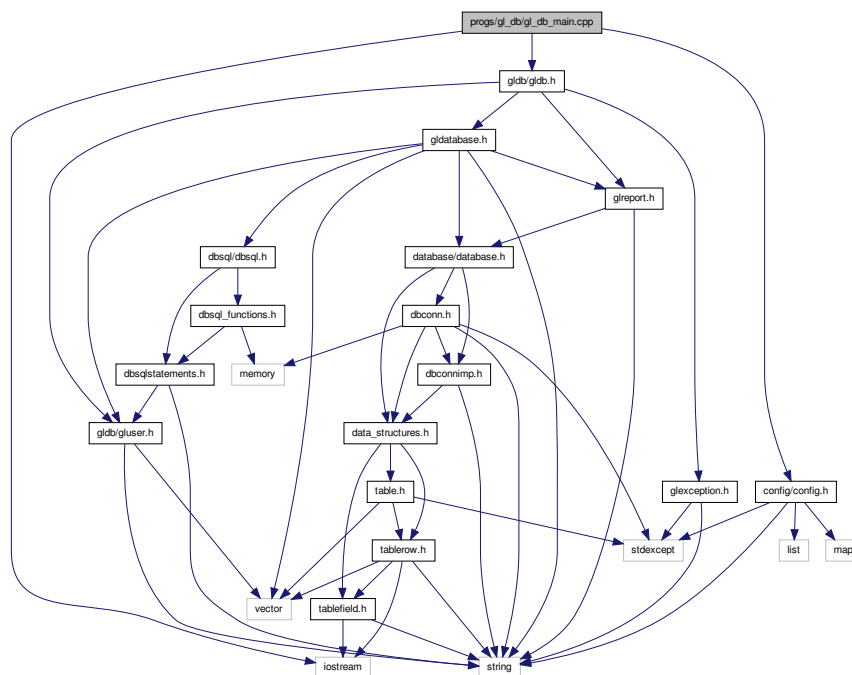
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.41 progs/gl\_db/gl\_db\_main.cpp File Reference

Main functionality for gl\_db program.

```
#include <iostream>
#include "gldb/gldb.h"
#include "config/config.h"
```

Include dependency graph for gl\_db\_main.cpp:



## Functions

- static void `set_configuration` (`Config` &config, int argc, char \*argv[])  
*Sets program configuration options.*
- static bool `check_help_and_version` (const `Config` &config)  
*Prints help or version messages if requested.*
- static bool `check_db_parameters` (const `Config` &config)  
*Checks if database, hostname and username were provided.*
- static void `print_usage_message` ()  
*Prints a program usage message.*
- static void `print_version_message` ()  
*Prints a program version message.*
- static void `print_help_message` ()  
*Prints a program help message.*
- static std::string `login` (void)  
*Gets a password from the terminal.*
- int `main` (int argc, char \*argv[])  
*Main function.*

## Variables

- static const char \* `progrname` = "gl\_db"  
*Static variable for program name.*

### 10.41.1 Detailed Description

Main functionality for gl\_db program.

