VDBL (Vienna Database Library)

Version 1.2

Reference Manual

Hermann Schichl
University of Vienna, Department of Mathematics
Nordbergstr. 15
A-1090 Wien, Austria
email: `Hermann.Schichl@univie.ac.at`

Technical Report
January 2010

Contents

1 Introduction 1

2 Module Index 4

3 Namespace Index 4

4 Class Index 4

5 Class Index 6

6 File Index 8

7 Module Documentation 9

8 Namespace Documentation 19

9 Class Documentation 19

10 File Documentation 185

1 Introduction

The Vienna Database Library (VDBL) is a in-memory database developed with generic programming in mind. It uses STL containers like `map` and `vector` to organize its internal structure. It structure is designed in such a way that the in-memory structure can be mixed with standard SQL databases.

Databases in the VDBL consist of tables, which are constructed from columns and rows as ordinary relational databases.

Columns can take arbitrary types, and their values need not be constant. Using function objects, called *methods*, the column values can change according to an *evaluation context*.

It is possible to construct *views* onto the tables of a database.

1.1 Database

A VDBL database consists of a number of tables which can be dynamically constructed, changed and destroyed. Every table (see Section Tables) has a unique **name** (a `std::string`) and a unique **table id**, which is used for organizing the internal structure.

There is a general table interface defined in `class table`, which defines the minimal functionality needed for implementing a VDBL table. The structure is defined in such a way that `SQL` interfaces could be written, as well as tables which keep all their data in memory.

In addition to tables, the database knows of *users*. There are access control lists (at the moment not fully implemented) for restricting the access of users to the tables on a global and a column-wise base. The users are defined in the `class user`.

Users can construct **views** onto tables (see Section Views). These views can restrict a table to a subset of columns and/or rows. Also, additional rows can be defined for a view, and it is even possible to *join* various tables into one view. All views onto tables are constructed within a prespecified *context* (see Section Contexts). Using this mechanism, columns can change their value automatically according to the evaluation context. This is, e.g., useful in the COCONUT project for organizing points, where some of the properties change from work node to work node, like whether the point is feasible or not.

1.2 Tables

A VDBL table consists of a number of columns and rows. The definition of a table is always done by specifying its columns (see Section Columns). The type of the columns value, which can be any `C++` type, is fixed upon creating the column. This can be done dynamically, like modifying and removing. Optionally, for each column a *default value* can be given (this default value may also change w.r.t. the evaluation context). All columns within a table have a **name** (a `std::string`) and a **column id**, which is used for organizing the column structure of the table internally. A column of a table can be accessed by specifying its name or, equivalently, its column id.

In addition to the column structure, which determines the outline of the table, the table's data is organized in **rows** (see Section Rows). Every row has a **row id**, which is used for internal organization. Rows themselves consist of columns. When creating a new row, strict type checking is done between the row's column entries and the column type stored in the table. Column entries of a row can be left out, if a default value for the column is specified in the table definition.

It is possible to implement differently organized tables, as long as they are subclasses of the `class table`.

Implemented are two table subclasses:

- **Standard Table:** A table which keeps all data in memory, internally organized as STL maps (`vdbl::standard_table`).
- **View Table:** A table which is constructed from an arbitrary view using the view's internal structure to organize the data (`vdbl::view_table`).

1.3 Columns

VDBL columns are built in a very complicated way using three classes on top of each other, making it possible that arbitrary `C++` types can be stored in a column.

There are two main column classes implemented:

- `typed_col`: This column holds constant values of arbitrary types. Their values are independent of the evaluation context (`class vdbl::typed_ol<_TT>`).
- `method_col`: A column of this type holds data, whose value is computed whenever it is retrieved and may depend on the evaluation context (`vdbl::method_col<_TT>`). Instead of holding data, its contents are function objects (methods), which are subclasses of `class vdbl::method<_TT>`. These function objects are used to calculate the column value.

Within a table different column types can be mixed within different rows and the default value, as long as their content types (strict run-time type checking) coincide.

1.4 Rows

The VDBL rows (`class vdbl::row`) are internally defined as STL maps of columns, organized with **column id** keys and column entries.

In principle, different types of rows could be defined, but at the moment only standard rows are implemented.

Every row contains a number of columns, whose values can be retrieved within an evaluation context (see Section Contexts). The column type within a row is arbitrary. If you want to make sure, that type checking is used, you have to change the rows through the table methods.

1.5 Views

A **view** (`class vdbl::view_base`) onto a table is table-like construct built from table structures. They may be restricted to a subset of the rows and/or columns of the table.

The most important properties of a view is that it is always created within a given context (see Section Contexts). The contents of the view can vary depending on this context. Two different views to the same table can at the same time show different data in the same column of the same row.

Two different classes of views have been implemented:

- **Standard View:** This view (of `class vdbl::view`) is constructed over **one** table, and it can only be restricted to subsets of rows and columns of this table.
- **Hierarchical View:** A hierarchical view (in `class vdbl::hierarchical_view`) looks onto a **stack of tables**, the top ones "overlaying" the lower ones. This makes it possible to have, e.g., a globally valid table on bottom and a stack of locally valid tables on top of them.

Some views can hold an internal **cache** of table entries, which are used for fast access, reducing the number of calls to function objects within columns.

1.6 View Database

A **view database** is a view onto a complete database, automatically constructing views (standard or hierarchical) for every table defined in the database. These views can be accessed under the same names as the defining tables, having a subset of their columns (also with identical names). The defining class is `vdbl::viewdbase`.

1.7 Contexts

Evaluation contexts are subclasses of `class vdbl::context`. They may hold arbitrary data and are keeping a `const vdbl::table *` to their associated table.

This context is passed to every function object along with the row the column belongs to for constructing the columns value. The contexts have no influence on `typed_col` columns, whose values don't change within different contexts.

2 Module Index

2.1 Modules

Here is a list of all modules:

Classes and types for external use 9

Classes and types for internal use 15

3 Namespace Index

3.1 Namespace List

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

vdbl **(Main namespace of the VDBL)** 19

4 Class Index

4.1 Class Hierarchy

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

- **__VDBL_colbase** 19
  - **_VDBL_colbase< _TR >** 34
  - **_VDBL_colbase< _TM::return_type >** 34
    - **_VDBL_mthdcol< _TM::context, _TM, _TM::return_type >** 68
      - **method_col< _TM >** 146
  - **_VDBL_colbase< _TR >** 34
    - **_VDBL_mthdcol< _TC, _TM, _TR >** 68
    - **col_base< _TR >** 126
  - **_VDBL_colbase< _TT >** 34
    - **_VDBL_stdcol< _TT >** 87
    - **_VDBL_stdcol< _TT >** 87
      - **typed_col< _TT >** 167
- **_VDBL_acl** 21
- **_VDBL_aclentry** 23

_VDBL_alltype_base 29
    _VDBL_alltype< _TR > 25
    _VDBL_alltype< _TT > 25
        alltype< _TT > 123
_VDBL_col 31
_VDBL_colflags 38
_VDBL_context 39
_VDBL_database 41
    database 130
_VDBL_date 46
_VDBL_dateinterval 48
_VDBL_method< _TR > 58
_VDBL_method< _TR > 58
    method< _TR > 145
_VDBL_mixtype 59
_VDBL_row 72
    row 150
_VDBL_table 92
    _VDBL_standardtable 74
        standard_table 153
    _VDBL_viewtable 118
    table 163
_VDBL_table::_VDBL_table::_col_iterator_base 98
    _VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr > 97
_VDBL_table::_VDBL_table::_row_iterator< _Tp, _Ref, _Ptr > 99
_VDBL_tableflags 99
_VDBL_userflags 102
_VDBL_view 104
    _VDBL_hierarchicalview 51

hierarchical_view 138
_VDBL_standardview 80
view 172
_VDBL_view::_VDBL_view::_col_iterator_base 108
_VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr > 108
_VDBL_view::_VDBL_view::_default_iterator< _Tp, _Ref, _Ptr > 109
_VDBL_view::_VDBL_view::_row_iterator_base 111
_VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr > 110
_VDBL_viewdbase 111
viewdbase 180
_VDBL_viewflags 116
col_spec 128

5 Class Index

5.1 Class List

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

__VDBL_colbase (The base class of the internal column structure) 19
_VDBL_acl (Access control list) 21
_VDBL_aclentry (Entry in the access control list) 23
_VDBL_alltype< _TR > (The templated class for the all_type class) 25
_VDBL_alltype_base (The base class for the all_type class) 29
_VDBL_col (The generic column class (the external structure)) 31
_VDBL_colbase< _TR > (The type dependent base class of the internal column structure) 34
_VDBL_colflags (Additional table information for a column) 38
_VDBL_context (Base class for context objects) 39
_VDBL_database (Database class) 41
_VDBL_date (The VDBL date class) 46
_VDBL_dateinterval (The VDBL date interval class) 48
_VDBL_hierarchicalview (Hierarchical view class) 51

_VDBL_method< _TR > (Base class for methods usable in _VDBL_mthdcol columns) 58

_VDBL_mixtype (Mixed type) 59

_VDBL_mthdcol< _TC, _TM, _TR > (Generic column class for methods) 68

_VDBL_row (Row class) 72

_VDBL_standardtable (Standard table in databases, constructed from rows and columns) 74

_VDBL_standardview (Standard view onto one table) 80

_VDBL_stdcol< _TT > (Generic column class for constant values) 87

_VDBL_table (Base class describing database tables) 92

_VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr > 97

_VDBL_table::_VDBL_table::_col_iterator_base 98

_VDBL_table::_VDBL_table::_row_iterator< _Tp, _Ref, _Ptr > 99

_VDBL_tableflags (Flags for one table) 99

_VDBL_userflags (The permission flags for a user) 102

_VDBL_view (Base class of all views) 104

_VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr > 108

_VDBL_view::_VDBL_view::_col_iterator_base 108

_VDBL_view::_VDBL_view::_default_iterator< _Tp, _Ref, _Ptr > 109

_VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr > 110

_VDBL_view::_VDBL_view::_row_iterator_base 111

_VDBL_viewdbase (View to a complete database) 111

_VDBL_viewflags (Flags for one view) 116

_VDBL_viewtable 118

alltype< _TT > (The templated alltype class) 123

col_base< _TR > (Column base class) 126

col_spec (Column specification) 128

database (Database class) 130

hierarchical_view (Hierarchical view class onto a stack of tables) 138

method< _TR > (Base class for methods usable in method columns) 145

method_col< _TM > (External name for computed columns) 146

row (Class implementing table rows) 150
standard_table (Standard table of a database) 153
table (Base class for tables in a database) 163
typed_col< _TT > (External name for constant data columns) 167
view (Standard view class onto a single table) 172
viewdbase (View to a complete database) 180

6 File Index

6.1 File List

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

database 185
db_alltype 187
db_col 187
db_context 188
db_expression ??
db_hrview 188
db_index 189
db_joinview 190
db_lexiindex 190
db_listindex 191
db_method 192
db_row 193
db_selector ??
db_table 193
db_user ??
db_view 194
dbs ??
vdbl_alltype.h 195
vdbl_col.h 198

vdbl_config.h 200

vdbl_context.h 201

vdbl_database.h 202

vdbl_dbs.h ??

vdbl_expression.h 204

vdbl_extradocu.h 205

vdbl_hrview.h 206

vdbl_index.h 207

vdbl_joinview.h 208

vdbl_lexiindex.h 209

vdbl_listindex.h 210

vdbl_method.h 211

vdbl_row.h 212

vdbl_selector.h ??

vdbl_stview.h 213

vdbl_table.h 214

vdbl_types.h 215

vdbl_user.h 217

vdbl_view.h 218

vdbl_viewdbase.h 220

vdbl_vtable.h 221

viewdbase 221

7 Module Documentation

7.1 Classes and types for external use

Classes

- class alltype< _TT >

  The templated alltype class.

- class col_base< _TR >

  column base class

- class typed_col< _TT >

    external name for constant data columns

- class method_col< _TM >

    external name for computed columns

- class database

    the database class

- class hierarchical_view

    hierarchical view class onto a stack of tables

- class method< _TR >

    base class for methods usable in `method` columns.

- class row

    class implementing table rows

- class view

    standard view class onto a single table

- class table

    base class for tables in a database

- class col_spec

    column specification

- class standard_table

    standard table of a database

- class viewdbase

    a view to a complete database

Defines

- #define VDBL_MAXUSERID ((__VDBL::_VDBL_userid)-1)
- #define VDBL_MAXVIEWID ((__VDBL::_VDBL_viewid)-1)
- #define VDBL_MAXCOLID ((__VDBL::_VDBL_colid)-1)
- #define VDBL_MAXROWID ((__VDBL::_VDBL_rowid)-1)
- #define VDBL_MAXTABLEID ((__VDBL::_VDBL_tableid)-1)

Typedefs

- typedef _VDBL_date date

    the date type

- typedef _VDBL_dateinterval dateinterval *the dateinterval type*

- typedef _VDBL_mixtype mixtype

  a mixed type of various scalars and vectors

- typedef _VDBL_alltype_base alltype_base

  the base class of the alltype

- typedef _VDBL_col col

  the column class

- typedef _VDBL_stdcol< mixtype > standard_col

  the standard column class with constant `mixtype` *data*

- typedef _VDBL_context context

  evaluation context base class

- typedef _VDBL_userflags userflags

  user flags and permissions

- typedef _VDBL_viewflags viewflags

  view flags and ACLs

- typedef _VDBL_tableflags tableflags

  table flags and ACLs

- typedef _VDBL_aclentry aclentry

  entry in the access control list (ACL)

- typedef _VDBL_acl acl

  ACL for one user.

- typedef _VDBL_userid userid

  user id

- typedef _VDBL_viewid viewid

  view id

- typedef _VDBL_colid colid

  column id

- typedef _VDBL_rowid rowid

  row id

- typedef _VDBL_tableid tableid

  table id

- typedef _VDBL_colflags colflags

  additional column properties

- typedef _V_enum view_enum

    the view type

Variables

- bool _VDBL_colflags::master_index

7.1.1 Detailed Description

The classes and types in this section are for external use.

7.1.2 Define Documentation

7.1.2.1 #define VDBL_MAXCOLID ((__VDBL::_VDBL_colid)-1)

The maximal column id

Definition at line 151 of file vdbl_types.h.

7.1.2.2 #define VDBL_MAXROWID ((__VDBL::_VDBL_rowid)-1)

The maximal row id

Definition at line 156 of file vdbl_types.h.

7.1.2.3 #define VDBL_MAXTABLEID ((__VDBL::_VDBL_tableid)-1)

The maximal table id

Definition at line 161 of file vdbl_types.h.

7.1.2.4 #define VDBL_MAXUSERID ((__VDBL::_VDBL_userid)-1)

The maximal possible values. They mean not initialized The maximal user id

Definition at line 141 of file vdbl_types.h.

7.1.2.5 #define VDBL_MAXVIEWID ((__VDBL::_VDBL_viewid)-1)

The maximal view id

Definition at line 146 of file vdbl_types.h.

7.1.3 Typedef Documentation

7.1.3.1 typedef _VDBL_acl acl

this is the external name for access control lists

Definition at line 1042 of file vdbl_database.h.

7.1.3.2 typedef _VDBL_aclentry aclentry

this is the external name for access control list entries

Definition at line 1036 of file vdbl_database.h.

7.1.3.3 typedef _VDBL_alltype_base alltype_base

The base class of the alltype templated classes

Definition at line 607 of file vdbl_alltype.h.

7.1.3.4 typedef _VDBL_col col

this is the external name of the column class

Definition at line 569 of file vdbl_col.h.

7.1.3.5 typedef _VDBL_colflags colflags

This type describes the additional properties of a column in a table.

Definition at line 192 of file vdbl_types.h.

7.1.3.6 typedef _VDBL_colid colid

The column id type

Definition at line 177 of file vdbl_types.h.

7.1.3.7 typedef _VDBL_context context

this is the external name of the base class for context objects

Definition at line 88 of file vdbl_context.h.

7.1.3.8 typedef _VDBL_date date

the 'official' name of the date class

Definition at line 588 of file vdbl_alltype.h.

7.1.3.9 typedef _VDBL_dateinterval dateinterval

the 'official' name of the dateinterval class

Definition at line 594 of file vdbl_alltype.h.

7.1.3.10 typedef _VDBL_mixtype mixtype

the official name of the mixtype class

Definition at line 600 of file vdbl_alltype.h.

7.1.3.11 typedef _VDBL_rowid rowid

The row id type

Definition at line 182 of file vdbl_types.h.

7.1.3.12 typedef _VDBL_stdcol<mixtype> standard_col

the standard column holds constant data of type mixtype

Definition at line 576 of file vdbl_col.h.

7.1.3.13 typedef _VDBL_tableflags tableflags

this is the external name for table flags

Definition at line 1029 of file vdbl_database.h.

7.1.3.14 typedef _VDBL_tableid tableid

The table id type

Definition at line 187 of file vdbl_types.h.

7.1.3.15 typedef _VDBL_userflags userflags

this is the external name for user flags

Definition at line 1017 of file vdbl_database.h.

7.1.3.16 typedef _VDBL_userid userid

The user id type

Definition at line 167 of file vdbl_types.h.

7.1.3.17 typedef _V_enum view_enum

This type describes the various view types.

Definition at line 198 of file vdbl_types.h.

7.1.3.18 typedef _VDBL_viewflags viewflags

this is the external name for view flags

Definition at line 1023 of file vdbl_database.h.

7.1.3.19 typedef _VDBL_viewid viewid

The view id type

Definition at line 172 of file vdbl_types.h.

7.1.4 Variable Documentation

7.1.4.1 bool _VDBL_colflags::master_index `[inherited]`

see class description

Definition at line 71 of file vdbl_types.h.

7.2 Classes and types for internal use

Classes

- class _VDBL_alltype_base

The base class for the all_type class.

- class _VDBL_alltype< _TR >

  The templated class for the all_type class.

- class _VDBL_date

  The VDBL date class.

- class _VDBL_dateinterval

  The VDBL date interval class.

- class _VDBL_mixtype

  mixed type

- class __VDBL_colbase

  The base class of the internal column structure.

- class _VDBL_colbase< _TR >

  The type dependent base class of the internal column structure.

- class _VDBL_col

  The generic column class (the external structure).

- class _VDBL_stdcol< _TT >

  generic column class for constant values

- class _VDBL_mthdcol< _TC, _TM, _TR >

  generic column class for methods

- class _VDBL_context

  base class for context objects

- class _VDBL_userflags

  The permission flags for a user.

- class _VDBL_aclentry

  entry in the access control list

- class _VDBL_acl

  Access control list.

- class _VDBL_tableflags

  flags for one table

- class _VDBL_viewflags

  flags for one view

- class _VDBL_database

  the database class

- class _VDBL_hierarchicalview

  hierarchical view class

- class _VDBL_method< _TR >

  base class for methods usable in `_VDBL_mthdcol` *columns.*

- class _VDBL_row

  row class

- class _VDBL_standardview

  standard view onto **one** *table*

- class _VDBL_table

  the base class describing database tables

- class _VDBL_standardtable

  standard table in databases, constructed from rows and columns

- class _VDBL_colflags

  additional table information for a column

- class _VDBL_view

  base class of all views.

- class _VDBL_viewdbase

  a view to a complete database

Typedefs

- typedef uint32_t _VDBL_userid
- typedef uint32_t _VDBL_viewid
- typedef uint64_t _VDBL_colid
- typedef uint64_t _VDBL_rowid
- typedef uint32_t _VDBL_tableid

Enumerations

- enum _V_enum

  different view properties

Functions

- template<class _TC >
  std::ostream & print_it (std::ostream &o, const _TC &t)

7.2.1 Detailed Description

The classes and types in this section are used VDBL internally.

7.2.2 Typedef Documentation

7.2.2.1 typedef uint64_t _VDBL_colid

The column id type

Definition at line 93 of file vdbl_types.h.

7.2.2.2 typedef uint64_t _VDBL_rowid

The row id type

Definition at line 97 of file vdbl_types.h.

7.2.2.3 typedef uint32_t _VDBL_tableid

The table id type

Definition at line 101 of file vdbl_types.h.