#### Author

Paul Griffiths

#### Copyright

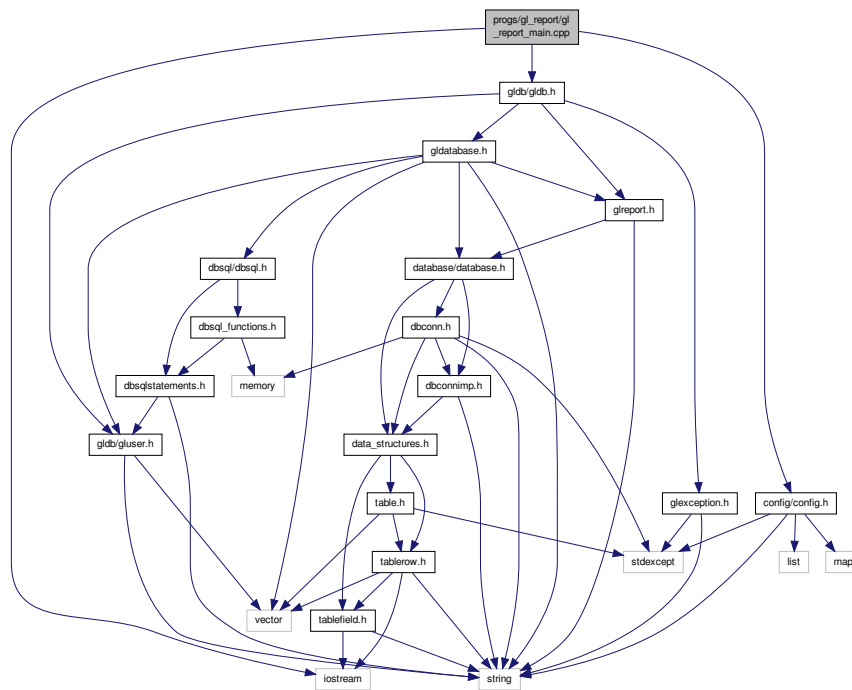
Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

## 10.42 progs/gl\_report/gl\_report\_main.cpp File Reference

Main functionality for gl\_report program.

```
#include <iostream>
#include "gldb/gldb.h"
#include "config/config.h"
```

Include dependency graph for gl\_report\_main.cpp:



## Functions

- static void `set_configuration` (`Config` &config, int argc, char \*argv[])  
*Sets program configuration options.*
- static bool `check_help_and_version` (const `Config` &config)  
*Prints help or version messages if requested.*
- static bool `check_db_parameters` (const `Config` &config)  
*Checks if database, hostname and username were provided.*
- static void `print_usage_message` ()  
*Prints a program usage message.*
- static void `print_version_message` ()  
*Prints a program version message.*
- static void `print_help_message` ()  
*Prints a program help message.*
- static std::string `login` (void)  
*Gets a password from the terminal.*
- int `main` (int argc, char \*argv[])  
*Main function.*

## Variables

- static const char \* `progrname` = "gl\_report"  
*Static variable for program name.*





- static bool `check_db_parameters` (const `Config` &config)  
*Checks if database, hostname and username were provided.*
- `GLUser` `get_user` (`Config` &config, `GLDatabase` &gdb)  
*Returns a user from either an ID or a name.*
- static void `show_user_details` (const `GLUser` &user)  
*Outputs details for a user.*
- static void `enable_user` (`GLUser` &user, `Config` &config, `GLDatabase` &gdb)  
*Enables or disables a user.*
- static void `set_user_password` (`GLUser` &user, `Config` &config, `GLDatabase` &gdb)  
*Sets a user's password.*
- static void `check_user_password` (`GLUser` &user, `Config` &config)  
*Checks a user's password.*
- static void `print_usage_message` ()  
*Prints a program usage message.*
- static void `print_version_message` ()  
*Prints a program version message.*
- static void `print_help_message` ()  
*Prints a program help message.*
- static std::string `login` (void)  
*Gets a password from the terminal.*
- int `main` (int argc, char \*argv[])  
*Main function.*

## Variables

- static const char \* `progrname` = "gl\_user"  
*Static variable for program name.*

### 10.43.1 Detailed Description

Main functionality for gl\_user program.