7.2.2.4 typedef uint32_t _VDBL_userid

The user id type

Definition at line 85 of file vdbl_types.h.

7.2.2.5 typedef uint32_t _VDBL_viewid

The view id type

Definition at line 89 of file vdbl_types.h.

7.2.3 Enumeration Type Documentation

7.2.3.1 enum _V_enum

This enum describes different view properties. Depending on this type, the view behaves differently.

- V_hole: This view looks through to a table. It is possible to change the table contents through the view.
- V_window: This view looks through to a table. It is impossible to change the table contents through the view.

- V_transparent: This view does not change the underlying table. It can be expanded, but changes are not committed to the table
- V_frozen: This view is a constant view to a table. It does not change and cannot be changed.
- V_materialized: The view is the result of a select, and there is not an underlying table
- V_independent: The view is just a temporary collection of rows and columns w/o a table.

Definition at line 121 of file vdbl_types.h.

7.2.4 Function Documentation

7.2.4.1 template<class _TC > std::ostream& print_it (std::ostream & *o*, const _TC & *t*) `[inline]`

This internal function is called from operator<< for columns. This hack was necessary, because a direct call of the operator<< for the class _TC did not work properly with g++ 3.2.

Definition at line 55 of file vdbl_col.h.

8 Namespace Documentation

8.1 vdbl Namespace Reference

Main namespace of the VDBL.

8.1.1 Detailed Description

This is the main namespace holding all classes and functions of the Vienna DataBase Library (VDBL)

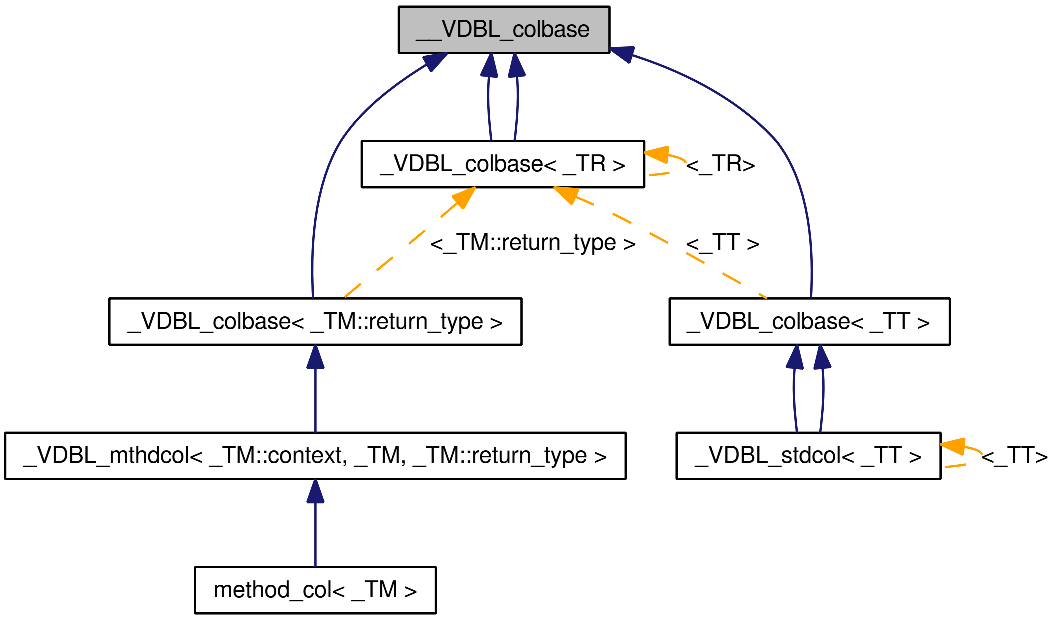
9 Class Documentation

9.1 __VDBL_colbase Class Reference

The base class of the internal column structure.

```
#include <vdbl_col.h>
```

Inheritance diagram for __VDBL_colbase:



Public Member Functions

- virtual __VDBL_colbase ∗ new_copy () const VDBL_PURE_VIRTUALvirtual void setcontext(const context ∗_c

  - __VDBL_colbase ()
  - __VDBL_colbase (const __VDBL_colbase &__v)
  - virtual ∼__VDBL_colbase ()

9.1.1 Detailed Description

__VDBL_colbase is the base class of all columns. This class defines a few virtual functions needed for all columns independent of their type. Especially important is the overloading trick for the (not overloadable) copy-constructor. This is the type independent part of the column implementation

Definition at line 73 of file vdbl_col.h.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 __VDBL_colbase::__VDBL_colbase () `[inline]`

Standard constructor, copy constructor, and destructor

Definition at line 80 of file vdbl_col.h.

9.1.2.2 __VDBL_colbase::__VDBL_colbase (const __VDBL_colbase & *__v*) `[inline]`

Standard constructor, copy constructor, and destructor

Definition at line 81 of file vdbl_col.h.

9.1.2.3 virtual __VDBL_colbase::∼__VDBL_colbase () `[inline, virtual]`

Standard constructor, copy constructor, and destructor

Definition at line 82 of file vdbl_col.h.

9.1.3 Member Function Documentation

9.1.3.1 virtual __VDBL_colbase∗ __VDBL_colbase::new_copy () const `[virtual]`

This function is used to overload the copy constructor. A call to this function defines the context for column value retrieval

Reimplemented in _VDBL_colbase< _TR >, _VDBL_stdcol< _TT >, _VDBL_mthdcol< _TC, _TM, _TR >, _VDBL_colbase< _TM::return_type >, _VDBL_colbase< _TR >, _VDBL_colbase< _TT >, _VDBL_stdcol< _TT >, and _VDBL_mthdcol< _TM::context, _TM, _TM::return_type >.

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

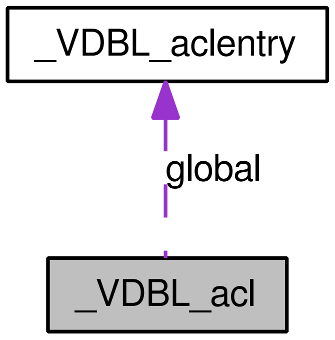
- vdbl_col.h

9.2 _VDBL_acl Class Reference

Access control list.

`#include <vdbl_database.h>`

Collaboration diagram for _VDBL_acl:



Public Member Functions

- _VDBL_acl (bool _gr=false, bool _ga=false, bool _gw=false, bool _gd=false, bool _gctp=false)
- _VDBL_acl (const _VDBL_acl &_a)
- virtual ∼_VDBL_acl ()

- _VDBL_acl & operator= (const _VDBL_acl &__a)

Public Attributes

- _VDBL_aclentry global
- std::map< _VDBL_colid, _VDBL_aclentry > colwise

9.2.1 Detailed Description

This class defines an access control list There is an entry for global access (valid for all columns) and special permissions for various columns

Definition at line 170 of file vdbl_database.h.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 _VDBL_acl::_VDBL_acl (bool *_gr* = `false`, bool *_ga* = `false`, bool *_gw* = `false`, bool *_gd* = `false`, bool *_gctp* = `false`) `[inline]`

standard constructor which optionally initializes the global ACL entry

Definition at line 186 of file vdbl_database.h.

9.2.2.2 _VDBL_acl::_VDBL_acl (const _VDBL_acl & *_a*) `[inline]`

copy constructor

Definition at line 193 of file vdbl_database.h.

9.2.2.3 virtual _VDBL_acl::~_VDBL_acl () `[inline, virtual]`

standard destructor

Definition at line 199 of file vdbl_database.h.

9.2.3 Member Function Documentation

9.2.3.1 _VDBL_acl& _VDBL_acl::operator= (const _VDBL_acl & *__a*) `[inline]`

assignment operator

Definition at line 204 of file vdbl_database.h.

9.2.4 Member Data Documentation

9.2.4.1 std::map<_VDBL_colid,_VDBL_aclentry> _VDBL_acl::colwise

this defines permissions for single columns

Definition at line 180 of file vdbl_database.h.

9.2.4.2 _VDBL_aclentry _VDBL_acl::global

this defines permissions for all columns

Definition at line 176 of file vdbl_database.h.

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

- vdbl_database.h

9.3 _VDBL_aclentry Class Reference

entry in the access control list

`#include <vdbl_database.h>`

Public Member Functions

- _VDBL_aclentry (bool _r=false, bool _a=false, bool _w=false, bool _d=false, bool _ctp=false)
- _VDBL_aclentry (const _VDBL_aclentry &_a)
- virtual ~_VDBL_aclentry ()
- _VDBL_aclentry & operator= (const _VDBL_aclentry &__a)

Public Attributes

- bool read
- bool write
- bool append
- bool drop
- bool commit_to_parent

9.3.1 Detailed Description

This class describes one entry in the access control list of the table or view.

Definition at line 112 of file vdbl_database.h.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 _VDBL_aclentry::_VDBL_aclentry (bool *_r* = `false`, bool *_a* = `false`, bool *_w* = `false`, bool *_d* = `false`, bool *_ctp* = `false`) `[inline]`

standard constructor which optionally initializes the data members

Definition at line 130 of file vdbl_database.h.

9.3.2.2 _VDBL_aclentry::_VDBL_aclentry (const _VDBL_aclentry & *_a*) `[inline]`

copy constructor

Definition at line 139 of file vdbl_database.h.

9.3.2.3 virtual _VDBL_aclentry::∼_VDBL_aclentry () `[inline, virtual]`

standard destructor

Definition at line 147 of file vdbl_database.h.

9.3.3 Member Function Documentation

9.3.3.1 _VDBL_aclentry& _VDBL_aclentry::operator= (const _VDBL_aclentry & *__a*) `[inline]`

assignment operator

Definition at line 152 of file vdbl_database.h.

9.3.4 Member Data Documentation

9.3.4.1 bool _VDBL_aclentry::append

These flags describe the permissions

Definition at line 121 of file vdbl_database.h.

9.3.4.2 bool _VDBL_aclentry::commit_to_parent

These flags describe the permissions

Definition at line 123 of file vdbl_database.h.

9.3.4.3 bool _VDBL_aclentry::drop

These flags describe the permissions

Definition at line 122 of file vdbl_database.h.

9.3.4.4 bool _VDBL_aclentry::read

These flags describe the permissions

Definition at line 119 of file vdbl_database.h.

9.3.4.5 bool _VDBL_aclentry::write

These flags describe the permissions

Definition at line 120 of file vdbl_database.h.

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

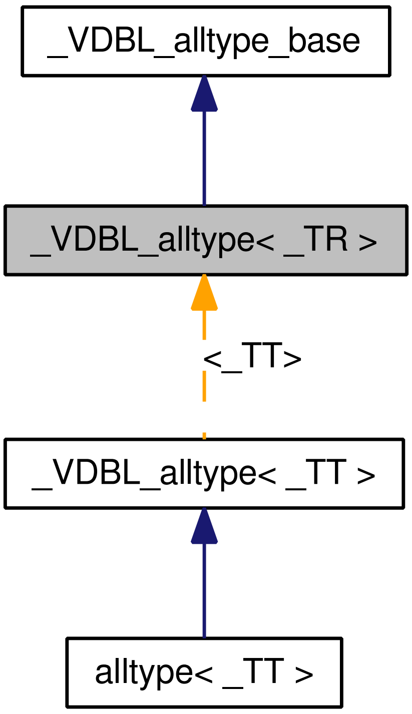
- vdbl_database.h

9.4 _VDBL_alltype< _TR > Class Template Reference

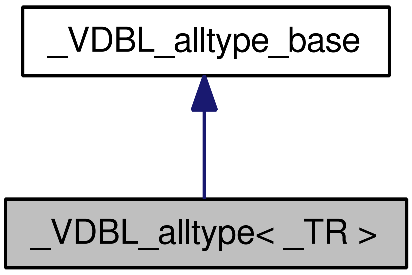
The templated class for the all_type class.

```
#include <vdbl_alltype.h>
```

Inheritance diagram for _VDBL_alltype< _TR >:



Collaboration diagram for _VDBL_alltype< _TR >:



Public Types

- typedef _TR cont_type

    The type this object holds.

Public Member Functions

- _VDBL_alltype ()
- _VDBL_alltype (const cont_type &_p)
- _VDBL_alltype (cont_type ∗_p)
- virtual ∼_VDBL_alltype ()

- const std::type_info & get_type () const
- const cont_type & content () const

- void operator= (const void ∗p)
- void operator= (const cont_type ∗p)

- bool operator== (const _VDBL_alltype_base &p) const
- bool operator!= (const _VDBL_alltype_base &p) const

- bool operator== (const _Self &p) const
- bool operator!= (const _Self &p) const

9.4.1 Detailed Description

template<class _TR> class _VDBL_alltype< _TR >

This class is the templated part of the all_type class. Here the member functions are implemented for every possible type. The class is merely used if values (mostly columns) of unknown type have to be returned and later need to be referenced.

Definition at line 91 of file vdbl_alltype.h.

9.4.2 Constructor & Destructor Documentation

9.4.2.1 template<class _TR> _VDBL_alltype< _TR >::_VDBL_alltype () `[inline]`

This is the empty constructor which produces an empty all_type

Definition at line 111 of file vdbl_alltype.h.

9.4.2.2 template<class _TR> _VDBL_alltype< _TR >::_VDBL_alltype (const cont_type & *_p*) `[inline]`

The standard copy constructor allocates a new data member. Note, that valid data members must provide a copy constructor.

Definition at line 117 of file vdbl_alltype.h.

9.4.2.3 template<class _TR> _VDBL_alltype< _TR >::_VDBL_alltype (cont_type ∗ *_p*) `[inline]`

this constructor is for direct setting of ALREADY allocated values! ONLY use this constructor with pointers whose contents have been allocated using new! This constructor is merely used VDBL internal.

Definition at line 124 of file vdbl_alltype.h.

9.4.2.4 template<class _TR> virtual _VDBL_alltype< _TR >::~_VDBL_alltype () `[inline, virtual]`

The destructor removes the allocated data to prevent memory leaks.

Definition at line 129 of file vdbl_alltype.h.

9.4.3 Member Function Documentation

9.4.3.1 template<class _TR> const cont_type& _VDBL_alltype< _TR >::content () const `[inline]`

This method returns a const reference to the stored data

Definition at line 140 of file vdbl_alltype.h.

9.4.3.2 template<class _TR> const std::type_info& _VDBL_alltype< _TR >::get_type () const `[inline]`

This member function is used for run-time type checking. It returns the of the .

Definition at line 135 of file vdbl_alltype.h.

9.4.3.3 template<class _TR> bool _VDBL_alltype< _TR >::operator!= (const _Self & *p*) const `[inline]`

The standard comparison operators are mainly used for expressions and selectors.

Definition at line 175 of file vdbl_alltype.h.

9.4.3.4 template<class _TR> bool _VDBL_alltype< _TR >::operator!= (const _VDBL_alltype_base & *p*) const `[inline]`

The standard comparison operators are mainly used for expressions and selectors.

Definition at line 164 of file vdbl_alltype.h.

9.4.3.5 template<class _TR> void _VDBL_alltype< _TR >::operator= (const cont_type ∗ *p*) `[inline]`

The assignment operators can take either pointers to the or void pointers. Anyway, the data passed is copied and reallocated. So it is safe to use or destroy the data passed after the assignment operator has been called.

Definition at line 152 of file vdbl_alltype.h.

9.4.3.6 template<class _TR> void _VDBL_alltype< _TR >::operator= (const void * *p*) [inline]

The assignment operators can take either pointers to the or void pointers. Anyway, the data passed is copied and reallocated. So it is safe to use or destroy the data passed after the assignment operator has been called.

Definition at line 149 of file vdbl_alltype.h.

9.4.3.7 template<class _TR> bool _VDBL_alltype< _TR >::operator== (const _Self & *p*) const [inline]

The standard comparison operators are mainly used for expressions and selectors.

Definition at line 174 of file vdbl_alltype.h.

9.4.3.8 template<class _TR> bool _VDBL_alltype< _TR >::operator== (const _VDBL_alltype_base & *p*) const [inline]

The standard comparison operators are mainly used for expressions and selectors.

Definition at line 161 of file vdbl_alltype.h.

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

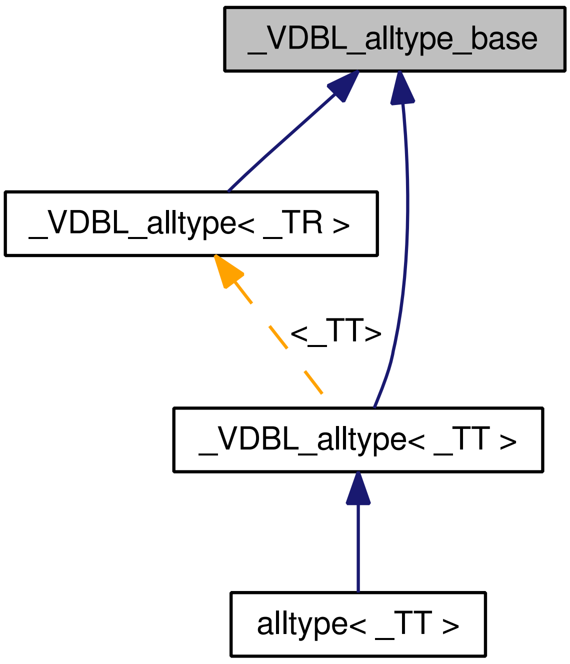
- vdbl_alltype.h

9.5 _VDBL_alltype_base Class Reference

The base class for the all_type class.

`#include <vdbl_alltype.h>`

Inheritance diagram for _VDBL_alltype_base:



Public Member Functions

- _VDBL_alltype_base ()
- virtual ∼_VDBL_alltype_base ()

9.5.1 Detailed Description

This class is the base for all templated all_type classes. All important members are purely virtual. The class is merely used if values (mostly columns) of unknown type have to be returned.

Definition at line 55 of file vdbl_alltype.h.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 _VDBL_alltype_base::_VDBL_alltype_base () `[inline]`

standard constructor

Definition at line 61 of file vdbl_alltype.h.

9.5.2.2 virtual _VDBL_alltype_base::∼_VDBL_alltype_base () `[inline, virtual]`

standard destructor

Definition at line 65 of file vdbl_alltype.h.

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

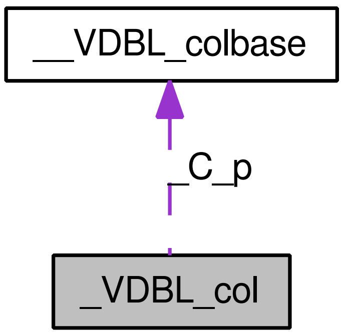
- vdbl_alltype.h

9.6 _VDBL_col Class Reference

The generic column class (the external structure).

`#include <vdbl_col.h>`

Collaboration diagram for _VDBL_col:



Public Member Functions

- _VDBL_col ()

  generic constructor

- _VDBL_col (const _VDBL_col &__c)

  copy constructor - using copy constructor overloading of the base class

- _VDBL_col (__VDBL_colbase ∗__p)
- template<class _RR >
  _VDBL_col (const _RR &_c)
- virtual ∼_VDBL_col ()

  standard destructor

- void setcontext (const context ∗_c, const _VDBL_row ∗_r)

  set the context for value retrieval

- template<class _TR >
  void set (_VDBL_colbase< _TR > ∗_p)
- template<class _TR >
  void get (_TR &c) const
- template<class _TR >
  void def (_TR &d) const
- template<class _TR >
  void get_ptr (_TR const ∗&c) const

- template<class _TR >
  void get_copy (_TR ∗&p) const
- template<class _TR >
  void def_copy (_TR ∗&p) const
- void get_copy (_VDBL_alltype_base ∗&v) const
- void def_copy (_VDBL_alltype_base ∗&d) const
- const std::type_info & return_type_id () const
- const __VDBL_colbase ∗ get_ptr_to_val () const

Friends

- std::ostream & operator<< (std::ostream &o, const _VDBL_col &c)

9.6.1 Detailed Description

`_VDBL_col` is the generic column class. It contains the actual data, and columns of this type are stored in rows. The copy constructor and operator<< are overloaded using the virtual functions defined in the `__VDBL_colbase` and afterwards in the `_VDBL_colbase<_TT>` classes.