#### Author

Paul Griffiths

#### Copyright

Copyright 2014 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

# Index

- ~Config
  - genleg::Config, [35](#)
- ~DBConnDummy
  - gldb::DBConnDummy, [49](#)
- ~DBConnImp
  - gldb::DBConnImp, [51](#)
- ~DBConnMySQL
  - gldb::DBConnMySQL, [54](#)
- ~DBSQLStatements
  - genleg::DBSQLStatements, [58](#)
- ~GLDatabase
  - genleg::GLDatabase, [62](#)
- ~GLReport
  - genleg::GLReport, [67](#)
- ~GLUser
  - genleg::GLUser, [69](#)
- ~MySQLResult
  - gldb::MySQLResult, [73](#)
- ~Table
  - gldb::Table, [77](#)
- ~TableField
  - gldb::TableField, [86](#)
- ~TableRow
  - gldb::TableRow, [95](#)
- \_XOPEN\_SOURCE
  - config\_getopt.cpp, [102](#)
  - gluser\_pass.cpp, [145](#)
- add\_cmdline\_option
  - genleg::Config, [36](#)
- append\_field
  - gldb::TableRow, [95](#), [96](#)
- append\_record
  - gldb::Table, [77](#)
- backend
  - genleg::GLDatabase, [63](#)
- begin
  - gldb::Table, [77](#)
  - gldb::TableRow, [96](#)
- boolstring\_to\_bool
  - gldatabase.cpp, [135](#)
- check\_db\_parameters
  - Database program., [27](#)
  - Reporting program., [29](#)
  - User administration program., [31](#)
- check\_help\_and\_version
  - Database program., [27](#)
  - Reporting program., [29](#)
  - User administration program., [32](#)
- check\_password
  - genleg::GLUser, [69](#)
- check\_user\_password
  - User administration program., [32](#)
- Config
  - genleg::Config, [35](#)
- config\_getopt.cpp
  - \_XOPEN\_SOURCE, [102](#)
- ConfigBadConfigFile
  - genleg::ConfigBadConfigFile, [38](#)
- ConfigBadOption
  - genleg::ConfigBadOption, [39](#)
- ConfigCouldNotOpenFile
  - genleg::ConfigCouldNotOpenFile, [41](#)
- ConfigException
  - genleg::ConfigException, [42](#)
- ConfigOptionNotSet
  - genleg::ConfigOptionNotSet, [43](#)
- content\_lines
  - General purpose utilities., [23](#)
- create\_from\_file
  - gldb::Table, [77](#)
- create\_structure
  - genleg::GLDatabase, [63](#)
- create\_table
  - genleg::DBSQLStatements, [58](#)
- create\_user
  - genleg::GLDatabase, [63](#)
- create\_view
  - genleg::DBSQLStatements, [58](#)
- current\_trial\_balance\_report
  - genleg::GLDatabase, [63](#)
- currenttb
  - genleg::DBSQLStatements, [58](#)
- currenttb\_by\_entity
  - genleg::DBSQLStatements, [58](#)
- DBConn
  - gldb::DBConn, [44](#)
- DBConnCouldNotConnect
  - gldb::DBConnCouldNotConnect, [46](#)
- DBConnCouldNotQuery
  - gldb::DBConnCouldNotQuery, [47](#)
- DBConnDummy
  - gldb::DBConnDummy, [49](#)
- DBConnException
  - gldb::DBConnException, [50](#)
- DBConnImp
  - gldb::DBConnImp, [51](#)

- DBConnMySQL
  - gldb::DBConnMySQL, [53](#), [54](#)
- DBSQLStatements
  - genleg::DBSQLStatements, [58](#)
- Database interaction module, [16](#)
  - get\_connection, [17](#)
  - get\_database\_type, [17](#)
  - get\_field\_names, [17](#)
  - get\_row, [17](#)
- Database program., [27](#)
  - check\_db\_parameters, [27](#)
  - check\_help\_and\_version, [27](#)
  - login, [28](#)
  - main, [28](#)
  - set\_configuration, [28](#)
- decorated\_report\_from\_table
  - General Ledger database module., [20](#)
- decorated\_row
  - General Ledger database module., [20](#)
- destroy\_structure
  - genleg::GLDatabase, [63](#)
- drop\_table
  - genleg::DBSQLStatements, [59](#)
- drop\_view
  - genleg::DBSQLStatements, [59](#)
- enable\_user
  - User administration program., [32](#)
- enabled
  - genleg::GLUser, [70](#)
- end
  - gldb::Table, [78](#)
  - gldb::TableRow, [96](#)
- firstname
  - genleg::GLUser, [70](#)
- GLDBException
  - genleg::GLDBException, [66](#)
- GLDatabase
  - genleg::GLDatabase, [62](#)
- GLReport
  - genleg::GLReport, [67](#)
- GLUser
  - genleg::GLUser, [69](#)
- General Ledger database module., [20](#)
  - decorated\_report\_from\_table, [20](#)
  - decorated\_row, [20](#)
  - grow\_widths, [21](#)
  - max\_column\_widths, [21](#)
  - plain\_report\_from\_table, [21](#)
  - plain\_row, [21](#)
  - separator\_row, [22](#)
- General purpose utilities., [23](#)
  - content\_lines, [23](#)
  - join, [23](#)
  - next\_content\_line, [24](#)
  - replace, [24](#)
  - split, [24](#)
  - split\_lines, [25](#)
  - trim, [25](#)
  - trim\_back, [25](#)
  - trim\_front, [25](#)
- generate\_salt
  - gluser\_pass.cpp, [146](#)
- genleg::Config, [35](#)
  - ~Config, [35](#)
  - add\_cmdline\_option, [36](#)
  - Config, [35](#)
  - is\_set, [36](#)
  - m\_opts\_set, [37](#)
  - m\_opts\_supp, [37](#)
  - populate\_from\_cmdline, [36](#)
  - populate\_from\_file, [37](#)
- genleg::ConfigBadConfigFile, [37](#)
  - ConfigBadConfigFile, [38](#)
- genleg::ConfigBadOption, [39](#)
  - ConfigBadOption, [39](#)
- genleg::ConfigCouldNotOpenFile, [40](#)
  - ConfigCouldNotOpenFile, [41](#)
- genleg::ConfigException, [41](#)
  - ConfigException, [42](#)
- genleg::ConfigOptionNotSet, [42](#)
  - ConfigOptionNotSet, [43](#)
- genleg::DBSQLDummy, [55](#)
- genleg::DBSQLMySQL, [56](#)
- genleg::DBSQLStatements, [57](#)
  - ~DBSQLStatements, [58](#)
  - create\_table, [58](#)
  - create\_view, [58](#)
  - currenttb, [58](#)
  - currenttb\_by\_entity, [58](#)
  - DBSQLStatements, [58](#)
  - drop\_table, [59](#)
  - drop\_view, [59](#)
  - get\_perms, [59](#)
  - grant, [59](#)
  - revoke, [60](#)
  - update\_user, [60](#)
  - user\_by\_id, [60](#)
  - user\_by\_username, [60](#)
- genleg::GLDBException, [66](#)
  - GLDBException, [66](#)
- genleg::GLDatabase, [61](#)
  - ~GLDatabase, [62](#)
  - backend, [63](#)
  - create\_structure, [63](#)
  - create\_user, [63](#)
  - current\_trial\_balance\_report, [63](#)
  - destroy\_structure, [63](#)
  - GLDatabase, [62](#)
  - get\_user\_by\_id, [64](#)
  - get\_user\_by\_username, [64](#)
  - grant, [64](#)
  - load\_sample\_data, [64](#)
  - m\_dbc, [65](#)
  - m\_sql, [65](#)