This is the third and final step in constructing columns of arbitrary type.

Definition at line 242 of file vdbl_col.h.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 _VDBL_col::_VDBL_col (__VDBL_colbase ∗ *__p*) `[inline, explicit]`

direct constructor - handle with care, no implicit copying is done, the destructor, however, will try to delete __p. This is mostly used VDBL internal. If you want to use it, you should KNOW WHAT YOU ARE DOING!

Definition at line 261 of file vdbl_col.h.

9.6.2.2 template<class _RR > _VDBL_col::_VDBL_col (const _RR & *_c*) `[inline, explicit]`

This is a generic type independent constructor. It produces a column of type _RR. The column data is copied, so it is save to destroy the `_c` data afterwards.

Definition at line 366 of file vdbl_col.h.

9.6.3 Member Function Documentation

9.6.3.1 template<class _TR > void _VDBL_col::def (_TR & *d*) const `[inline]`

This function stores a copy of the column default value into `d`.

Definition at line 380 of file vdbl_col.h.

9.6.3.2 void _VDBL_col::def_copy (_VDBL_alltype_base *& *d*) const [inline]

This version of get_copy returns a copy of the columns default value within an alltype. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Definition at line 397 of file vdbl_col.h.

9.6.3.3 template<class _TR > void _VDBL_col::def_copy (_TR *& *p*) const [inline]

This function returns a pointer to a copy of the column's default value. The copy of the value is allocated by new. It has to be deleted by the user to avoid memory leaks.

Definition at line 391 of file vdbl_col.h.

9.6.3.4 template<class _TR > void _VDBL_col::get (_TR & *c*) const [inline]

This function stores a copy of the column value into c.

Definition at line 377 of file vdbl_col.h.

9.6.3.5 void _VDBL_col::get_copy (_VDBL_alltype_base *& *v*) const [inline]

This version of get_copy returns a copy of the columns value within an alltype. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Definition at line 394 of file vdbl_col.h.

9.6.3.6 template<class _TR > void _VDBL_col::get_copy (_TR *& *p*) const [inline]

This function returns a pointer to a copy of the column's value. The copy of the value is allocated by new. It has to be deleted by the user to avoid memory leaks.

Definition at line 387 of file vdbl_col.h.

9.6.3.7 template<class _TR > void _VDBL_col::get_ptr (_TR const *& *c*) const [inline]

This function sets c to a const pointer pointing to the column's actual value. Here, no copying is done.

Definition at line 383 of file vdbl_col.h.

9.6.3.8 const __VDBL_colbase* _VDBL_col::get_ptr_to_val () const [inline]

This function is needed for the operator<< for columns of type `return_type`.

Definition at line 349 of file vdbl_col.h.

9.6.3.9 const std::type_info& _VDBL_col::return_type_id () const [inline]

This function returns the `type_info` of the column type. This information is used during run-time type checking.

Definition at line 342 of file vdbl_col.h.

9.6.3.10 template<class _TR > void _VDBL_col::set (_VDBL_colbase< _TR > * *_p*) [inline]

This function sets the data to the pointer passed. This is a direct set operation - handle with care, no implicit copying is done. The destructor, however, will try to delete __p. This is mostly used VDBL internal. If you want to use it, you should KNOW WHAT YOU ARE DOING!

Definition at line 285 of file vdbl_col.h.

9.6.4 Friends And Related Function Documentation

9.6.4.1 std::ostream& operator<< (std::ostream & *o*, const _VDBL_col & *c*) [friend]

The print operation for generic columns. This implicitely calls operator<< for the columns type. So it is necessary that this operator is indeed defined.

Definition at line 360 of file vdbl_col.h.

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

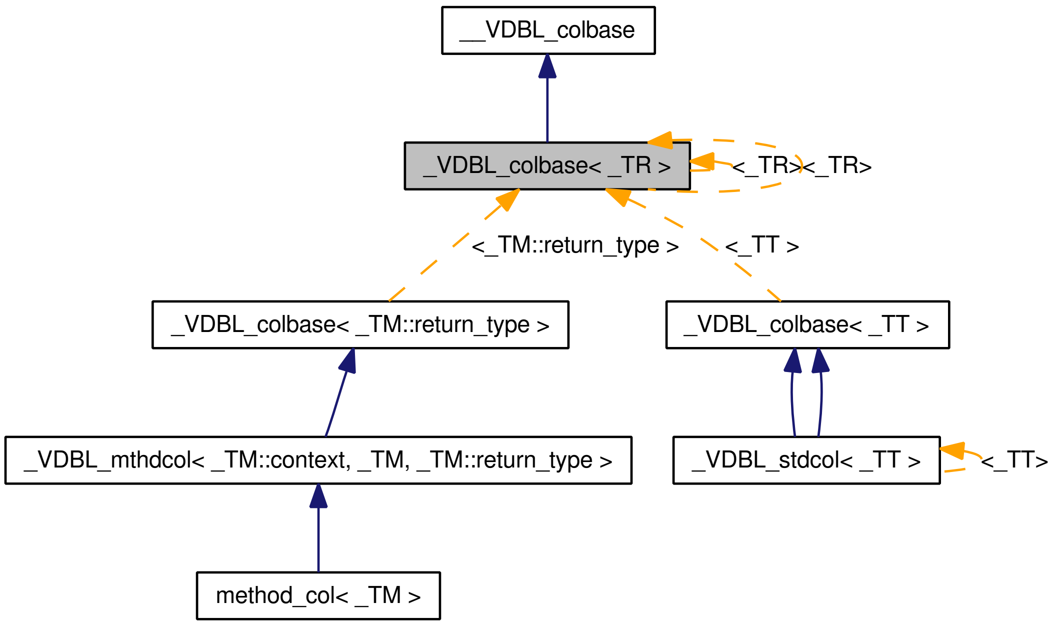
- vdbl_col.h

9.7 _VDBL_colbase< _TR > Class Template Reference

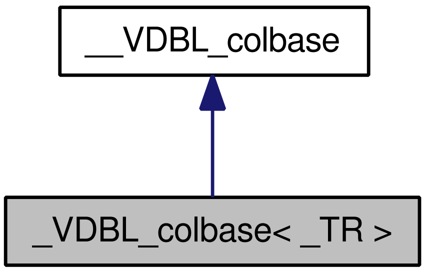
The type dependent base class of the internal column structure.

```
#include <vdbl_col.h>
```

Inheritance diagram for _VDBL_colbase< _TR >:



Collaboration diagram for _VDBL_colbase< _TR >:



Public Types

- typedef _TR return_type

  `return_type` *is the type of object stored*

Public Member Functions

- virtual _Self ∗ new_copy () const VDBL_PURE_VIRTUALvirtual void setcontext(const context ∗_c
- virtual _Self ∗const _VDBL_row ∗_r virtual VDBL_PURE_VIRTUAL void get (return_type &c) const VDBL_PURE_VIRTUALvirtual void def(return_type &d) const VDBL_PURE_-VIRTUALvirtual void get_ptr(return_type const ∗&c) const VDBL_PURE_VIRTUALvirtual void get_copy(return_type ∗&c) const
- virtual void def_copy (return_type ∗&d) const
- virtual void get_copy (_VDBL_alltype_base ∗&v) const

- virtual void def_copy (_VDBL_alltype_base ∗&v) const
- virtual const std::type_info & return_type_id () const
- virtual std::ostream & print_contents (std::ostream &o) const

  - _VDBL_colbase ()
  - _VDBL_colbase (const _Self &__c)
  - virtual ∼_VDBL_colbase ()

9.7.1 Detailed Description

template<class _TR> class _VDBL_colbase< _TR >

`_VDBL_colbase` is the templated base class of all columns of the same type - for copy-constructor and get-operation overload. This class is the second step. The first step is done in __VDBL_colbase, which makes columns "type independent". The second step makes it possible to have different kinds of columns of the same type. All type dependent member functions are virtual in this class.

Definition at line 132 of file vdbl_col.h.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 template<class _TR> _VDBL_colbase< _TR >::_VDBL_colbase () `[inline]`

standard constructor, copy constructor, and destructor

Definition at line 144 of file vdbl_col.h.

9.7.2.2 template<class _TR> _VDBL_colbase< _TR >::_VDBL_colbase (const _Self & *__c*) `[inline]`

standard constructor, copy constructor, and destructor

Definition at line 145 of file vdbl_col.h.

9.7.2.3 template<class _TR> virtual _VDBL_colbase< _TR >::∼_VDBL_colbase () `[inline, virtual]`

standard constructor, copy constructor, and destructor

Definition at line 146 of file vdbl_col.h.

9.7.3 Member Function Documentation

9.7.3.1 template<class _TR> virtual void _VDBL_colbase< _TR >::def_copy (_VDBL_alltype_base ∗& *v*) const `[inline, virtual]`

This version of `get_copy` returns a copy of the columns default value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Reimplemented in _VDBL_stdcol< _TT >, and _VDBL_stdcol< _TT >.

Definition at line 210 of file vdbl_col.h.

9.7.3.2 template<class _TR> virtual void _VDBL_colbase< _TR >::def_copy (return_type ∗& *d*) const `[inline, virtual]`

This function returns a pointer to a copy of the column's default value. The copy of the value is allocated by `new`. It has to be `delete`d by the user to avoid memory leaks.

Definition at line 188 of file vdbl_col.h.

9.7.3.3 template<class _TR> virtual _Self∗ const _VDBL_row∗ _r virtual VDBL_PURE_VIRTUAL void _VDBL_colbase< _TR >::get (return_type & *c*) const `[inline, virtual]`

This function stores a copy of the column value into `c`. This function stores a copy of the column default value into `d`. This function sets `c` to a const pointer pointing to the column's actual value. Here, no copying is done. This function returns a pointer to a copy of the column's value. The copy of the value is allocated by `new`. It has to be `delete`d by the user to avoid memory leaks.

Reimplemented in _VDBL_stdcol< _TT >, _VDBL_mthdcol< _TC, _TM, _TR >, _VDBL_stdcol< _TT >, and _VDBL_mthdcol< _TM::context, _TM, _TM::return_type >.

Definition at line 162 of file vdbl_col.h.

9.7.3.4 template<class _TR> virtual void _VDBL_colbase< _TR >::get_copy (_VDBL_alltype_base ∗& *v*) const `[inline, virtual]`

This version of `get_copy` returns a copy of the columns value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Reimplemented in _VDBL_stdcol< _TT >, and _VDBL_stdcol< _TT >.

Definition at line 197 of file vdbl_col.h.

9.7.3.5 template<class _TR> virtual _Self∗ _VDBL_colbase< _TR >::new_copy () const `[virtual]`

`new_copy` is the clone operation for copy-constructor overloading. `setcontext` sets the context for value retrieval.

Reimplemented from __VDBL_colbase.

Reimplemented in _VDBL_stdcol< _TT >, _VDBL_mthdcol< _TC, _TM, _TR >, _VDBL_stdcol< _TT >, and _VDBL_mthdcol< _TM::context, _TM, _TM::return_type >.

9.7.3.6 template<class _TR> virtual std::ostream& _VDBL_colbase< _TR >::print_contents (std::ostream & *o*) const `[inline, virtual]`

This function is needed for the operator<< for columns of type `return_type`.

Reimplemented in _VDBL_stdcol< _TT >, _VDBL_mthdcol< _TC, _TM, _TR >, _VDBL_stdcol< _TT >, and _VDBL_mthdcol< _TM::context, _TM, _TM::return_type >.

Definition at line 228 of file vdbl_col.h.

9.7.3.7 template<class _TR> virtual const std::type_info& _VDBL_colbase< _TR >::return_type_id () const `[inline, virtual]`

This function returns the `type_info` of the column type. This information is used during run-time type checking.

Definition at line 221 of file vdbl_col.h.

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

- vdbl_col.h

9.8 _VDBL_colflags Class Reference

additional table information for a column

`#include <vdbl_types.h>`

Public Member Functions

- _VDBL_colflags (bool __d=false, bool __mi=false)
- ~_VDBL_colflags ()

Public Attributes

- bool master_index

9.8.1 Detailed Description

`_VDBL_colflags` contains the additional table information for a column

- `has_default` does this column have a default value?
- `master_index` is this column a master_index, i.e. are all entries throughout the table unique?

Definition at line 66 of file vdbl_types.h.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 _VDBL_colflags::_VDBL_colflags (bool *__d* = `false`, bool *__mi* = `false`) `[inline]`

standard constructor, optionally setting the entries

Definition at line 76 of file vdbl_types.h.

9.8.2.2 _VDBL_colflags::∼_VDBL_colflags () `[inline]`

standard destructor

Definition at line 79 of file vdbl_types.h.

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

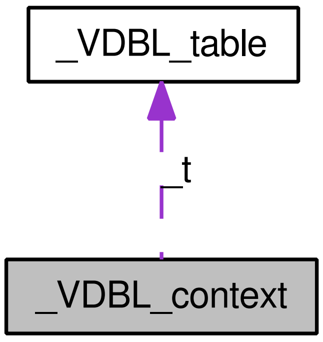
- vdbl_types.h

9.9 _VDBL_context Class Reference

base class for context objects

`#include <vdbl_context.h>`

Collaboration diagram for _VDBL_context:



Public Member Functions

- _VDBL_context (const _VDBL_table ∗t)
- const _VDBL_table ∗ table () const
- void table (const _VDBL_table ∗t)

  - _VDBL_context ()
  - _VDBL_context (const _VDBL_context &_c)
  - virtual ∼_VDBL_context ()

9.9.1 Detailed Description

this is the base class for all context objects in the VDBL.

Definition at line 51 of file vdbl_context.h.

9.9.2 Constructor & Destructor Documentation

9.9.2.1 _VDBL_context::_VDBL_context () `[inline]`

standard constructor, copy constructor, and destructor.

Definition at line 62 of file vdbl_context.h.

9.9.2.2 _VDBL_context::_VDBL_context (const _VDBL_context & *_c*) `[inline]`

standard constructor, copy constructor, and destructor.

Definition at line 63 of file vdbl_context.h.

9.9.2.3 virtual _VDBL_context::∼_VDBL_context () `[inline, virtual]`

standard constructor, copy constructor, and destructor.

Definition at line 64 of file vdbl_context.h.

9.9.2.4 _VDBL_context::_VDBL_context (const _VDBL_table ∗ *t*) `[inline]`

constructor which explicitely sets the table pointer

Definition at line 70 of file vdbl_context.h.

9.9.3 Member Function Documentation

9.9.3.1 void _VDBL_context::table (const _VDBL_table ∗ *t*) `[inline]`

set the table pointer

Definition at line 80 of file vdbl_context.h.

9.9.3.2 const _VDBL_table∗ _VDBL_context::table () const `[inline]`

retrieve a pointer to the table this object belongs to

Definition at line 75 of file vdbl_context.h.

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

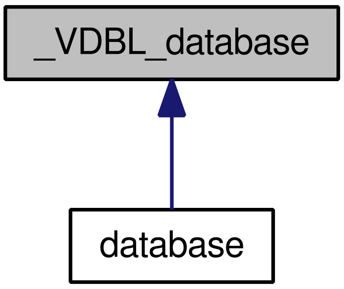
- vdbl_context.h

9.10 _VDBL_database Class Reference

the database class

`#include <vdbl_database.h>`

Inheritance diagram for _VDBL_database:



Public Member Functions

- bool create_table (const std::string &_C_i, const _VDBL_userid &_C_u, const _VDBL_tableflags &__f=_VDBL_tableflags())
- _VDBL_tableid get_tableid (const std::string &_C_i, const _VDBL_userid &_C_u) const
- bool drop_table (const _D_tables::iterator &__t, const _D_table_names::iterator &__tn, const _VDBL_userid &_C_u)
- bool drop_table (const _VDBL_tableid &_C_i, const _VDBL_userid &_C_u)
- bool drop_table (const std::string &_C_i, const _VDBL_userid &_C_u)
- bool has_table (const _VDBL_tableid &_C_i, const _VDBL_userid &_C_u) const
- bool has_table (const std::string &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_table ∗ get_table (const _VDBL_tableid &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_table ∗ get_table (const std::string &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_viewid get_viewid (const std::string &_C_i, const _VDBL_userid &_C_u) const
- bool create_view (const std::string &_C_i, const _VDBL_userid &_C_u, const _VDBL_context &__c, const std::string &_C_t, const _V_enum &__e)
- bool drop_view (const _D_views::iterator &__v, const _D_view_names::iterator &__vn, const _VDBL_userid &_C_u)
- bool drop_view (const _VDBL_viewid &_C_i, const _VDBL_userid &_C_u)
- bool drop_view (const std::string &_C_i, const _VDBL_userid &_C_u)
- bool has_view (const _VDBL_viewid &_C_i, const _VDBL_userid &_C_u) const
- bool has_view (const std::string &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_view ∗ get_view (const _VDBL_viewid &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_view ∗ get_view (const std::string &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_database ()
- _VDBL_database (const _VDBL_database &__t)
- virtual ∼_VDBL_database ()

Protected Member Functions

- _VDBL_tableid get_tableid ()
- _VDBL_userid get_userid ()
- _VDBL_viewid get_viewid ()

Friends

- class _VDBL_viewdbase

9.10.1 Detailed Description

This is the base class for a whole database including tables and views.

Definition at line 337 of file vdbl_database.h.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 _VDBL_database::_VDBL_database () `[inline]`

standard constructor

Definition at line 794 of file vdbl_database.h.

9.10.2.2 _VDBL_database::_VDBL_database (const _VDBL_database & *__t*) `[inline]`

copy constructor

Definition at line 800 of file vdbl_database.h.

9.10.2.3 virtual _VDBL_database::∼_VDBL_database () `[inline, virtual]`

standard destructor

Definition at line 814 of file vdbl_database.h.

9.10.3 Member Function Documentation

9.10.3.1 bool _VDBL_database::create_table (const std::string & *_C_i*, const _VDBL_userid & *_C_u*, const _VDBL_tableflags & *__f* = _VDBL_tableflags`()`) `[inline]`

create a new table

- `_C_i`: name
- `_C_u`: user id

- `__f`: the table flags (if they are not default) return `true`, if creating the table was successful.

Reimplemented in database.

Definition at line 406 of file vdbl_database.h.

9.10.3.2 bool _VDBL_database::create_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*, const _VDBL_context & *__c*, const std::string & *_C_t*, const _V_enum & *__e*) `[inline]`

create a new standard view with name `_C_i`, evaluation context `__c`, for table `_C_t`, of type `__e`. return `true` if creating worked, and `false` otherwise.

Reimplemented in database.

Definition at line 675 of file vdbl_database.h.

9.10.3.3 bool _VDBL_database::drop_table (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) `[inline]`

delete a table, whose name is provided. return `true`, if deleting the table has worked.

Reimplemented in database.

Definition at line 509 of file vdbl_database.h.

9.10.3.4 bool _VDBL_database::drop_table (const _VDBL_tableid & *_C_i*, const _VDBL_userid & *_C_u*) `[inline]`

delete a table, whose table id is provided. return `true`, if deleting the table has worked.

Definition at line 496 of file vdbl_database.h.

9.10.3.5 bool _VDBL_database::drop_table (const _D_tables::iterator & *__t*, const _D_table_names::iterator & *__tn*, const _VDBL_userid & *_C_u*) `[inline]`

delete a table, internal function. The first argument is the iterator into the table map, the second argument is the iterator into the table names map, return `true`, if deleting the table has worked.

Definition at line 471 of file vdbl_database.h.

9.10.3.6 bool _VDBL_database::drop_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) `[inline]`

delete a view, whose name is provided. return `true`, if deleting the table has worked.

Reimplemented in database.

Definition at line 742 of file vdbl_database.h.

9.10.3.7 bool _VDBL_database::drop_view (const _VDBL_viewid & *_C_i*, const _VDBL_userid & *_C_u*) `[inline]`

delete a view, whose id is provided. return `true`, if deleting the table has worked.

Definition at line 729 of file vdbl_database.h.

9.10.3.8 bool _VDBL_database::drop_view (const _D_views::iterator & *__v*, const _D_view_names::iterator & *__vn*, const _VDBL_userid & *_C_u*) `[inline]`

delete a view, internal function. The first argument is the iterator into the table map, the second argument is the iterator into the table names map, return `true`, if deleting the table has worked.