- m\_tables, 65
  - m\_views, 65
  - report, 65
  - revoke, 65
  - update\_user, 65
- genleg::GLReport, 66
  - ~GLReport, 67
  - GLReport, 67
  - m\_report\_text, 67
  - operator<<, 67
- genleg::GLUser, 67
  - ~GLUser, 69
  - check\_password, 69
  - enabled, 70
  - firstname, 70
  - GLUser, 69
  - id, 70
  - lastname, 70
  - m\_enabled, 72
  - m\_firstname, 72
  - m\_id, 72
  - m\_lastname, 72
  - m\_pass\_hash, 72
  - m\_pass\_salt, 72
  - m\_perms, 72
  - m\_username, 72
  - pass\_hash, 70
  - pass\_salt, 70
  - permissions, 70
  - set\_enabled, 71
  - set\_firstname, 71
  - set\_lastname, 71
  - set\_password, 71
  - set\_username, 71
  - username, 71
- get\_connection
  - Database interaction module, 17
- get\_database\_type
  - Database interaction module, 17
- get\_field
  - gldb::Table, 78
- get\_field\_names
  - Database interaction module, 17
- get\_headers
  - gldb::Table, 78
- get\_perms
  - genleg::DBSQLStatements, 59
- get\_row
  - Database interaction module, 17
- get\_user
  - User administration program., 32
- get\_user\_by\_id
  - genleg::GLDatabase, 64
- get\_user\_by\_username
  - genleg::GLDatabase, 64
- gldatabase.cpp
  - boolstring\_to\_bool, 135
- gldb::DBConn, 43
  - DBConn, 44
  - m\_imp, 45
  - operator=, 44
  - query, 44
  - select, 45
- gldb::DBConnCouldNotConnect, 45
  - DBConnCouldNotConnect, 46
- gldb::DBConnCouldNotQuery, 46
  - DBConnCouldNotQuery, 47
- gldb::DBConnDummy, 48
  - ~DBConnDummy, 49
  - DBConnDummy, 49
  - operator=, 49
  - query, 49
  - select, 49
- gldb::DBConnException, 50
  - DBConnException, 50
- gldb::DBConnImp, 51
  - ~DBConnImp, 51
  - DBConnImp, 51
  - query, 52
  - select, 52
- gldb::DBConnMySQL, 52
  - ~DBConnMySQL, 54
  - DBConnMySQL, 53, 54
  - m\_conn, 55
  - operator=, 54
  - query, 54
  - select, 54
- gldb::MySQLResult, 72
  - ~MySQLResult, 73
  - m\_num\_fields, 74
  - m\_result, 74
  - MySQLResult, 73
  - num\_fields, 73
  - operator=, 74
  - result, 74
- gldb::Table, 74
  - ~Table, 77
  - append\_record, 77
  - begin, 77
  - create\_from\_file, 77
  - end, 78
  - get\_field, 78
  - get\_headers, 78
  - insert\_query, 79
  - m\_headers, 80
  - m\_quoted, 80
  - m\_records, 80
  - num\_fields, 79
  - num\_records, 79
  - operator=, 79
  - set\_quoted, 80
  - Table, 76, 77
- gldb::TableBadInputFile, 81
  - TableBadInputFile, 81
- gldb::TableCouldNotOpenInputFile, 82
  - TableCouldNotOpenInputFile, 83