Definition at line 704 of file vdbl_database.h.

9.10.3.9 _VDBL_table* _VDBL_database::get_table (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the table with name `_C_i`.

Reimplemented in database.

Definition at line 550 of file vdbl_database.h.

9.10.3.10 _VDBL_table* _VDBL_database::get_table (const _VDBL_tableid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the table with id `_C_i`.

Definition at line 537 of file vdbl_database.h.

9.10.3.11 _VDBL_tableid _VDBL_database::get_tableid (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return the table id for a given name

Definition at line 448 of file vdbl_database.h.

9.10.3.12 _VDBL_tableid _VDBL_database::get_tableid () `[inline, protected]`

generate a new unique id for tables, views, and users

Definition at line 392 of file vdbl_database.h.

9.10.3.13 _VDBL_userid _VDBL_database::get_userid () `[inline, protected]`

generate a new unique id for tables, views, and users

Definition at line 393 of file vdbl_database.h.

9.10.3.14 _VDBL_view∗ _VDBL_database::get_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the view with name `_C_i`.

Reimplemented in database.

Definition at line 781 of file vdbl_database.h.

9.10.3.15 _VDBL_view∗ _VDBL_database::get_view (const _VDBL_viewid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the view with id `_C_i`.

Definition at line 769 of file vdbl_database.h.

9.10.3.16 _VDBL_viewid _VDBL_database::get_viewid (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return the view id of view `_C_i`.

Definition at line 661 of file vdbl_database.h.

9.10.3.17 _VDBL_viewid _VDBL_database::get_viewid () `[inline, protected]`

generate a new unique id for tables, views, and users

Definition at line 394 of file vdbl_database.h.

9.10.3.18 bool _VDBL_database::has_table (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

check whether the table `_C_i` exists

Reimplemented in database.

Definition at line 530 of file vdbl_database.h.

9.10.3.19 bool _VDBL_database::has_table (const _VDBL_tableid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

check whether the table `_C_i` exists

Definition at line 518 of file vdbl_database.h.

9.10.3.20 bool _VDBL_database::has_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

check whether the view `_C_i` exists

Reimplemented in database.

Definition at line 763 of file vdbl_database.h.

9.10.3.21 bool _VDBL_database::has_view (const _VDBL_viewid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

check whether the view with id `_C_i` exists

Definition at line 751 of file vdbl_database.h.

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

- vdbl_database.h

9.11 _VDBL_date Class Reference

The VDBL date class.

`#include <vdbl_alltype.h>`

Public Member Functions

- _VDBL_dateinterval operator- (const _VDBL_date &_v) const

  - _VDBL_date & operator- (const _VDBL_dateinterval &_v) const
  - _VDBL_date & operator-= (const _VDBL_dateinterval &_v)
  - _VDBL_date & operator+ (const _VDBL_dateinterval &_v) const
  - _VDBL_date & operator+= (const _VDBL_dateinterval &_v)

Public Attributes

- int timezone

  - int year
  - signed char month
  - signed char day
  - unsigned int seconds
  - unsigned int microseconds

9.11.1 Detailed Description

The date base class for the database

Definition at line 186 of file vdbl_alltype.h.

9.11.2 Member Function Documentation

9.11.2.1 _VDBL_date& _VDBL_date::operator+ (const _VDBL_dateinterval & *_v*) const

These operators add or subtract time differences to a date

9.11.2.2 _VDBL_date& _VDBL_date::operator+= (const _VDBL_dateinterval & *_v*)

These operators add or subtract time differences to a date

9.11.2.3 _VDBL_date& _VDBL_date::operator- (const _VDBL_dateinterval & *_v*) const

These operators add or subtract time differences to a date

9.11.2.4 _VDBL_dateinterval _VDBL_date::operator- (const _VDBL_date & *_v*) const

This method computes the time difference between two dates.

9.11.2.5 _VDBL_date& _VDBL_date::operator-= (const _VDBL_dateinterval & *_v*)

These operators add or subtract time differences to a date

9.11.3 Member Data Documentation

9.11.3.1 signed char _VDBL_date::day

the date is stored as year, month, day with the time in seconds and microseconds.

Definition at line 199 of file vdbl_alltype.h.

9.11.3.2 unsigned int _VDBL_date::microseconds

the date is stored as year, month, day with the time in seconds and microseconds.

Definition at line 201 of file vdbl_alltype.h.

9.11.3.3 signed char _VDBL_date::month

the date is stored as year, month, day with the time in seconds and microseconds.

Definition at line 199 of file vdbl_alltype.h.

9.11.3.4 unsigned int _VDBL_date::seconds

the date is stored as year, month, day with the time in seconds and microseconds.

Definition at line 200 of file vdbl_alltype.h.

9.11.3.5 int _VDBL_date::timezone

This defines the timezone in minutes deviation from GMT.

Definition at line 192 of file vdbl_alltype.h.

9.11.3.6 int _VDBL_date::year

the date is stored as year, month, day with the time in seconds and microseconds.

Definition at line 198 of file vdbl_alltype.h.

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

- vdbl_alltype.h

9.12 _VDBL_dateinterval Class Reference

The VDBL date interval class.

`#include <vdbl_alltype.h>`

Public Member Functions

- _VDBL_date operator+ (const _VDBL_date &_d) const

  - _VDBL_dateinterval operator∗ (double d) const
  - _VDBL_dateinterval operator/ (double d) const
  - _VDBL_dateinterval & operator∗= (double d)
  - _VDBL_dateinterval & operator/= (double d)

  - _VDBL_dateinterval & operator- (const _VDBL_dateinterval &_v) const

- _VDBL_dateinterval & operator-= (const _VDBL_dateinterval &_v)
- _VDBL_dateinterval & operator+ (const _VDBL_dateinterval &_v) const
- _VDBL_dateinterval & operator+= (const _VDBL_dateinterval &_v)

Public Attributes

- int years
- short int days
- int seconds
- int microseconds

9.12.1 Detailed Description

The date interval base class for the database. This class describes the difference between two dates.

Definition at line 254 of file vdbl_alltype.h.

9.12.2 Member Function Documentation

9.12.2.1 _VDBL_dateinterval _VDBL_dateinterval::operator∗ (double *d*) const

Date differences can be multiplied by or divided through doubles. E.g., half the time between two dates would be (d1-d2)/2.

9.12.2.2 _VDBL_dateinterval& _VDBL_dateinterval::operator∗= (double *d*)

Date differences can be multiplied by or divided through doubles. E.g., half the time between two dates would be (d1-d2)/2.

9.12.2.3 _VDBL_date _VDBL_dateinterval::operator+ (const _VDBL_date & *_d*) const

Add a date difference to a date

9.12.2.4 _VDBL_dateinterval& _VDBL_dateinterval::operator+ (const _VDBL_dateinterval & *_v*) const

Date differences can be added/subtracted using this operator

9.12.2.5 _VDBL_dateinterval& _VDBL_dateinterval::operator+= (const _VDBL_dateinterval & *_v*)

Date differences can be added/subtracted using this operator

9.12.2.6 _VDBL_dateinterval& _VDBL_dateinterval::operator- (const _VDBL_dateinterval & *_v*) const

Date differences can be added/subtracted using this operator

9.12.2.7 _VDBL_dateinterval& _VDBL_dateinterval::operator-= (const _VDBL_dateinterval & *_v*)

Date differences can be added/subtracted using this operator

9.12.2.8 _VDBL_dateinterval _VDBL_dateinterval::operator/ (double *d*) const

Date differences can be multiplied by or divided through doubles. E.g., half the time between two dates would be (d1-d2)/2.

9.12.2.9 _VDBL_dateinterval& _VDBL_dateinterval::operator/= (double *d*)

Date differences can be multiplied by or divided through doubles. E.g., half the time between two dates would be (d1-d2)/2.

9.12.3 Member Data Documentation

9.12.3.1 short int _VDBL_dateinterval::days

the date differencs is stored as year, day with the time difference in seconds and microseconds.

Definition at line 263 of file vdbl_alltype.h.

9.12.3.2 int _VDBL_dateinterval::microseconds

the date differencs is stored as year, day with the time difference in seconds and microseconds.

Definition at line 265 of file vdbl_alltype.h.

9.12.3.3 int _VDBL_dateinterval::seconds

the date differencs is stored as year, day with the time difference in seconds and microseconds.

Definition at line 264 of file vdbl_alltype.h.

9.12.3.4 int _VDBL_dateinterval::years

the date differencs is stored as year, day with the time difference in seconds and microseconds.

Definition at line 262 of file vdbl_alltype.h.

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

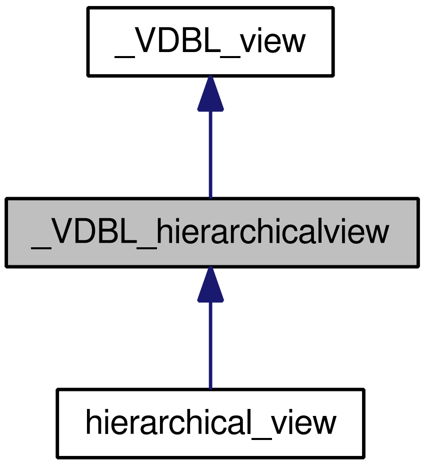
- vdbl_alltype.h

9.13 _VDBL_hierarchicalview Class Reference

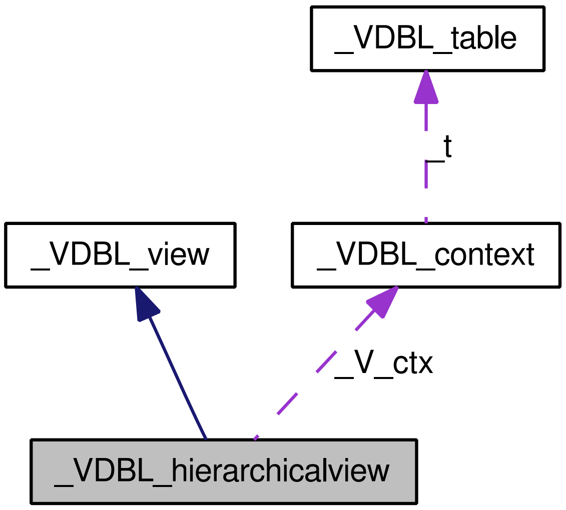
hierarchical view class

`#include <vdbl_hrview.h>`

Inheritance diagram for _VDBL_hierarchicalview:



Collaboration diagram for _VDBL_hierarchicalview:



Public Types

- typedef std::pair< std::string, _VDBL_col > _T_colspec

Public Member Functions

- _VDBL_hierarchicalview (const _VDBL_tableid &__ti, _VDBL_table ∗__t, const _VDBL_context &__c, _V_enum __en)
- _VDBL_hierarchicalview (const _VDBL_tableid &__ti, _VDBL_table ∗__t, const _VDBL_context &__c, _V_enum __en, const std::vector< _VDBL_rowid > &_rs)
- _VDBL_hierarchicalview (const _VDBL_hierarchicalview &__v)
- virtual ∼_VDBL_hierarchicalview ()
- void push_table (const _VDBL_tableid &__ti, _VDBL_table ∗__t)
- void push_table (const _VDBL_tableid &__ti, _VDBL_table ∗__t, const std::vector< _VDBL_-rowid > &_rs)
- _VDBL_tableid pop_table ()
- bool insert (const std::vector< _T_colspec > &_row)
- std::ostream & print_col (std::ostream &o, const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, bool &printed) const
- template<class _TR >
  bool get_raw_ptr (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _TR const ∗&r) const
- template<class _TR >
  bool get (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _TR &r) const
- virtual _VDBL_view ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_-view ∗v) VDBL_PURE_VIRTUALvirtual const std

Protected Types

- typedef _default_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col ∗ > default_-const_iterator
- typedef _col_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col ∗ > col_const_iterator
- typedef _row_iterator< _VDBL_row, const _VDBL_row &, const _VDBL_row ∗ > row_const_-iterator

Protected Member Functions

- std::triple< _VDBL_tableid, _VDBL_colid, void ∗ > _next_def_col (const _VDBL_tableid &_t, const _VDBL_colid &_c, void ∗_d) const
- void ∗ _copy_def_data (void ∗_d) const
- void ∗ _copy_col_data (void ∗_d) const
- void ∗ _copy_row_data (void ∗_d) const
- void made_change ()

  increment the `change` *counter.*

- unsigned int get_change_ctr () const

  read the change counter

Protected Attributes

- _V_rows _V_r
- _V_cols _V_c
- _V_colxref _V_cx

Friends

- class _VDBL_table

9.13.1 Detailed Description

This class implements a hierarchical view onto various tables.

Definition at line 49 of file vdbl_hrview.h.

9.13.2 Member Typedef Documentation

9.13.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_view::_T_colspec `[inherited]`

This is the description of one column

Definition at line 61 of file vdbl_view.h.

9.13.2.2 typedef _col_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col∗> _VDBL_view::col_const_iterator `[protected, inherited]`

const iterator over all columns

Definition at line 439 of file vdbl_view.h.

9.13.2.3 typedef _default_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col∗> _VDBL_view::default_const_iterator `[protected, inherited]`

const iterator over all default columns

Definition at line 301 of file vdbl_view.h.

9.13.2.4 typedef _row_iterator<_VDBL_row, const _VDBL_row&, const _VDBL_row∗> _VDBL_view::row_const_iterator `[protected, inherited]`

const iterator over all rows

Definition at line 568 of file vdbl_view.h.

9.13.3 Constructor & Destructor Documentation

9.13.3.1 _VDBL_hierarchicalview::_VDBL_hierarchicalview (const _VDBL_tableid & *__ti*, _VDBL_table ∗ *__t*, const _VDBL_context & *__c*, _V_enum *__en*) `[inline]`

standard constructor which initalizes the table and the tableid of the master table, the evaluation context, and the view type.

Definition at line 342 of file vdbl_hrview.h.

9.13.3.2 _VDBL_hierarchicalview::_VDBL_hierarchicalview (const _VDBL_tableid & *__ti*, _VDBL_table ∗ *__t*, const _VDBL_context & *__c*, _V_enum *__en*, const std::vector< _VDBL_rowid > & *_rs*) `[inline]`

standard constructor which initalizes the `table` and the `tableid` of the master table, the evaluation context, and the view type. In addition the vector `_rs` contains a list of rows, which should be visible in this view.

Definition at line 369 of file vdbl_hrview.h.

9.13.3.3 _VDBL_hierarchicalview::_VDBL_hierarchicalview (const _VDBL_hierarchicalview & *__v*) `[inline]`

copy constructor

Definition at line 395 of file vdbl_hrview.h.

9.13.3.4 virtual _VDBL_hierarchicalview::∼_VDBL_hierarchicalview () `[inline, virtual]`

standard destructor

Definition at line 404 of file vdbl_hrview.h.

9.13.4 Member Function Documentation

9.13.4.1 void∗ _VDBL_hierarchicalview::_copy_col_data (void ∗ *_d*) const `[inline, protected, virtual]`

This function copies the additional data needed by a _col_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next row for a `_row_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous row for a `_row_iterator`. This function destroys the additional data needed by a _row_iterator

Reimplemented from _VDBL_view.

Definition at line 290 of file vdbl_hrview.h.

9.13.4.2 void∗ _VDBL_hierarchicalview::_copy_def_data (void ∗ *_d*) const `[inline, protected, virtual]`

This function copies the additional data needed by a _default_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous column for a `_col_iterator`. This function destroys the additional data needed by a _col_iterator

Reimplemented from _VDBL_view.

Definition at line 233 of file vdbl_hrview.h.

9.13.4.3 void∗ _VDBL_hierarchicalview::_copy_row_data (void ∗ *_d*) const `[inline, protected, virtual]`

This function copies the additional data needed by a _row_iterator This is the fundamental class for iterators over all default columns, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented from _VDBL_view.

Definition at line 327 of file vdbl_hrview.h.

9.13.4.4 std::triple<_VDBL_tableid,_VDBL_colid,void*> _VDBL_hierarchicalview::_next_def_-col (const _VDBL_tableid & *_t*, const _VDBL_colid & *_c*, void * *_d*) const `[inline, protected, virtual]`

This virtual function has to be overloaded by the derived view classes, and it performs the step to the next default of a column a `_default_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous default of a column a `_default_iterator`. This function destroys the additional data needed by a _default_iterator

Reimplemented from _VDBL_view.

Definition at line 181 of file vdbl_hrview.h.

9.13.4.5 template<class _TR > bool _VDBL_hierarchicalview::get (const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, _TR & *r*) const `[inline]`

get the data from column `_ci` in row `_ri.second` of table `_ri.first`. The data stored in the column must be of type `_TR`.

Definition at line 651 of file vdbl_hrview.h.

9.13.4.6 template<class _TR > bool _VDBL_hierarchicalview::get_raw_ptr (const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, _TR const *& *r*) const `[inline]`

get a const ptr to the data from column `_ci` in row `_ri.second` of table `_ri.first`. The data stored in the column must be of type `_TR`. In this function no data copying is done. Note that this function returns a pointer to the columns raw data, so it can only be used to refer to constant columns.

Definition at line 632 of file vdbl_hrview.h.

9.13.4.7 bool _VDBL_hierarchicalview::insert (const std::vector< _T_colspec > & *_row*) `[inline]`

for now window views can only make changes in the top table in the list of tables

Definition at line 506 of file vdbl_hrview.h.

9.13.4.8 virtual _VDBL_view* _VDBL_view::new_copy () const `[inline, virtual, inherited]`

clone operation clone destructor return the column information for column `_C_n`. the return value is a reference to the `type_info` of the column's value type. The reference `_r` contains upon returning whether the function was successful, the column id, and the column flags. return a reference to the `type_-info` of the column's value type. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful. remove the row with it `_ri` from the table return whether the column `_C_n` is defined in the table return table id and column id of column `_C_n` return column name of column `_C_n` return a const reference to the column with column id `_ci` in row `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. Note that in this function the context for the column is NOT set. This method adds the row ids in `_rs` to the visible part of the view. This method removes the row ids in `_rs` from the visible part of the view. print the contents of the column with column id `_ci` in row `_ri.second` of table `ri.first`. If the row is empty and no default is defined, print nothing. return a const reference to the row with row id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return a const reference to the default column with column id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return iterator to first default column return iterator beyond last default column return iterator to first row return iterator beyond last row return iterator to first column return iterator beyond last column return the type of this view

Definition at line 587 of file vdbl_view.h.

9.13.4.9 _VDBL_tableid _VDBL_hierarchicalview::pop_table () `[inline]`

remove the topmost table from the view, and return its table id.

Definition at line 459 of file vdbl_hrview.h.

9.13.4.10 std::ostream& _VDBL_hierarchicalview::print_col (std::ostream & *o*, const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, bool & *printed*) const `[inline]`

print the contents od column `_ci` in row `_ri.second` of table `_ri.first`.

Definition at line 606 of file vdbl_hrview.h.

9.13.4.11 void _VDBL_hierarchicalview::push_table (const _VDBL_tableid & *__ti*, _VDBL_table ∗ *__t*, const std::vector< _VDBL_rowid > & *_rs*) `[inline]`

This pushes a new table onto the top of the hierarchical view stack. Additionally, a subset of the table's rows, which are visible in the view, can be specified.

Definition at line 434 of file vdbl_hrview.h.

9.13.4.12 void _VDBL_hierarchicalview::push_table (const _VDBL_tableid & *__ti*, _VDBL_table ∗ *__t*) `[inline]`

This pushes a new table onto the top of the hierarchical view stack.

Definition at line 409 of file vdbl_hrview.h.

9.13.5 Member Data Documentation

9.13.5.1 _V_cols _VDBL_hierarchicalview::_V_c [protected]

This contains all columns of the view

Definition at line 90 of file vdbl_hrview.h.

9.13.5.2 _V_colxref _VDBL_hierarchicalview::_V_cx [protected]

This is the cross reference: view col id -> <tableid, real col id>

Definition at line 94 of file vdbl_hrview.h.

9.13.5.3 _V_rows _VDBL_hierarchicalview::_V_r [protected]

This contains all rows of the view

Definition at line 86 of file vdbl_hrview.h.