- gldb::TableException, [83](#)
  - TableException, [84](#)
- gldb::TableField, [84](#)
  - ~TableField, [86](#)
  - length, [86](#)
  - m\_data, [89](#)
  - operator std::string, [86](#)
  - operator<<, [89](#)
  - operator+=, [86](#), [87](#)
  - operator=, [87](#), [88](#)
  - TableField, [85](#), [86](#)
- gldb::TableMismatchedRecordLength, [89](#)
  - TableMismatchedRecordLength, [90](#)
- gldb::TableNoSuchField, [90](#)
  - TableNoSuchField, [91](#)
- gldb::TableNoSuchRecord, [92](#)
  - TableNoSuchRecord, [92](#)
- gldb::TableRow, [93](#)
  - ~TableRow, [95](#)
  - append\_field, [95](#), [96](#)
  - begin, [96](#)
  - end, [96](#)
  - m\_fields, [98](#)
  - operator=, [97](#)
  - print, [97](#)
  - record\_string, [98](#)
  - size, [98](#)
  - TableRow, [94](#), [95](#)
- gluser\_pass.cpp
  - \_XOPEN\_SOURCE, [145](#)
  - generate\_salt, [146](#)
- grant
  - genleg::DBSQLStatements, [59](#)
  - genleg::GLDatabase, [64](#)
- grow\_widths
  - General Ledger database module., [21](#)
- id
  - genleg::GLUser, [70](#)
- insert\_query
  - gldb::Table, [79](#)
- is\_set
  - genleg::Config, [36](#)
- join
  - General purpose utilities., [23](#)
- lastname
  - genleg::GLUser, [70](#)
- length
  - gldb::TableField, [86](#)
- lib/config/config.cpp, [99](#)
- lib/config/config.h, [100](#)
- lib/config/config\_getopt.cpp, [101](#)
- lib/database/data\_structures.h, [102](#)
- lib/database/database.h, [103](#)
- lib/database/dbconn.cpp, [105](#)
- lib/database/dbconn.h, [106](#)
- lib/database/dbconnimp.h, [107](#)
- lib/database/table.cpp, [109](#)
- lib/database/table.h, [110](#)
- lib/database/tablefield.cpp, [112](#)
- lib/database/tablefield.h, [112](#)
- lib/database/ablerow.cpp, [114](#)
- lib/database/ablerow.h, [115](#)
- lib/database\_imp/database\_imp.h, [116](#)
- lib/database\_imp/dummy/dbconn\_dummy\_imp.cpp, [118](#)
- lib/database\_imp/dummy/dbconn\_dummy\_imp.h, [119](#)
- lib/database\_imp/mysql/dbconn\_mysql\_functions.cpp, [121](#)
- lib/database\_imp/mysql/dbconn\_mysql\_imp.cpp, [122](#)
- lib/database\_imp/mysql/dbconn\_mysql\_imp.h, [123](#)
- lib/database\_imp/mysql/dbconn\_mysql\_result.cpp, [125](#)
- lib/database\_imp/mysql/dbconn\_mysql\_result.h, [126](#)
- lib/dbsql/dbsql.h, [127](#)
- lib/dbsql/dbsql\_dummy.h, [128](#)
- lib/dbsql/dbsql\_implementations.h, [130](#)
- lib/dbsql/dbsql\_mysql.h, [131](#)
- lib/dbsql/dbsqlstatements.cpp, [132](#)
- lib/dbsql/dbsqlstatements.h, [133](#)
- lib/gldb/gldatabase.cpp, [134](#)
- lib/gldb/gldatabase.h, [135](#)
- lib/gldb/gldb.h, [137](#)
- lib/gldb/glexception.h, [138](#)
- lib/gldb/glreport.cpp, [139](#)
- lib/gldb/glreport.h, [141](#)
- lib/gldb/gluser.cpp, [142](#)
- lib/gldb/gluser.h, [143](#)
- lib/gldb/gluser\_pass.cpp, [144](#)
- lib/pgutils/pgutils.h, [146](#)
- lib/pgutils/stringhelp.cpp, [147](#)
- lib/pgutils/stringhelp.h, [147](#)
- load\_sample\_data
  - genleg::GLDatabase, [64](#)
- login
  - Database program., [28](#)
  - Reporting program., [30](#)
  - User administration program., [32](#)
- m\_conn
  - gldb::DBConnMySQL, [55](#)
- m\_data
  - gldb::TableField, [89](#)
- m\_dbc
  - genleg::GLDatabase, [65](#)
- m\_enabled
  - genleg::GLUser, [72](#)
- m\_fields
  - gldb::TableRow, [98](#)
- m\_firstname
  - genleg::GLUser, [72](#)
- m\_headers
  - gldb::Table, [80](#)
- m\_id
  - genleg::GLUser, [72](#)
- m\_imp
  - gldb::DBConn, [45](#)
- m\_lastname