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

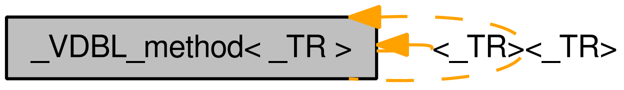
- vdbl_hrview.h

9.14 _VDBL_method< _TR > Class Template Reference

base class for methods usable in `_VDBL_mthdcol` columns.

`#include <vdbl_method.h>`

Inheritance diagram for _VDBL_method< _TR >:



Public Member Functions

- _VDBL_method ()
- _VDBL_method (const _VDBL_method &__m)
- virtual ~_VDBL_method ()
- virtual return_type operator() () const VDBL_PURE_VIRTUALvirtual return_type def() const VDBL_PURE_VIRTUALvirtual void setcontext(const context ∗_c

9.14.1 Detailed Description

template<class _TR> class _VDBL_method< _TR >

This is the base class, from which all methods should be derived that are used in `_VDBL_mthdcol`. Its virtual methods are those required from a method used for computing column names dynamically. Such a method is a function object with two additional methods described below.

Definition at line 56 of file vdbl_method.h.

9.14.2 Constructor & Destructor Documentation

9.14.2.1 template<class _TR> _VDBL_method< _TR >::_VDBL_method () `[inline]`

standard constructor

Definition at line 66 of file vdbl_method.h.

9.14.2.2 template<class _TR> _VDBL_method< _TR >::_VDBL_method (const _VDBL_method< _TR > & *__m*) `[inline]`

copy constructor

Definition at line 70 of file vdbl_method.h.

9.14.2.3 template<class _TR> virtual _VDBL_method< _TR >::~_VDBL_method () `[inline, virtual]`

standard destructor

Definition at line 75 of file vdbl_method.h.

9.14.3 Member Function Documentation

9.14.3.1 template<class _TR> virtual return_type _VDBL_method< _TR >::operator() () const `[virtual]`

compute the value compute the default value set the evaluation context and the evaluation row.

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

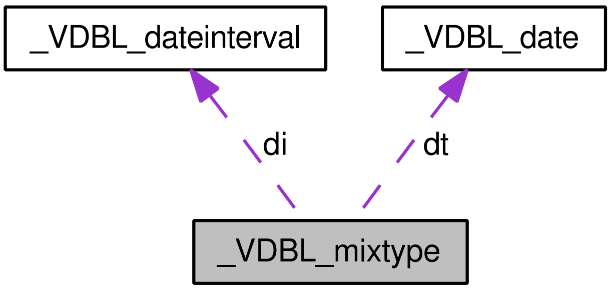
- vdbl_method.h

9.15 _VDBL_mixtype Class Reference

mixed type

```
#include <vdbl_alltype.h>
```

Collaboration diagram for _VDBL_mixtype:



Public Member Functions

- virtual ~_VDBL_mixtype ()

  destructor

- _VDBL_mixtype (const _VDBL_mixtype &__a)

  copy constructor

- _VDBL_mixtype & operator= (const _VDBL_mixtype &__a)

  assignment operator

- _VDBL_mixtype & clear ()

  deallocate all data and reset the `mixtype` *to its empty state*

- bool is_vector () const

  returns whether the mixtype stores a vector

- bool empty () const

  returns whether the mixtype is empty

- _VDBL_mixtype ()
- _VDBL_mixtype (bool __x)
- _VDBL_mixtype (int __x)
- _VDBL_mixtype (double __x)
- _VDBL_mixtype (unsigned int __x)
- _VDBL_mixtype (_VDBL_date __x)
- _VDBL_mixtype (_VDBL_dateinterval __x)
- _VDBL_mixtype (const char ∗__cp)
- _VDBL_mixtype (const std::string &__x)
- _VDBL_mixtype (const std::vector< bool > &__x)
- _VDBL_mixtype (const std::vector< int > &__x)
- _VDBL_mixtype (const std::vector< double > &__x)
- _VDBL_mixtype (const std::vector< unsigned int > &__x)

- _VDBL_mixtype & operator= (bool __x)
- _VDBL_mixtype & operator= (int __x)
- _VDBL_mixtype & operator= (double __x)

- _VDBL_mixtype & operator= (unsigned int __x)
- _VDBL_mixtype & operator= (const _VDBL_date &__x)
- _VDBL_mixtype & operator= (const _VDBL_dateinterval &__x)
- _VDBL_mixtype & operator= (const std::string &__x)
- _VDBL_mixtype & operator= (const char *__x)
- _VDBL_mixtype & operator= (const std::vector< bool > &__x)
- _VDBL_mixtype & operator= (const std::vector< int > &__x)
- _VDBL_mixtype & operator= (const std::vector< double > &__x)
- _VDBL_mixtype & operator= (const std::vector< unsigned int > &__x)

- bool nb () const
- int nn () const
- double nd () const
- unsigned int nu () const
- _VDBL_date dt () const
- _VDBL_dateinterval di () const
- std::string & s () const
- std::vector< bool > & b () const
- std::vector< int > & n () const
- std::vector< double > & d () const
- std::vector< unsigned int > & u () const

9.15.1 Detailed Description

This is an alternative definition of something like an all_type. It has a useful copy constructor and could be used as column type. The class can hold data of several basic types:

- bool
- int
- unsigned int
- double
- date
- dateinterval
- string
- vector<bool>
- vector<int>
- vector<unsigned int>
- vector<double>

Data is allocated and destroyed automatically.

Definition at line 327 of file vdbl_alltype.h.

9.15.2 Constructor & Destructor Documentation

9.15.2.1 _VDBL_mixtype::_VDBL_mixtype () `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 429 of file vdbl_alltype.h.

9.15.2.2 _VDBL_mixtype::_VDBL_mixtype (bool *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 430 of file vdbl_alltype.h.

9.15.2.3 _VDBL_mixtype::_VDBL_mixtype (int *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 432 of file vdbl_alltype.h.

9.15.2.4 _VDBL_mixtype::_VDBL_mixtype (double *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 434 of file vdbl_alltype.h.

9.15.2.5 _VDBL_mixtype::_VDBL_mixtype (unsigned int *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 436 of file vdbl_alltype.h.

9.15.2.6 _VDBL_mixtype::_VDBL_mixtype (_VDBL_date *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 438 of file vdbl_alltype.h.

9.15.2.7 _VDBL_mixtype::_VDBL_mixtype (_VDBL_dateinterval *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 440 of file vdbl_alltype.h.

9.15.2.8 _VDBL_mixtype::_VDBL_mixtype (const char * *__cp*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 442 of file vdbl_alltype.h.

9.15.2.9 _VDBL_mixtype::_VDBL_mixtype (const std::string & *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 444 of file vdbl_alltype.h.

9.15.2.10 _VDBL_mixtype::_VDBL_mixtype (const std::vector< bool > & *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 446 of file vdbl_alltype.h.

9.15.2.11 _VDBL_mixtype::_VDBL_mixtype (const std::vector< int > & *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 448 of file vdbl_alltype.h.

9.15.2.12 _VDBL_mixtype::_VDBL_mixtype (const std::vector< double > & *__x*) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 450 of file vdbl_alltype.h.

9.15.2.13 _VDBL_mixtype::_VDBL_mixtype (const std::vector< unsigned int > & __x) `[inline]`

For every type which can be stored there exists a constructor. The constructor without arguments produces an empty `_VDBL_mixtype`, and a `const char *` argument is stored as a `string`.

Definition at line 453 of file vdbl_alltype.h.

9.15.3 Member Function Documentation

9.15.3.1 std::vector<bool>& _VDBL_mixtype::b () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 569 of file vdbl_alltype.h.

9.15.3.2 std::vector<double>& _VDBL_mixtype::d () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 571 of file vdbl_alltype.h.

9.15.3.3 _VDBL_dateinterval _VDBL_mixtype::di () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 567 of file vdbl_alltype.h.

9.15.3.4 _VDBL_date _VDBL_mixtype::dt () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 566 of file vdbl_alltype.h.

9.15.3.5 std::vector<int>& _VDBL_mixtype::n () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 570 of file vdbl_alltype.h.

9.15.3.6 bool _VDBL_mixtype::nb () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 562 of file vdbl_alltype.h.

9.15.3.7 double _VDBL_mixtype::nd () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 564 of file vdbl_alltype.h.

9.15.3.8 int _VDBL_mixtype::nn () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 563 of file vdbl_alltype.h.

9.15.3.9 unsigned int _VDBL_mixtype::nu () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 565 of file vdbl_alltype.h.

9.15.3.10 _VDBL_mixtype& _VDBL_mixtype::operator= (const std::vector< unsigned int > & *__x*) `[inline]`

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 531 of file vdbl_alltype.h.

9.15.3.11 _VDBL_mixtype& _VDBL_mixtype::operator= (const std::vector< double > & *__x*) `[inline]`

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 525 of file vdbl_alltype.h.

9.15.3.12 _VDBL_mixtype& _VDBL_mixtype::operator= (const std::vector< int > & __*x*) [inline]

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 519 of file vdbl_alltype.h.

9.15.3.13 _VDBL_mixtype& _VDBL_mixtype::operator= (const std::vector< bool > & __*x*) [inline]

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 513 of file vdbl_alltype.h.

9.15.3.14 _VDBL_mixtype& _VDBL_mixtype::operator= (const char * __*x*) [inline]

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 507 of file vdbl_alltype.h.

9.15.3.15 _VDBL_mixtype& _VDBL_mixtype::operator= (const std::string & __*x*) [inline]

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 501 of file vdbl_alltype.h.

9.15.3.16 _VDBL_mixtype& _VDBL_mixtype::operator= (const _VDBL_dateinterval & __*x*) [inline]

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 496 of file vdbl_alltype.h.

9.15.3.17 _VDBL_mixtype& _VDBL_mixtype::operator= (const _VDBL_date & __*x*) [inline]

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 491 of file vdbl_alltype.h.

9.15.3.18 _VDBL_mixtype& _VDBL_mixtype::operator= (unsigned int __*x*) `[inline]`

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 486 of file vdbl_alltype.h.

9.15.3.19 _VDBL_mixtype& _VDBL_mixtype::operator= (double __*x*) `[inline]`

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 481 of file vdbl_alltype.h.

9.15.3.20 _VDBL_mixtype& _VDBL_mixtype::operator= (int __*x*) `[inline]`

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 476 of file vdbl_alltype.h.

9.15.3.21 _VDBL_mixtype& _VDBL_mixtype::operator= (bool __*x*) `[inline]`

For every type which can be stored there exists an assignment operator, and a `const char *` argument is stored as a `string`.

Definition at line 471 of file vdbl_alltype.h.

9.15.3.22 std::string& _VDBL_mixtype::s () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 568 of file vdbl_alltype.h.

9.15.3.23 std::vector<unsigned int>& _VDBL_mixtype::u () const `[inline]`

Retrieve the value of the appropriate type. Note, that no internal type checking is done, whatsoever. So you should have a good idea what is stored in the mixtype, if you want to call one of the retrieval routines.

Definition at line 572 of file vdbl_alltype.h.

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

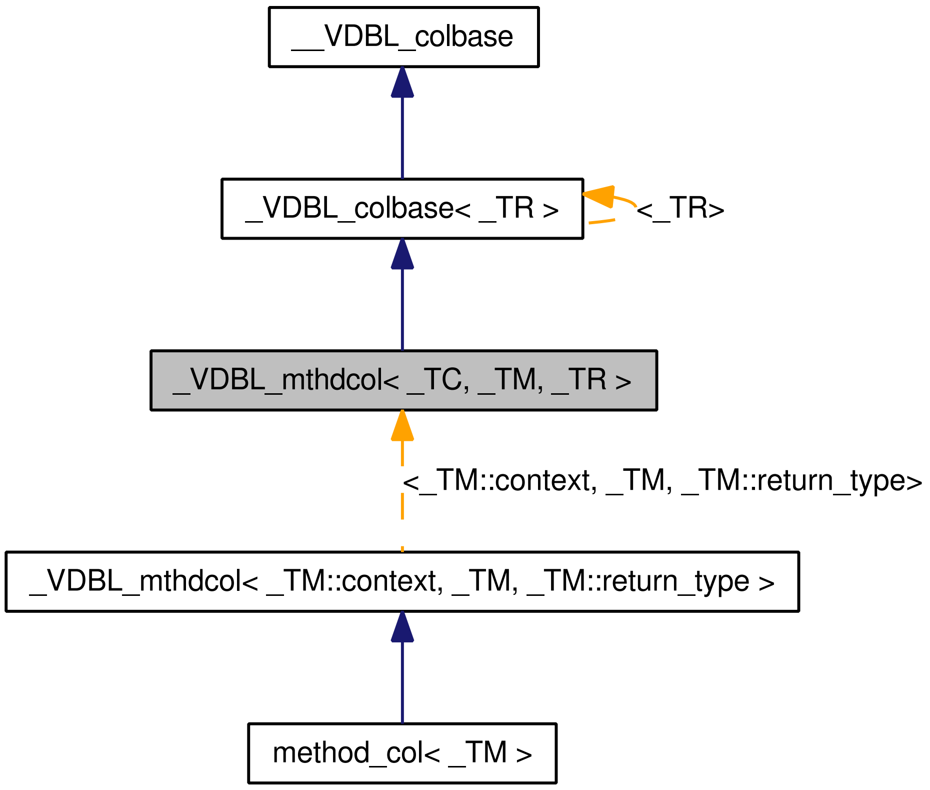
- vdbl_alltype.h

9.16 _VDBL_mthdcol< _TC, _TM, _TR > Class Template Reference

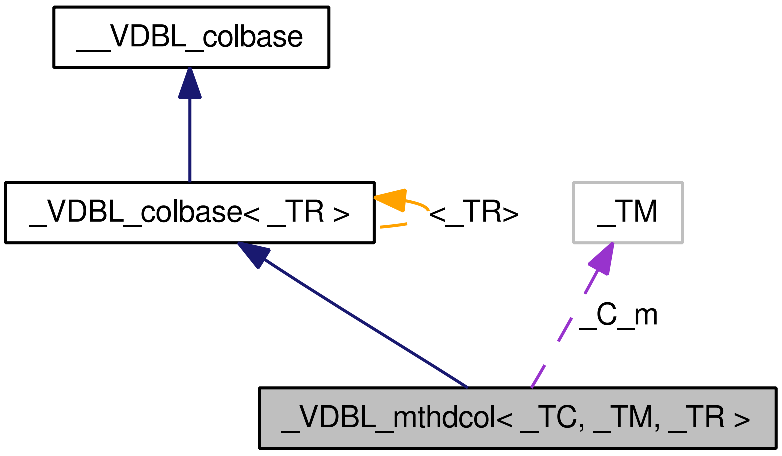
generic column class for methods

`#include <vdbl_col.h>`

Inheritance diagram for _VDBL_mthdcol< _TC, _TM, _TR >:


__VDBL_colbase
_VDBL_colbase< _TR >
<_TR>
_VDBL_mthdcol< _TC, _TM, _TR >
<_TM::context, _TM, _TM::return_type>
_VDBL_mthdcol< _TM::context, _TM, _TM::return_type >
method_col< _TM >

Collaboration diagram for _VDBL_mthdcol< _TC, _TM, _TR >:



Public Member Functions

- _VDBL_mthdcol (const method &__m)
- _Self ∗ new_copy () const
- virtual void setcontext (const context ∗_c, const _VDBL_row ∗_r)
- void get (type &c) const

  the function object provides us with the retrieval method

- void get_ptr (type const ∗&c) const

  there is no way to get a pointer to the method's result properly

- void def (type &d) const

  the default value might be different, and might be computed differently

- std::ostream & print_contents (std::ostream &o) const
- virtual void def_copy (return_type ∗&d) const
- virtual void def_copy (_VDBL_alltype_base ∗&v) const
- virtual void get_copy (_VDBL_alltype_base ∗&v) const
- virtual const std::type_info & return_type_id () const

  - _VDBL_mthdcol ()

    > `_C_m` *holds the function object used*

  - _VDBL_mthdcol (const _Self &__c)

    > `_C_m` *holds the function object used*

  - virtual ∼_VDBL_mthdcol ()

    > `_C_m` *holds the function object used*

9.16.1 Detailed Description

template<class _TC, class _TM, class _TR> class _VDBL_mthdcol< _TC, _TM, _TR >

`_VDBL_mthdcol` is the generic column class for computed values. It allows to define a method, which is called within the given context together with a reference to the row, whenever a column is accessed.

Definition at line 505 of file vdbl_col.h.

9.16.2 Constructor & Destructor Documentation

9.16.2.1 template<class _TC, class _TM, class _TR> _VDBL_mthdcol< _TC, _TM, _TR >::_VDBL_mthdcol () `[inline]`

standard constructor, copy constructor, and destructor

Definition at line 524 of file vdbl_col.h.

9.16.2.2 template<class _TC, class _TM, class _TR> _VDBL_mthdcol< _TC, _TM, _TR >::_VDBL_mthdcol (const _Self & *__c*) `[inline]`

standard constructor, copy constructor, and destructor

Definition at line 525 of file vdbl_col.h.

9.16.2.3 template<class _TC, class _TM, class _TR> virtual _VDBL_mthdcol< _TC, _TM, _TR >::~_VDBL_mthdcol () `[inline, virtual]`

standard constructor, copy constructor, and destructor

Definition at line 526 of file vdbl_col.h.

9.16.2.4 template<class _TC, class _TM, class _TR> _VDBL_mthdcol< _TC, _TM, _TR >::_VDBL_mthdcol (const method & *__m*) `[inline]`

constructor for explicitely setting the method

Definition at line 532 of file vdbl_col.h.

9.16.3 Member Function Documentation

9.16.3.1 virtual void _VDBL_colbase< _TR >::def_copy (_VDBL_alltype_base *& *v*) const `[inline, virtual, inherited]`

This version of `get_copy` returns a copy of the columns default value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Definition at line 210 of file vdbl_col.h.

9.16.3.2 virtual void _VDBL_colbase< _TR >::def_copy (return_type ∗& *d*) const `[inline, virtual, inherited]`

This function returns a pointer to a copy of the column's default value. The copy of the value is allocated by `new`. It has to be `delete`d by the user to avoid memory leaks.

Definition at line 188 of file vdbl_col.h.

9.16.3.3 virtual void _VDBL_colbase< _TR >::get_copy (_VDBL_alltype_base ∗& *v*) const `[inline, virtual, inherited]`

This version of `get_copy` returns a copy of the columns value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Definition at line 197 of file vdbl_col.h.

9.16.3.4 template<class _TC, class _TM, class _TR> _Self∗ _VDBL_mthdcol< _TC, _TM, _TR >::new_copy () const `[inline, virtual]`

`new_copy` is the clone operation for copy-constructor overloading. `setcontext` sets the context for value retrieval.

Reimplemented from _VDBL_colbase< _TR >.

Definition at line 534 of file vdbl_col.h.

9.16.3.5 template<class _TC, class _TM, class _TR> std::ostream& _VDBL_mthdcol< _TC, _TM, _TR >::print_contents (std::ostream & *o*) const `[inline, virtual]`

This function is needed for the operator<< for columns of type `return_type`.

Reimplemented from _VDBL_colbase< _TR >.

Definition at line 561 of file vdbl_col.h.

9.16.3.6 virtual const std::type_info& _VDBL_colbase< _TR >::return_type_id () const `[inline, virtual, inherited]`

This function returns the `type_info` of the column type. This information is used during run-time type checking.

Definition at line 221 of file vdbl_col.h.

9.16.3.7 template<class _TC, class _TM, class _TR> virtual void _VDBL_mthdcol< _TC, _TM, _TR >::setcontext (const context * *_c*, const _VDBL_row * *_r*) `[inline, virtual]`

for setting the context, the setcontext method of the function object is used.

Definition at line 540 of file vdbl_col.h.

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

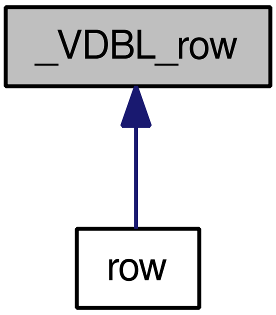
- vdbl_col.h

9.17 _VDBL_row Class Reference

row class

`#include <vdbl_row.h>`

Inheritance diagram for _VDBL_row:



Public Member Functions

- _VDBL_row ()
- _VDBL_row (const _VDBL_row &__r)
- virtual ∼_VDBL_row ()
- const _VDBL_col & get_col (const _VDBL_colid &_id, bool &error) const
- _VDBL_col & get_col (const _VDBL_colid &_id, bool &error)
- bool has_col (const _VDBL_colid &_id) const
- bool insert (const _VDBL_colid &_id, const _VDBL_col &_col)
- bool drop (const _VDBL_colid &_id)
- void update (const _VDBL_colid &_id, const _VDBL_col &_col)

9.17.1 Detailed Description

This class implements rows of a table as a map column id -> column

Definition at line 54 of file vdbl_row.h.

9.17.2 Constructor & Destructor Documentation

9.17.2.1 _VDBL_row::_VDBL_row () `[inline]`

standard constructor which optionally initializes the global ACL entry

Definition at line 67 of file vdbl_row.h.

9.17.2.2 _VDBL_row::_VDBL_row (const _VDBL_row & *__r*) `[inline]`

copy constructor

Definition at line 71 of file vdbl_row.h.

9.17.2.3 virtual _VDBL_row::∼_VDBL_row () `[inline, virtual]`

standard destructor

Definition at line 75 of file vdbl_row.h.

9.17.3 Member Function Documentation

9.17.3.1 bool _VDBL_row::drop (const _VDBL_colid & *_id*) `[inline]`

remove the column with id `_id` from this row. Return `true` if erasing was successful, and `false` if the column does not exist.

Definition at line 149 of file vdbl_row.h.

9.17.3.2 _VDBL_col& _VDBL_row::get_col (const _VDBL_colid & *_id*, bool & *error*) `[inline]`

get a reference to the column with id `_id`. If the column existed, `error` will be `false`, otherwise `error` will be `true`.

Definition at line 103 of file vdbl_row.h.

9.17.3.3 const _VDBL_col& _VDBL_row::get_col (const _VDBL_colid & *_id*, bool & *error*) const `[inline]`

get a const reference to the column with id `_id`. If the column existed, `error` will be `false`, otherwise `error` will be `true`.

Definition at line 82 of file vdbl_row.h.

9.17.3.4 bool _VDBL_row::has_col (const _VDBL_colid & *_id*) const `[inline]`

return whether a column with id `_id` exists in this row.

Definition at line 122 of file vdbl_row.h.

9.17.3.5 bool _VDBL_row::insert (const _VDBL_colid & *_id*, const _VDBL_col & *_col*) `[inline]`

insert the new column `_col` with id `_id` in this row. If this id exists, return `false`, otherwise return `true`.

Definition at line 132 of file vdbl_row.h.

9.17.3.6 void _VDBL_row::update (const _VDBL_colid & *_id*, const _VDBL_col & *_col*) `[inline]`

update the column with id `_id` with the value `_col`. If the column does not yet exist, insert it. Otherwise, change its value.

Definition at line 166 of file vdbl_row.h.

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

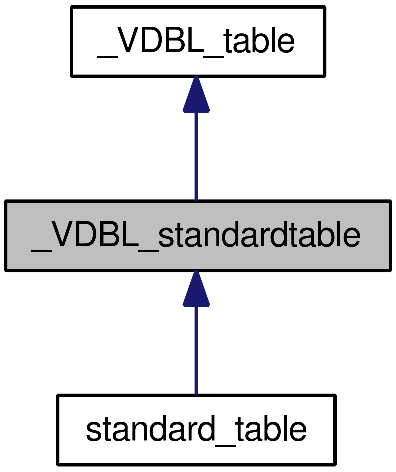
- vdbl_row.h

9.18 _VDBL_standardtable Class Reference

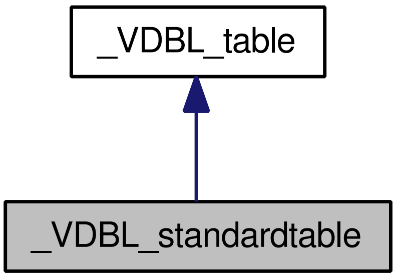
standard table in databases, constructed from rows and columns

`#include <vdbl_table.h>`

Inheritance diagram for _VDBL_standardtable:



Collaboration diagram for _VDBL_standardtable:



Public Types

- typedef std::pair< std::string, _VDBL_col > _T_colspec
- typedef std::pair< const std::string ∗, const _VDBL_col ∗ > _T_ptrcolspec
- typedef _Base::col_const_iterator col_const_iterator
- typedef _row_iterator< _VDBL_rowid, const _VDBL_rowid &, const _VDBL_rowid ∗ > row_-const_iterator

Public Member Functions

- const std::type_info & get_colinfo (const std::string &_C_n, std::triple< bool, _VDBL_colid, _-VDBL_colflags > &_r) const
- _VDBL_colid get_col_id (const std::string &_C_n) const
- std::string get_col_name (const _VDBL_colid &_C_id) const
- _VDBL_standardtable ()
- _VDBL_standardtable (const _VDBL_standardtable &__t)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 >
  _VDBL_standardtable (const __SequenceCtr< std::triple< std::string, _VDBL_col, _VDBL_-colflags >, Allocator1 > &__cc)
- virtual ∼_VDBL_standardtable ()
- virtual _VDBL_table ∗ new_copy ()
- virtual void destroy_copy (_VDBL_table ∗t)
- template<class _VALclass >
  bool retrieve (const _VDBL_rowid &_r, const _VDBL_colid &_c, const _VDBL_context ∗_ctx, _VALclass ∗&_val) const
- std::pair< std::string, _VDBL_colid > _next_col (const std::pair< std::string, _VDBL_colid > &_-ci) const
- _VDBL_rowid _next_row (const _VDBL_rowid &_ci) const
- virtual _VDBL_table ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_-table ∗t) VDBL_PURE_VIRTUALvirtual bool add_col(const std
- template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn >
  bool insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > &_rows)

Protected Member Functions

- void made_change ()
  - _VDBL_colid get_colid ()
  - _VDBL_rowid get_rowid ()

Friends

- class _VDBL_view

9.18.1 Detailed Description

This is the class describing standard tables as they usually appear in databases, constructed from rows and columns.

Definition at line 573 of file vdbl_table.h.

9.18.2 Member Typedef Documentation

9.18.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_standardtable::_T_colspec

specifier of one column, a pair of column name (`string`) and entry (`_VDBL_col`).

Reimplemented from _VDBL_table.

Definition at line 591 of file vdbl_table.h.

9.18.2.2 typedef std::pair<const std::string*,const _VDBL_col*> _VDBL_standardtable::_T_-ptrcolspec

specifier of pointers to one column, a pair of column name (`string*`) and entry (`_VDBL_col*`).

Reimplemented from _VDBL_table.

Definition at line 596 of file vdbl_table.h.

9.18.2.3 typedef _Base::col_const_iterator _VDBL_standardtable::col_const_iterator

const iterator over all columns

Reimplemented from _VDBL_table.

Definition at line 601 of file vdbl_table.h.

9.18.2.4 typedef _row_iterator<_VDBL_rowid, const _VDBL_rowid&, const _VDBL_rowid*> _VDBL_table::row_const_iterator `[inherited]`

const iterator over all rows

Definition at line 321 of file vdbl_table.h.

9.18.3 Constructor & Destructor Documentation

9.18.3.1 _VDBL_standardtable::_VDBL_standardtable () `[inline]`

standard constructor which optionally initializes the global ACL entry

Definition at line 663 of file vdbl_table.h.

9.18.3.2 _VDBL_standardtable::_VDBL_standardtable (const _VDBL_standardtable & *__t*) `[inline]`

copy constructor

Definition at line 667 of file vdbl_table.h.

9.18.3.3 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 > _VDBL_standardtable::_VDBL_standardtable (const __SequenceCtr< std::triple< std::string, _VDBL_col, _VDBL_colflags >, Allocator1 > & *__cc*) `[inline]`

constructor which builds a table from a list of column definitions. This list can be contained in any sequential STL container.

Definition at line 677 of file vdbl_table.h.

9.18.3.4 virtual _VDBL_standardtable::∼_VDBL_standardtable () `[inline, virtual]`

standard destructor

Definition at line 707 of file vdbl_table.h.

9.18.4 Member Function Documentation

9.18.4.1 std::pair<std::string,_VDBL_colid> _VDBL_standardtable::_next_col (const std::pair< std::string, _VDBL_colid > & *_ci*) const `[inline, virtual]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous column for a `_col_iterator`. return iterator to first column return iterator beyond last column This is the fundamental class for iterators over all rows, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented from _VDBL_table.

Definition at line 1174 of file vdbl_table.h.

9.18.4.2 _VDBL_rowid _VDBL_standardtable::_next_row (const _VDBL_rowid & *_ci*) const `[inline, virtual]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next row for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous row for a `_col_iterator`. return iterator to first row return iterator beyond last row construct a new copy of a row iterator standard constructor

Reimplemented from _VDBL_table.

Definition at line 1209 of file vdbl_table.h.

9.18.4.3 virtual void _VDBL_standardtable::destroy_copy (_VDBL_table ∗ *t*) `[inline, virtual]`

clone destructor method

Definition at line 717 of file vdbl_table.h.

9.18.4.4 _VDBL_colid _VDBL_standardtable::get_col_id (const std::string & *_C_n*) const `[inline]`

return the column id of column `_C_n`

Definition at line 637 of file vdbl_table.h.

9.18.4.5 std::string _VDBL_standardtable::get_col_name (const _VDBL_colid & *_C_id*) const `[inline]`

return the column name of column `_C_id`

Definition at line 650 of file vdbl_table.h.

9.18.4.6 _VDBL_colid _VDBL_table::get_colid () `[inline, protected, inherited]`

generate new unique id's for rows and columns

Definition at line 99 of file vdbl_table.h.

9.18.4.7 const std::type_info& _VDBL_standardtable::get_colinfo (const std::string & *_C_n*, std::triple< bool, _VDBL_colid, _VDBL_colflags > & *_r*) const `[inline, virtual]`

return the column information for column _C_n. the return value is a referenc to the type_info of the column's value type. The reference _r contains upon returning whether the function was successful, the column id, and the column flags. what was the id of the last change to the table

Reimplemented from _VDBL_table.

Definition at line 614 of file vdbl_table.h.

9.18.4.8 _VDBL_rowid _VDBL_table::get_rowid () [inline, protected, inherited]

generate new unique id's for rows and columns

Definition at line 100 of file vdbl_table.h.

9.18.4.9 template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn > bool _VDBL_table::insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > & *_rows*) [inline, inherited]

insert a many new rows of specifications _rows into the table. The list of rows can be contained in any sequential STL container, which holds any other sequential STL container of column entries. The function returns true, if inserting was successful for all rows.

Definition at line 473 of file vdbl_table.h.

9.18.4.10 void _VDBL_table::made_change () [inline, protected, inherited]

increment the last_change counter.

Definition at line 106 of file vdbl_table.h.

9.18.4.11 virtual _VDBL_table∗ _VDBL_table::new_copy () const [inline, virtual, inherited]

clone method clone destructor method add a new column of name _C_n, which contains __c, and has column flags __f. The function returns true, if adding the column was successful. modify the column of name _C_n, to new contents __c, and new column flags __f. The function returns true, if modifying was successful. modify the contents of column of name _C_n The function returns true, if modifying was successful. modify the column flags of column of name _C_n The function returns true, if modifying was successful. remove the column _C_n from the table The function returns true, if erasing was successful. rename the column _C_old to new name _C_new. The function returns true, if renaming was successful. insert a new row of specification _row into the table, and return the row id of the newly created row in _r. The function returns true, if inserting was successful. insert a new row of specification _row into the table. The function returns true, if inserting was successful. insert a new row of specification _row into the table, and return the row id of the newly created row in _r. Take any sequential STL container to hold the row entries of the column. The function returns true, if inserting was successful.

Definition at line 380 of file vdbl_table.h.

9.18.4.12 virtual _VDBL_table* _VDBL_standardtable::new_copy () `[inline, virtual]`

clone method

Reimplemented in standard_table.

Definition at line 712 of file vdbl_table.h.

9.18.4.13 template<class _VALclass > bool _VDBL_standardtable::retrieve (const _VDBL_rowid & *_r*, const _VDBL_colid & *_c*, const _VDBL_context * *_ctx*, _VALclass *& *_val*) const `[inline]`

retrieve the value of column (id = `_c`) in row (id = `_r`) under evaluation context `_ctx` to `_val`. Return whether retrieval was successful. If there is no defined value in row `_r` for column `_c`, then return the default value of column `_c`, if defined.

Definition at line 1105 of file vdbl_table.h.

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

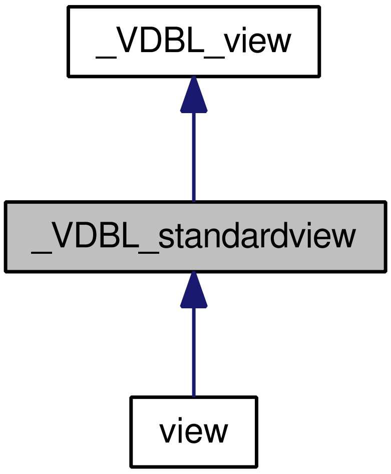
- vdbl_table.h

9.19 _VDBL_standardview Class Reference

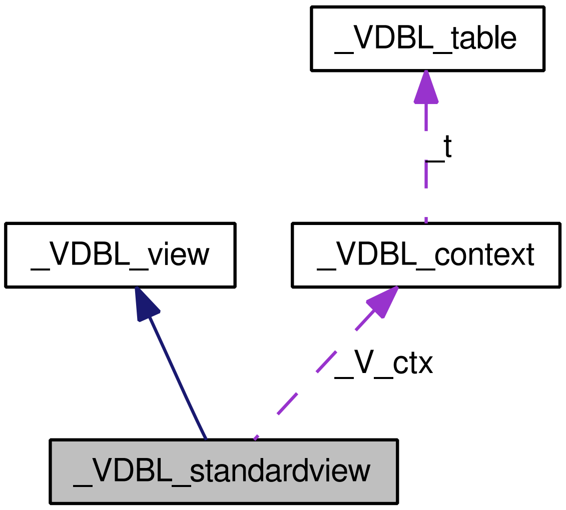
standard view onto **one** table

`#include <vdbl_stview.h>`

Inheritance diagram for _VDBL_standardview:



Collaboration diagram for _VDBL_standardview:



Public Types

- typedef _Base::default_const_iterator defaults_const_iterator

  iterator over all default columns

- typedef std::pair< std::string, _VDBL_col > _T_colspec

Public Member Functions

- _VDBL_standardview (const _VDBL_tableid &__ti, _VDBL_table ∗__t, const _VDBL_context &__c, _V_enum __e)
- _VDBL_standardview (const _VDBL_tableid &__ti, _VDBL_table ∗__t, const _VDBL_context &__c, _V_enum __e, const std::vector< _VDBL_rowid > &_rs)
- _VDBL_standardview (const _VDBL_standardview &__v)
- virtual ∼_VDBL_standardview ()
- virtual _VDBL_view ∗ new_copy ()
- virtual void destroy_copy (_VDBL_view ∗v)
- bool insert (const std::vector< _T_colspec > &_row)
- bool add_visible (const std::vector< _VDBL_rowid > &_rs)
- bool rm_visible (const std::vector< _VDBL_rowid > &_rs)
- std::ostream & print_col (std::ostream &o, const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, bool &printed) const
- template<class _TR >
  bool get (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _TR &r) const
- template<class _TR >
  bool get_raw_ptr (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _TR const ∗&r) const
- virtual _VDBL_view ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_-view ∗v) VDBL_PURE_VIRTUALvirtual const std

Protected Types

- typedef _default_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col * > default_-const_iterator
- typedef _col_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col * > col_const_iterator
- typedef _row_iterator< _VDBL_row, const _VDBL_row &, const _VDBL_row * > row_const_-iterator

Protected Member Functions

- std::triple< _VDBL_tableid, _VDBL_colid, void * > _next_def_col (const _VDBL_tableid &_t, const _VDBL_colid &_c, void *_d) const
- void * _copy_def_data (void *_d) const
- void * _copy_col_data (void *_d) const
- void * _copy_row_data (void *_d) const
- void made_change ()

  increment the `change` *counter.*

- unsigned int get_change_ctr () const

  read the change counter

Protected Attributes

- _V_rows _V_r
- _V_cols _V_c

Friends

- class _VDBL_table

9.19.1 Detailed Description

This class implements views onto a single table.

Definition at line 49 of file vdbl_stview.h.

9.19.2 Member Typedef Documentation

9.19.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_view::_T_colspec `[inherited]`

This is the description of one column

Definition at line 61 of file vdbl_view.h.

9.19.2.2 typedef _col_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col*> _VDBL_view::col_const_iterator `[protected, inherited]`

const iterator over all columns

Definition at line 439 of file vdbl_view.h.

9.19.2.3 typedef _default_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col*> _VDBL_view::default_const_iterator `[protected, inherited]`

const iterator over all default columns

Definition at line 301 of file vdbl_view.h.

9.19.2.4 typedef _row_iterator<_VDBL_row, const _VDBL_row&, const _VDBL_row*> _VDBL_view::row_const_iterator `[protected, inherited]`

const iterator over all rows

Definition at line 568 of file vdbl_view.h.

9.19.3 Constructor & Destructor Documentation

9.19.3.1 _VDBL_standardview::_VDBL_standardview (const _VDBL_tableid & *__ti*, _VDBL_table * *__t*, const _VDBL_context & *__c*, _V_enum *__e*) `[inline]`

standard constructor which initalizes the `table` and the `tableid`, the evaluation context, and the view type.

Definition at line 327 of file vdbl_stview.h.

9.19.3.2 _VDBL_standardview::_VDBL_standardview (const _VDBL_tableid & *__ti*, _VDBL_table * *__t*, const _VDBL_context & *__c*, _V_enum *__e*, const std::vector< _VDBL_rowid > & *_rs*) `[inline]`

standard constructor which initalizes the `table` and the `tableid`, the evaluation context, and the view type. In addition the vector `_rs` contains a list of rows, which should be visible in this view.

Definition at line 353 of file vdbl_stview.h.

9.19.3.3 _VDBL_standardview::_VDBL_standardview (const _VDBL_standardview & *__v*) `[inline]`

copy constructor

Definition at line 372 of file vdbl_stview.h.

9.19.3.4 virtual _VDBL_standardview::∼_VDBL_standardview () `[inline, virtual]`

standard destructor

Definition at line 382 of file vdbl_stview.h.

9.19.4 Member Function Documentation

9.19.4.1 void∗ _VDBL_standardview::_copy_col_data (void ∗ *_d*) const `[inline, protected, virtual]`

This function copies the additional data needed by a _col_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next row for a `_row_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous row for a `_row_iterator`. This function destroys the additional data needed by a _row_iterator

Reimplemented from _VDBL_view.

Definition at line 196 of file vdbl_stview.h.

9.19.4.2 void∗ _VDBL_standardview::_copy_def_data (void ∗ *_d*) const `[inline, protected, virtual]`

This function copies the additional data needed by a _default_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous column for a `_col_iterator`. This function destroys the additional data needed by a _col_iterator

Reimplemented from _VDBL_view.

Definition at line 146 of file vdbl_stview.h.

9.19.4.3 void∗ _VDBL_standardview::_copy_row_data (void ∗ *_d*) const `[inline, protected, virtual]`

This function copies the additional data needed by a _row_iterator This is the fundamental class for iterators over all default columns, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented from _VDBL_view.

Definition at line 294 of file vdbl_stview.h.

9.19.4.4 std::triple<_VDBL_tableid,_VDBL_colid,void*> _VDBL_standardview::_next_def_col (const _VDBL_tableid & *_t*, const _VDBL_colid & *_c*, void * *_d*) const `[inline, protected, virtual]`

This virtual function has to be overloaded by the derived view classes, and it performs the step to the next default of a column a `_default_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous default of a column a `_default_iterator`. This function destroys the additional data needed by a _default_iterator

Reimplemented from _VDBL_view.