- genleg::GLUser, 72
- m\_num\_fields
  - gldb::MySQLResult, 74
- m\_opts\_set
  - genleg::Config, 37
- m\_opts\_supp
  - genleg::Config, 37
- m\_pass\_hash
  - genleg::GLUser, 72
- m\_pass\_salt
  - genleg::GLUser, 72
- m\_perms
  - genleg::GLUser, 72
- m\_quoted
  - gldb::Table, 80
- m\_records
  - gldb::Table, 80
- m\_report\_text
  - genleg::GLReport, 67
- m\_result
  - gldb::MySQLResult, 74
- m\_sql
  - genleg::GLDatabase, 65
- m\_tables
  - genleg::GLDatabase, 65
- m\_username
  - genleg::GLUser, 72
- m\_views
  - genleg::GLDatabase, 65
- main
  - Database program., 28
  - Reporting program., 30
  - User administration program., 33
- max\_column\_widths
  - General Ledger database module., 21
- MySQLResult
  - gldb::MySQLResult, 73
- next\_content\_line
  - General purpose utilities., 24
- num\_fields
  - gldb::MySQLResult, 73
  - gldb::Table, 79
- num\_records
  - gldb::Table, 79
- operator std::string
  - gldb::TableField, 86
- operator<<
  - genleg::GLReport, 67
  - gldb::TableField, 89
- operator+=
  - gldb::TableField, 86, 87
- operator=
  - gldb::DBConn, 44
  - gldb::DBConnDummy, 49
  - gldb::DBConnMySQL, 54
  - gldb::MySQLResult, 74
  - gldb::Table, 79
- gldb::TableField, 87, 88
- gldb::TableRow, 97
- pass\_hash
  - genleg::GLUser, 70
- pass\_salt
  - genleg::GLUser, 70
- permissions
  - genleg::GLUser, 70
- plain\_report\_from\_table
  - General Ledger database module., 21
- plain\_row
  - General Ledger database module., 21
- populate\_from\_cmdline
  - genleg::Config, 36
- populate\_from\_file
  - genleg::Config, 37
- print
  - gldb::TableRow, 97
- Program configuration module, 15
- progs/gl\_db/gl\_db\_main.cpp, 149
- progs/gl\_report/gl\_report\_main.cpp, 150
- progs/gl\_user/gl\_user\_main.cpp, 152
- query
  - gldb::DBConn, 44
  - gldb::DBConnDummy, 49
  - gldb::DBConnImp, 52
  - gldb::DBConnMySQL, 54
- record\_string
  - gldb::TableRow, 98
- replace
  - General purpose utilities., 24
- report
  - genleg::GLDatabase, 65
- Reporting program., 29
  - check\_db\_parameters, 29
  - check\_help\_and\_version, 29
  - login, 30
  - main, 30
  - set\_configuration, 30
- result
  - gldb::MySQLResult, 74
- revoke
  - genleg::DBSQLStatements, 60
  - genleg::GLDatabase, 65
- SQL statements module, 19
- select
  - gldb::DBConn, 45
  - gldb::DBConnDummy, 49
  - gldb::DBConnImp, 52
  - gldb::DBConnMySQL, 54
- separator\_row
  - General Ledger database module., 22
- set\_configuration
  - Database program., 28
  - Reporting program., 30

- User administration program., 33
- set\_enabled
  - genleg::GLUser, 71
- set\_firstname
  - genleg::GLUser, 71
- set\_lastname
  - genleg::GLUser, 71
- set\_password
  - genleg::GLUser, 71
- set\_quoted
  - gldb::Table, 80
- set\_user\_password
  - User administration program., 33
- set\_username
  - genleg::GLUser, 71
- show\_user\_details
  - User administration program., 33
- size
  - gldb::TableRow, 98
- split
  - General purpose utilities., 24
- split\_lines
  - General purpose utilities., 25
- Table
  - gldb::Table, 76, 77
- TableBadInputFile
  - gldb::TableBadInputFile, 81
- TableCouldNotOpenInputFile
  - gldb::TableCouldNotOpenInputFile, 83
- TableException
  - gldb::TableException, 84
- TableField
  - gldb::TableField, 85, 86
- TableMismatchedRecordLength
  - gldb::TableMismatchedRecordLength, 90
- TableNoSuchField
  - gldb::TableNoSuchField, 91
- TableNoSuchRecord
  - gldb::TableNoSuchRecord, 92
- TableRow
  - gldb::TableRow, 94, 95
- trim
  - General purpose utilities., 25
- trim\_back
  - General purpose utilities., 25
- trim\_front
  - General purpose utilities., 25
- update\_user
  - genleg::DBSQLStatements, 60
  - genleg::GLDatabase, 65
- User administration program., 31
  - check\_db\_parameters, 31
  - check\_help\_and\_version, 32
  - check\_user\_password, 32
  - enable\_user, 32
  - get\_user, 32
  - login, 32
  - main, 33
  - set\_configuration, 33
  - set\_user\_password, 33
  - show\_user\_details, 33
  - user\_by\_id
    - genleg::DBSQLStatements, 60
  - user\_by\_username
    - genleg::DBSQLStatements, 60
  - username
    - genleg::GLUser, 71