Definition at line 102 of file vdbl_stview.h.

9.19.4.5 bool _VDBL_standardview::add_visible (const std::vector< _VDBL_rowid > & *_rs*) `[inline]`

This method adds the row ids in `_rs` to the visible part of the view.

Definition at line 497 of file vdbl_stview.h.

9.19.4.6 virtual void _VDBL_standardview::destroy_copy (_VDBL_view * *v*) `[inline, virtual]`

clone destructor

Definition at line 391 of file vdbl_stview.h.

9.19.4.7 template<class _TR > bool _VDBL_standardview::get (const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, _TR & *r*) const `[inline]`

get the data from column `_ci` in row `_ri.second` of table `_ri.first`. The data stored in the column must be of type `_TR`.

Definition at line 559 of file vdbl_stview.h.

9.19.4.8 template<class _TR > bool _VDBL_standardview::get_raw_ptr (const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, _TR const *& *r*) const `[inline]`

get a const ptr to the data from column `_ci` in row `_ri.second` of table `_ri.first`. The data stored in the column must be of type `_TR`. In this function no data copying is done. Note that this function returns a pointer to the columns raw data, so it can only be used to refer to constant columns.

Definition at line 581 of file vdbl_stview.h.

9.19.4.9 bool _VDBL_standardview::insert (const std::vector< _T_colspec > & *_row*) [inline]

for now window views can only have one table in the list of tables

Definition at line 412 of file vdbl_stview.h.

9.19.4.10 virtual _VDBL_view∗ _VDBL_view::new_copy () const [inline, virtual, inherited]

clone operation clone destructor return the column information for column `_C_n`. the return value is a reference to the `type_info` of the column's value type. The reference `_r` contains upon returning whether the function was successful, the column id, and the column flags. return a reference to the `type_-info` of the column's value type. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful. remove the row with it `_ri` from the table return whether the column `_C_n` is defined in the table return table id and column id of column `_C_n` return column name of column `_C_n` return a const reference to the column with column id `_ci` in row `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. Note that in this function the context for the column is NOT set. This method adds the row ids in `_rs` to the visible part of the view. This method removes the row ids in `_rs` from the visible part of the view. print the contents of the column with column id `_ci` in row `_ri.second` of table `ri.first`. If the row is empty and no default is defined, print nothing. return a const reference to the row with row id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return a const reference to the default column with column id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return iterator to first default column return iterator beyond last default column return iterator to first row return iterator beyond last row return iterator to first column return iterator beyond last column return the type of this view

Definition at line 587 of file vdbl_view.h.

9.19.4.11 virtual _VDBL_view∗ _VDBL_standardview::new_copy () [inline, virtual]

clone operation

Reimplemented in view.

Definition at line 387 of file vdbl_stview.h.

9.19.4.12 std::ostream& _VDBL_standardview::print_col (std::ostream & *o*, const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, bool & *printed*) const [inline]

print the contents of column `_ci` in row `_ri.second` of table `_ri.first`.

Definition at line 536 of file vdbl_stview.h.

9.19.4.13 bool _VDBL_standardview::rm_visible (const std::vector< _VDBL_rowid > & *_rs*) `[inline]`

This method removes the row ids in `_rs` from the visible part of the view.

Definition at line 514 of file vdbl_stview.h.

9.19.5 Member Data Documentation

9.19.5.1 _V_cols _VDBL_standardview::_V_c `[protected]`

This contains all columns of the view

Definition at line 88 of file vdbl_stview.h.

9.19.5.2 _V_rows _VDBL_standardview::_V_r `[protected]`

This contains all rows of the view unless it is a window view

Definition at line 84 of file vdbl_stview.h.

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

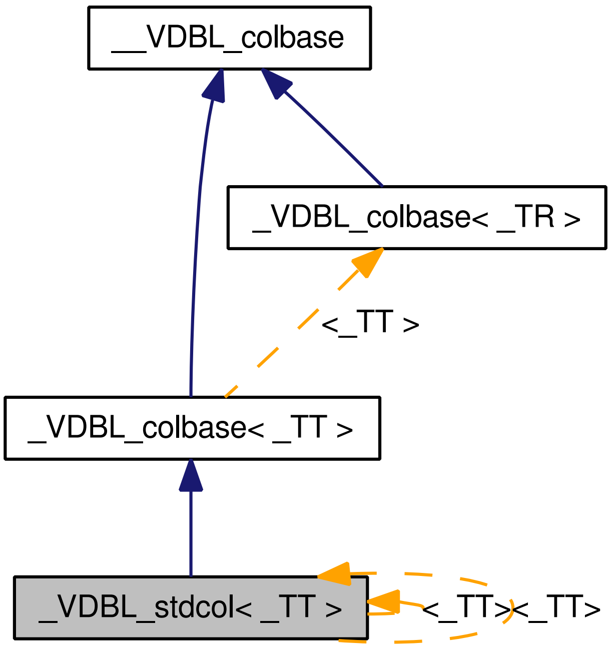
- vdbl_stview.h

9.20 _VDBL_stdcol< _TT > Class Template Reference

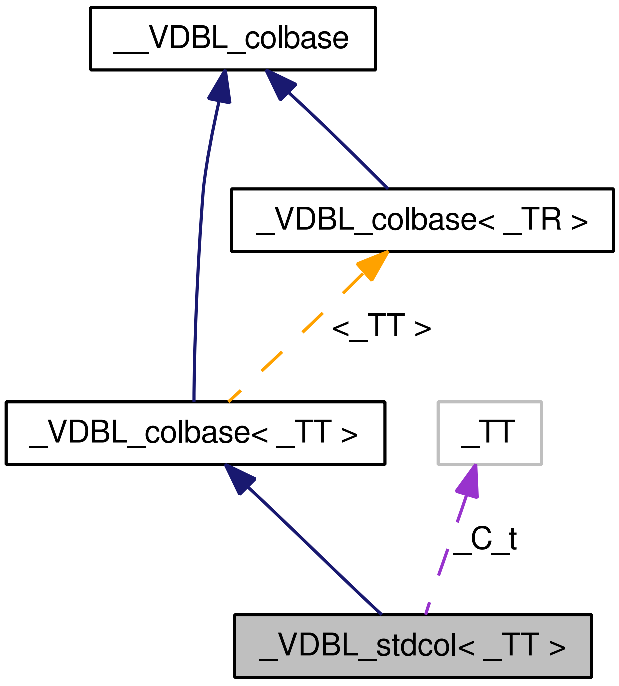
generic column class for constant values

`#include <vdbl_col.h>`

Inheritance diagram for _VDBL_stdcol< _TT >:



Collaboration diagram for _VDBL_stdcol< _TT >:



Public Member Functions

- _VDBL_stdcol (const type &__t)
- _Self ∗ new_copy () const
- void set (const _Self &_p)
- void setcontext (const context ∗_c, const _VDBL_row ∗_r)
- void get (type &c) const
- void def (type &d) const
- void get_copy (_VDBL_alltype_base ∗&v) const
- void def_copy (_VDBL_alltype_base ∗&v) const
- void set (const type &__t)
- void set_default (const type &__t)
- const type & get_val () const
- std::ostream & print_contents (std::ostream &o) const
- virtual void def_copy (return_type ∗&d) const
- virtual const std::type_info & return_type_id () const

  - _VDBL_stdcol ()

    > `_C_t` *holds the data*

  - _VDBL_stdcol (const _Self &__c)

    > `_C_t` *holds the data*

  - virtual ∼_VDBL_stdcol ()

    > `_C_t` *holds the data*

9.20.1 Detailed Description

template<class _TT> class _VDBL_stdcol< _TT >

`_VDBL_stdcol` is the generic column class for constant values.

Definition at line 406 of file vdbl_col.h.

9.20.2 Constructor & Destructor Documentation

9.20.2.1 template<class _TT> _VDBL_stdcol< _TT >::_VDBL_stdcol () `[inline]`

standard constructor, copy constructor, destructor

Definition at line 423 of file vdbl_col.h.

9.20.2.2 template<class _TT> _VDBL_stdcol< _TT >::_VDBL_stdcol (const _Self & *__c*) `[inline]`

standard constructor, copy constructor, destructor

Definition at line 424 of file vdbl_col.h.

9.20.2.3 template<class _TT> virtual _VDBL_stdcol< _TT >::~_VDBL_stdcol () `[inline, virtual]`

standard constructor, copy constructor, destructor

Definition at line 425 of file vdbl_col.h.

9.20.2.4 template<class _TT> _VDBL_stdcol< _TT >::_VDBL_stdcol (const type & *__t*) `[inline]`

explicit constructor setting the column's value

Definition at line 431 of file vdbl_col.h.

9.20.3 Member Function Documentation

9.20.3.1 template<class _TT> void _VDBL_stdcol< _TT >::def (type & *d*) const `[inline]`

the default for the constant value coincides with the value, since in the table definition the reference object of this class will hold the default, then. There have to be different access methods `get` and `def` for more complicated column types

Definition at line 454 of file vdbl_col.h.

9.20.3.2 virtual void _VDBL_colbase< _TT >::def_copy (return_type ∗& *d*) const `[inline, virtual, inherited]`

This function returns a pointer to a copy of the column's default value. The copy of the value is allocated by `new`. It has to be `delete`d by the user to avoid memory leaks.

Definition at line 188 of file vdbl_col.h.

9.20.3.3 template<class _TT> void _VDBL_stdcol< _TT >::def_copy (_VDBL_alltype_base ∗& *v*) const `[inline, virtual]`

This version of `get_copy` returns a copy of the columns default value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 469 of file vdbl_col.h.

9.20.3.4 template<class _TT> void _VDBL_stdcol< _TT >::get (type & *c*) const [inline, virtual]

This function stores a copy of the column value into `c`. This function stores a copy of the column default value into `d`. This function sets `c` to a const pointer pointing to the column's actual value. Here, no copying is done. This function returns a pointer to a copy of the column's value. The copy of the value is allocated by `new`. It has to be `delete`d by the user to avoid memory leaks.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 446 of file vdbl_col.h.

9.20.3.5 template<class _TT> void _VDBL_stdcol< _TT >::get_copy (_VDBL_alltype_base ∗& *v*) const [inline, virtual]

This version of `get_copy` returns a copy of the columns value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 462 of file vdbl_col.h.

9.20.3.6 template<class _TT> const type& _VDBL_stdcol< _TT >::get_val () const [inline]

get a const reference to the column value

Definition at line 491 of file vdbl_col.h.

9.20.3.7 template<class _TT> _Self∗ _VDBL_stdcol< _TT >::new_copy () const [inline, virtual]

`new_copy` is the clone operation for copy-constructor overloading. `setcontext` sets the context for value retrieval.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 433 of file vdbl_col.h.

9.20.3.8 template<class _TT> std::ostream& _VDBL_stdcol< _TT >::print_contents (std::ostream & *o*) const [inline, virtual]

This function is needed for the operator<< for columns of type `return_type`.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 493 of file vdbl_col.h.

9.20.3.9 virtual const std::type_info& _VDBL_colbase< _TT >::return_type_id () const `[inline, virtual, inherited]`

This function returns the `type_info` of the column type. This information is used during run-time type checking.

Definition at line 221 of file vdbl_col.h.

9.20.3.10 template<class _TT> void _VDBL_stdcol< _TT >::set (const type & *__t*) `[inline]`

set the column value

Definition at line 479 of file vdbl_col.h.

9.20.3.11 template<class _TT> void _VDBL_stdcol< _TT >::set (const _Self & *_p*) `[inline]`

explicit copy operation

Definition at line 438 of file vdbl_col.h.

9.20.3.12 template<class _TT> void _VDBL_stdcol< _TT >::set_default (const type & *__t*) `[inline]`

set the default value for this column. This is actually equivalent to `set`, since default and standard columns coincide for constant values.

Definition at line 486 of file vdbl_col.h.

9.20.3.13 template<class _TT> void _VDBL_stdcol< _TT >::setcontext (const context ∗ *_c*, const _VDBL_row ∗ *_r*) `[inline]`

this method is empty, since constant values are independend of the context.

Definition at line 444 of file vdbl_col.h.

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

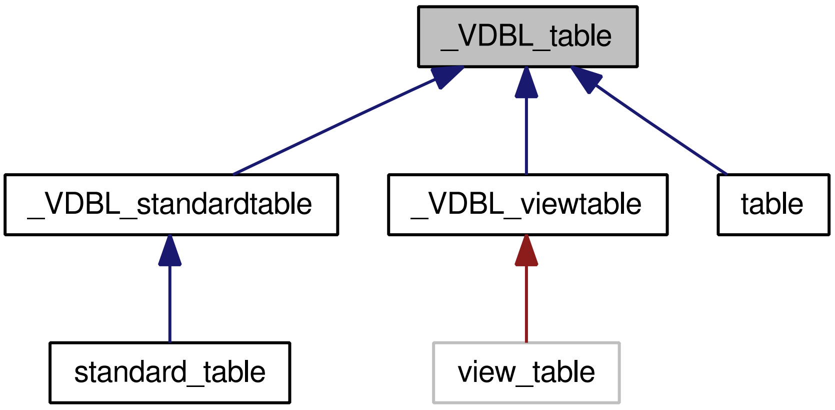
- vdbl_col.h

9.21 _VDBL_table Class Reference

the base class describing database tables

`#include <vdbl_table.h>`

Inheritance diagram for _VDBL_table:



Classes

- struct _col_iterator
- class _col_iterator_base
- struct _row_iterator

Public Types

- typedef std::pair< std::string, _VDBL_col > _T_colspec
- typedef std::pair< const std::string ∗, const _VDBL_col ∗ > _T_ptrcolspec
- typedef _col_iterator< std::pair< std::string, _VDBL_colid >, const std::pair< std::string, _-VDBL_colid > &, const std::pair< std::string, _VDBL_colid > ∗ > col_const_iterator
- typedef _row_iterator< _VDBL_rowid, const _VDBL_rowid &, const _VDBL_rowid ∗ > row_-const_iterator

Public Member Functions

- virtual const std::type_info & get_colinfo (const std::string &_C_n, std::triple< bool, _VDBL_colid, _VDBL_colflags > &_r) const VDBL_PURE_VIRTUAL public
- virtual std::pair< std::string, _VDBL_colid > _next_col (const std::pair< std::string, _VDBL_colid > &_ci) const VDBL_PURE_VIRTUALvirtual std
- virtual _VDBL_rowid _next_row (const _VDBL_rowid &_ci) const VDBL_PURE_-VIRTUALvirtual _VDBL_rowid _prev_row(const _VDBL_rowid &_ci) const VDBL_PURE_-VIRTUALvirtual row_const_iterator row_begin() const VDBL_PURE_VIRTUALvirtual row_-const_iterator row_end() const VDBL_PURE_VIRTUALvirtual row_const_iterator ∗row_iterator_-copy(const row_const_iterator &) const VDBL_PURE_VIRTUAL public
- _VDBL_table (const _VDBL_table &__t)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 >
  _VDBL_table (const __SequenceCtr< std::triple< std::string, _VDBL_col, _VDBL_colflags >, Al-locator1 > &__cc)
- virtual ∼_VDBL_table ()
- virtual _VDBL_table ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_-table ∗t) VDBL_PURE_VIRTUALvirtual bool add_col(const std

- template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn >
  bool insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > &_rows)

Protected Member Functions

- void made_change ()

  - _VDBL_colid get_colid ()
  - _VDBL_rowid get_rowid ()

Friends

- class _VDBL_view

9.21.1 Detailed Description

This is the base class of all tables within a database. A table is defined by a set of columns, which can be added and removed dynamically. The columns might have default values. Those are stored with the table, as well. The columns all consist of objects of class `_VDBL_col`.

The whole table then consists of a set of rows (`_VDBL_row` objects), organized in a map rowid -> row.

Definition at line 68 of file vdbl_table.h.

9.21.2 Member Typedef Documentation

9.21.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_table::_T_colspec

specifier of one column, a pair of column name (`string`) and entry (`_VDBL_col`).

Reimplemented in _VDBL_standardtable, and _VDBL_viewtable.

Definition at line 78 of file vdbl_table.h.

9.21.2.2 typedef std::pair<const std::string*,const _VDBL_col*> _VDBL_table::_T_ptrcolspec

specifier of pointers to one column, a pair of column name (`string*`) and entry (`_VDBL_col*`).

Reimplemented in _VDBL_standardtable.

Definition at line 83 of file vdbl_table.h.

9.21.2.3 typedef _col_iterator<std::pair<std::string,_VDBL_colid>, const std::pair<std::string,_VDBL_colid>&, const std::pair<std::string,_VDBL_colid>*> _VDBL_table::col_const_iterator

const iterator over all columns

Reimplemented in _VDBL_standardtable.

Definition at line 214 of file vdbl_table.h.

9.21.2.4 typedef _row_iterator<_VDBL_rowid, const _VDBL_rowid&, const _VDBL_rowid*> _VDBL_table::row_const_iterator

const iterator over all rows

Definition at line 321 of file vdbl_table.h.

9.21.3 Constructor & Destructor Documentation

9.21.3.1 _VDBL_table::_VDBL_table (const _VDBL_table & *__t*) `[inline]`

copy constructor

Definition at line 359 of file vdbl_table.h.

9.21.3.2 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 > _VDBL_table::_VDBL_table (const __SequenceCtr< std::triple< std::string, _VDBL_col, _VDBL_colflags >, Allocator1 > & *__cc*) `[inline]`

constructor which builds a table from a list of column definitions. This list can be contained in any sequential STL container.

Definition at line 368 of file vdbl_table.h.

9.21.3.3 virtual _VDBL_table::~_VDBL_table () `[inline, virtual]`

standard destructor

Definition at line 375 of file vdbl_table.h.

9.21.4 Member Function Documentation

9.21.4.1 virtual std::pair<std::string,_VDBL_colid> _VDBL_table::_next_col (const std::pair< std::string, _VDBL_colid > & *_ci*) const `[inline, virtual]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous column for a `_col_iterator`. return iterator to first column return iterator beyond last column This is the fundamental class for iterators over all rows, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented in _VDBL_standardtable.

Definition at line 221 of file vdbl_table.h.

9.21.4.2 virtual _VDBL_rowid _VDBL_table::_next_row (const _VDBL_rowid & *_ci*) const `[inline, virtual]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next row for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous row for a `_col_iterator`. return iterator to first row return iterator beyond last row construct a new copy of a row iterator standard constructor

Reimplemented in _VDBL_standardtable.

Definition at line 328 of file vdbl_table.h.

9.21.4.3 _VDBL_colid _VDBL_table::get_colid () `[inline, protected]`

generate new unique id's for rows and columns

Definition at line 99 of file vdbl_table.h.

9.21.4.4 virtual const std::type_info& _VDBL_table::get_colinfo (const std::string & *_C_n*, std::triple< bool, _VDBL_colid, _VDBL_colflags > & *_r*) const `[inline, virtual]`

return the column information for column `_C_n`. the return value is a referenc to the `type_info` of the column's value type. The reference `_r` contains upon returning whether the function was successful, the column id, and the column flags. what was the id of the last change to the table

Reimplemented in _VDBL_standardtable.

Definition at line 116 of file vdbl_table.h.

9.21.4.5 _VDBL_rowid _VDBL_table::get_rowid () `[inline, protected]`

generate new unique id's for rows and columns

Definition at line 100 of file vdbl_table.h.

9.21.4.6 template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn > bool _VDBL_table::insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > & *_rows*) `[inline]`

insert a many new rows of specifications `_rows` into the table. The list of rows can be contained in any sequential STL container, which holds any other sequential STL container of column entries. The function returns `true`, if inserting was successful for all rows.

Definition at line 473 of file vdbl_table.h.

9.21.4.7 void _VDBL_table::made_change () `[inline, protected]`

increment the `last_change` counter.

Definition at line 106 of file vdbl_table.h.

9.21.4.8 virtual _VDBL_table* _VDBL_table::new_copy () const `[inline, virtual]`

clone method clone destructor method add a new column of name `_C_n`, which contains `__c`, and has column flags `__f`. The function returns `true`, if adding the column was successful. modify the column of name `_C_n`, to new contents `__c`, and new column flags `__f`. The function returns `true`, if modifying was successful. modify the contents of column of name `_C_n` The function returns `true`, if modifying was successful. modify the column flags of column of name `_C_n` The function returns `true`, if modifying was successful. remove the column `_C_n` from the table The function returns `true`, if erasing was successful. rename the column `_C_old` to new name _C_new. The function returns `true`, if renaming was successful. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful. insert a new row of specification `_row` into the table. The function returns `true`, if inserting was successful. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. Take any sequential STL container to hold the row entries of the column. The function returns `true`, if inserting was successful.

Definition at line 380 of file vdbl_table.h.

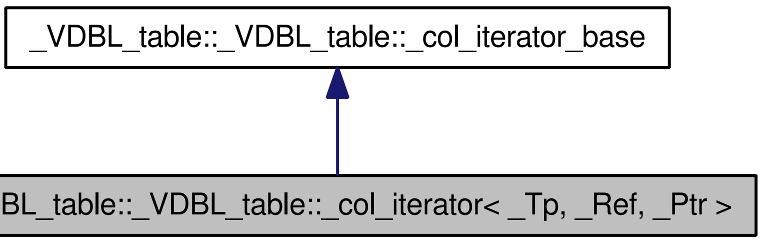
The documentation for this class was generated from the following file:

- vdbl_table.h

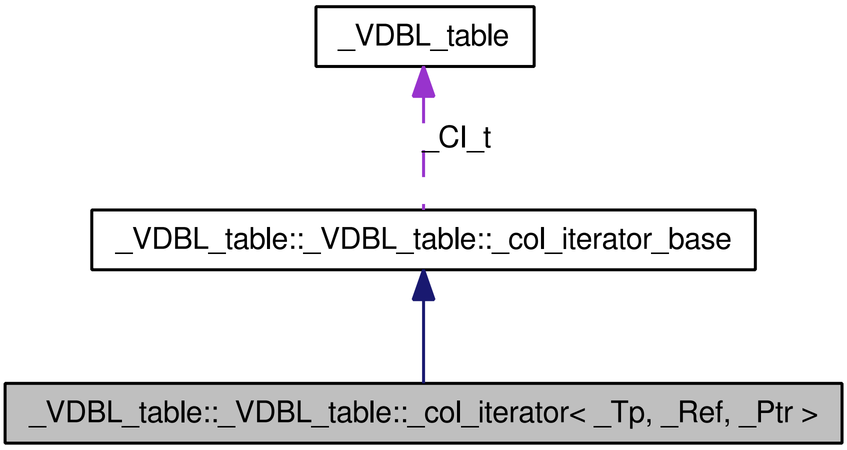
9.22 _VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

`#include <vdbl_table.h>`

Inheritance diagram for _VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr >:


_VDBL_table::_VDBL_table::_col_iterator_base
_VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr >

Collaboration diagram for _VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr >:


_VDBL_table
_CI_t
_VDBL_table::_VDBL_table::_col_iterator_base
_VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr >

9.22.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr> struct _VDBL_table::_col_iterator< _-Tp, _Ref, _Ptr >

This is the base class for iterators over all columns, defining constructors, in(de)crement operations, and dereference operations

Definition at line 169 of file vdbl_table.h.

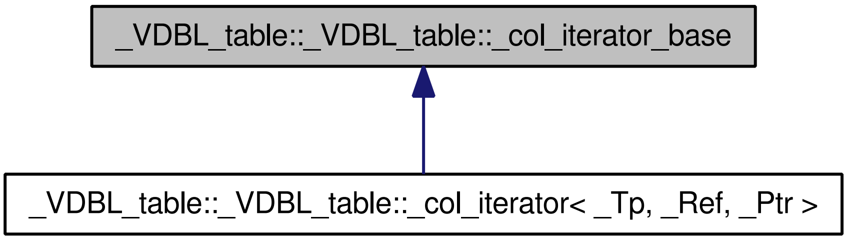
The documentation for this struct was generated from the following file:

- vdbl_table.h

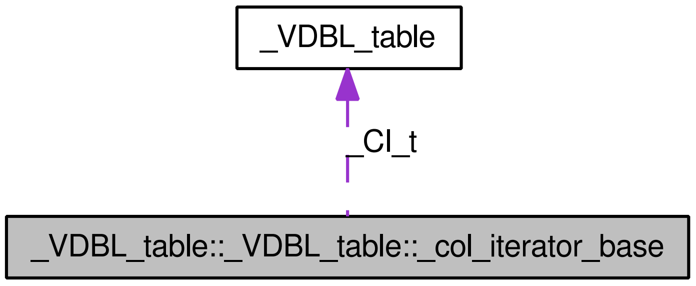
9.23 _VDBL_table::_VDBL_table::_col_iterator_base Class Reference

`#include <vdbl_table.h>`

Inheritance diagram for _VDBL_table::_VDBL_table::_col_iterator_base:


_VDBL_table::_VDBL_table::_col_iterator_base
_VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr >

Collaboration diagram for _VDBL_table::_VDBL_table::_col_iterator_base:



9.23.1 Detailed Description

This is the fundamental class for iterators over all columns, defining basic in(de)crementation for overloading, and basic comparison.

Definition at line 132 of file vdbl_table.h.

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

- vdbl_table.h

9.24 _VDBL_table::_VDBL_table::_row_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

```
#include <vdbl_table.h>
```

9.24.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr> struct _VDBL_table::_row_iterator< _Tp, _Ref, _Ptr >

This is the base class for iterators over all rows, defining constructors, in(de)crement operations, and dereference operations

Definition at line 283 of file vdbl_table.h.

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

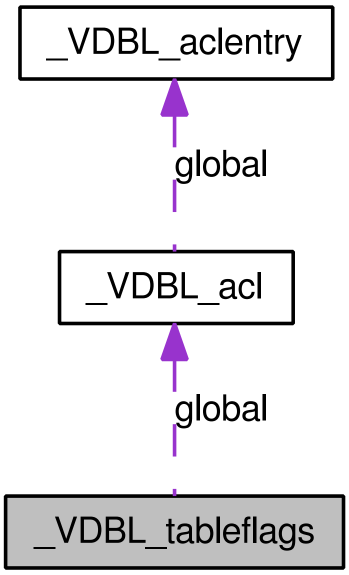
- vdbl_table.h

9.25 _VDBL_tableflags Class Reference

flags for one table

```
#include <vdbl_database.h>
```

Collaboration diagram for _VDBL_tableflags:



Public Member Functions

- _VDBL_tableflags (bool _temp=false, bool _unrest=true)
- _VDBL_tableflags (const _VDBL_acl &_gacl, bool _temp=false, bool _unrest=true)
- _VDBL_tableflags (const _VDBL_tableflags &__t)
- virtual ∼_VDBL_tableflags ()
- _VDBL_tableflags & operator= (const _VDBL_tableflags &__t)

Public Attributes

- bool temporary
- bool unrestricted
- _VDBL_acl global
- std::map< _VDBL_userid, _VDBL_acl > ACLs

9.25.1 Detailed Description

This class describes the additional information for a table within a database, including access control

Definition at line 218 of file vdbl_database.h.

9.25.2 Constructor & Destructor Documentation

9.25.2.1 _VDBL_tableflags::_VDBL_tableflags (bool *_temp* = `false`, bool *_unrest* = `true`) `[inline]`

standard constructor which optionally initializes some members

Definition at line 243 of file vdbl_database.h.

9.25.2.2 _VDBL_tableflags::_VDBL_tableflags (const _VDBL_acl & *_gacl*, bool *_temp* = `false`, bool *_unrest* = `true`) `[inline]`

constructor which initializes the global ACL and optionally some members

Definition at line 250 of file vdbl_database.h.

9.25.2.3 _VDBL_tableflags::_VDBL_tableflags (const _VDBL_tableflags & *__t*) `[inline]`

copy constructor

Definition at line 258 of file vdbl_database.h.

9.25.2.4 virtual _VDBL_tableflags::∼_VDBL_tableflags () `[inline, virtual]`

standard destructor

Definition at line 266 of file vdbl_database.h.

9.25.3 Member Function Documentation

9.25.3.1 _VDBL_tableflags& _VDBL_tableflags::operator= (const _VDBL_tableflags & *__t*) `[inline]`

assignment operator

Definition at line 271 of file vdbl_database.h.

9.25.4 Member Data Documentation

9.25.4.1 std::map<_VDBL_userid,_VDBL_acl> _VDBL_tableflags::ACLs

this is an access control list for every single user

Definition at line 237 of file vdbl_database.h.

9.25.4.2 _VDBL_acl _VDBL_tableflags::global

this is the global access control list (valid for all users)

Definition at line 233 of file vdbl_database.h.

9.25.4.3 bool _VDBL_tableflags::temporary

decides whether this is a temporary table

Definition at line 225 of file vdbl_database.h.

9.25.4.4 bool _VDBL_tableflags::unrestricted

decides whether this is table is completely unrestricted

Definition at line 229 of file vdbl_database.h.

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

- vdbl_database.h

9.26 _VDBL_userflags Class Reference

The permission flags for a user.

`#include <vdbl_database.h>`

Public Member Functions

- _VDBL_userflags ()
- _VDBL_userflags (const _VDBL_userflags &_a)
- virtual ~_VDBL_userflags ()

Public Attributes

- struct {
  } table_privileges
- struct {
  } view_privileges

9.26.1 Detailed Description

This class describes the global privileges of a user w.r.t. tables and views.

Definition at line 60 of file vdbl_database.h.

9.26.2 Constructor & Destructor Documentation

9.26.2.1 _VDBL_userflags::_VDBL_userflags () `[inline]`

standard constructor

Definition at line 93 of file vdbl_database.h.

9.26.2.2 _VDBL_userflags::_VDBL_userflags (const _VDBL_userflags & *_a*) `[inline]`

copy constructor

Definition at line 97 of file vdbl_database.h.

9.26.2.3 virtual _VDBL_userflags::∼_VDBL_userflags () `[inline, virtual]`

standard destructor

Definition at line 103 of file vdbl_database.h.

9.26.3 Member Data Documentation

9.26.3.1 struct { ... } _VDBL_userflags::table_privileges

The global table privileges

- read
- write
- modify
- append
- delete

9.26.3.2 struct { ... } _VDBL_userflags::view_privileges

The global view privileges

- commit to parent table through the view

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

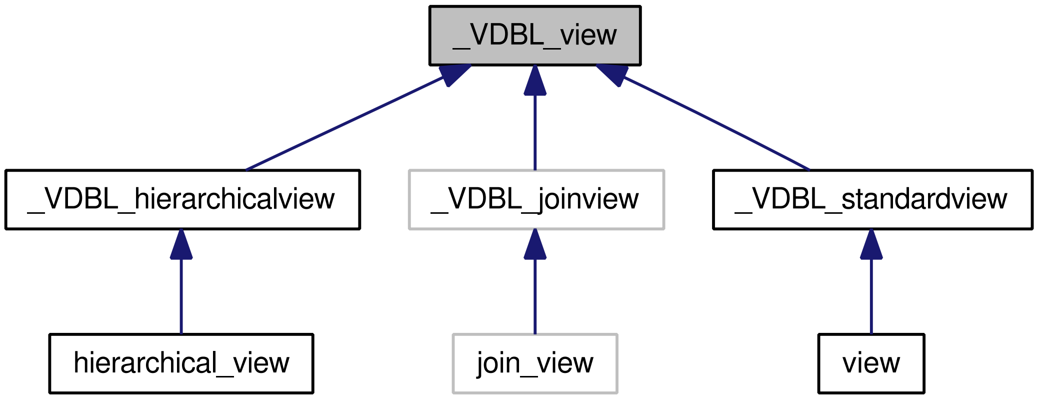
- vdbl_database.h

9.27 _VDBL_view Class Reference

base class of all views.

`#include <vdbl_view.h>`

Inheritance diagram for _VDBL_view:



Classes

- struct _col_iterator
- class _col_iterator_base
- struct _default_iterator
- struct _row_iterator
- class _row_iterator_base

Public Types

- typedef std::pair< std::string, _VDBL_col > _T_colspec

Public Member Functions

- _VDBL_view (_V_enum __e=V_independent)
- _VDBL_view (const _VDBL_view &__v)
- virtual ∼_VDBL_view ()
- virtual _VDBL_view ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_-view ∗v) VDBL_PURE_VIRTUALvirtual const std

Protected Types

- typedef _default_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col ∗ > default_-const_iterator
- typedef _col_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col ∗ > col_const_iterator
- typedef _row_iterator< _VDBL_row, const _VDBL_row &, const _VDBL_row ∗ > row_const_-iterator

Protected Member Functions

- void made_change ()

  increment the `change` *counter.*

- unsigned int get_change_ctr () const

  read the change counter

- virtual std::triple< _VDBL_tableid, _VDBL_colid, void ∗ > _next_def_col (const _VDBL_tableid &_t, const _VDBL_colid &_c, void ∗_d) const VDBL_PURE_VIRTUALvirtual std
- virtual void ∗ _copy_def_data (void ∗_d) const VDBL_PURE_VIRTUALvirtual std
- virtual void ∗ _copy_col_data (void ∗_d) const VDBL_PURE_VIRTUALvirtual std
- virtual void ∗ _copy_row_data (void ∗_d) const VDBL_PURE_VIRTUAL public

9.27.1 Detailed Description

This is the base class of all views. A view in the database must be a subclass of this class.

Definition at line 55 of file vdbl_view.h.

9.27.2 Member Typedef Documentation

9.27.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_view::_T_colspec

This is the description of one column

Definition at line 61 of file vdbl_view.h.

9.27.2.2 typedef _col_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col∗> _VDBL_view::col_const_iterator `[protected]`

const iterator over all columns

Definition at line 439 of file vdbl_view.h.

9.27.2.3 typedef _default_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col∗> _VDBL_view::default_const_iterator `[protected]`

const iterator over all default columns

Definition at line 301 of file vdbl_view.h.

9.27.2.4 typedef _row_iterator<_VDBL_row, const _VDBL_row&, const _VDBL_row∗> _VDBL_view::row_const_iterator `[protected]`

const iterator over all rows

Definition at line 568 of file vdbl_view.h.

9.27.3 Constructor & Destructor Documentation

9.27.3.1 _VDBL_view::_VDBL_view (_V_enum __*e* = `V_independent`) `[inline]`

standard constructor which optionally initializes the view type

Definition at line 574 of file vdbl_view.h.

9.27.3.2 _VDBL_view::_VDBL_view (const _VDBL_view & __*v*) `[inline]`

copy constructor

Definition at line 578 of file vdbl_view.h.

9.27.3.3 virtual _VDBL_view::∼_VDBL_view () `[inline, virtual]`

standard destructor

Definition at line 582 of file vdbl_view.h.

9.27.4 Member Function Documentation

9.27.4.1 virtual void∗ _VDBL_view::_copy_col_data (void ∗ _*d*) const `[inline, protected, virtual]`

This function copies the additional data needed by a _col_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next row for a `_row_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous row for a `_row_iterator`. This function destroys the additional data needed by a _row_iterator

Reimplemented in _VDBL_hierarchicalview, and _VDBL_standardview.

Definition at line 138 of file vdbl_view.h.

9.27.4.2 virtual void∗ _VDBL_view::_copy_def_data (void ∗ _*d*) const `[inline, protected, virtual]`

This function copies the additional data needed by a _default_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous column for a `_col_iterator`. This function destroys the additional data needed by a _col_iterator

Reimplemented in _VDBL_hierarchicalview, and _VDBL_standardview.

Definition at line 106 of file vdbl_view.h.

9.27.4.3 virtual void∗ _VDBL_view::_copy_row_data (void ∗ _d) const [inline, protected, virtual]

This function copies the additional data needed by a _row_iterator This is the fundamental class for iterators over all default columns, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented in _VDBL_hierarchicalview, and _VDBL_standardview.

Definition at line 170 of file vdbl_view.h.

9.27.4.4 virtual std::triple<_VDBL_tableid,_VDBL_colid,void∗> _VDBL_view::_next_def_col (const _VDBL_tableid & _t, const _VDBL_colid & _c, void ∗ _d) const [inline, protected, virtual]

This virtual function has to be overloaded by the derived view classes, and it performs the step to the next default of a column a `_default_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous default of a column a `_default_iterator`. This function destroys the additional data needed by a _default_iterator

Reimplemented in _VDBL_hierarchicalview, and _VDBL_standardview.

Definition at line 82 of file vdbl_view.h.

9.27.4.5 virtual _VDBL_view∗ _VDBL_view::new_copy () const [inline, virtual]

clone operation clone destructor return the column information for column `_C_n`. the return value is a reference to the `type_info` of the column's value type. The reference `_r` contains upon returning whether the function was successful, the column id, and the column flags. return a reference to the `type_-info` of the column's value type. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful. remove the row with it `_ri` from the table return whether the column `_C_n` is defined in the table return table id and column id of column `_C_n` return column name of column `_C_n` return a const reference to the column with column id `_ci` in row `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. Note that in this function the context for the column is NOT set. This method adds the row ids in `_rs` to the visible part of the view. This method removes the row ids in `_rs` from the visible part of the view. print the contents of the column with column id `_ci` in row `_ri.second` of table `ri.first`. If the row is empty and no default is defined, print nothing. return a const reference to the row with row id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return a const reference to the default column with column id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return iterator to first default column return iterator beyond last default column return iterator to first row return iterator beyond last row return iterator to first column return iterator beyond last column return the type of this view

Definition at line 587 of file vdbl_view.h.

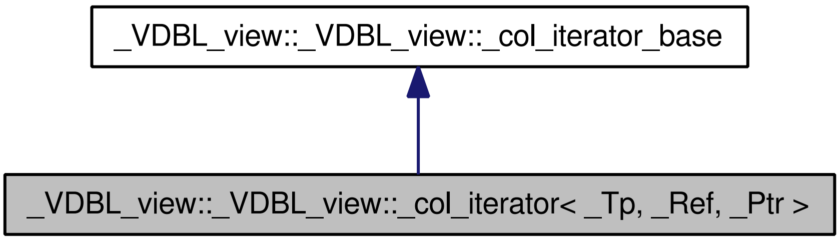
The documentation for this class was generated from the following file:

- vdbl_view.h

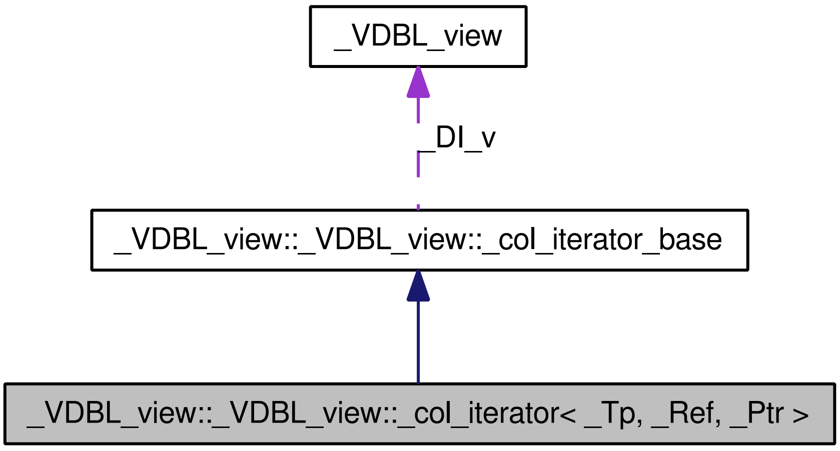
9.28 _VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

`#include <vdbl_view.h>`

Inheritance diagram for _VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr >:


_VDBL_view::_VDBL_view::_col_iterator_base
_VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr >

Collaboration diagram for _VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr >:


_VDBL_view
_DI_v
_VDBL_view::_VDBL_view::_col_iterator_base
_VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr >

9.28.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr> struct _VDBL_view::_col_iterator< _-Tp, _Ref, _Ptr >

This is the base class for iterators over all columns, defining constructors, in(de)crement operations, and dereference operations

Definition at line 387 of file vdbl_view.h.

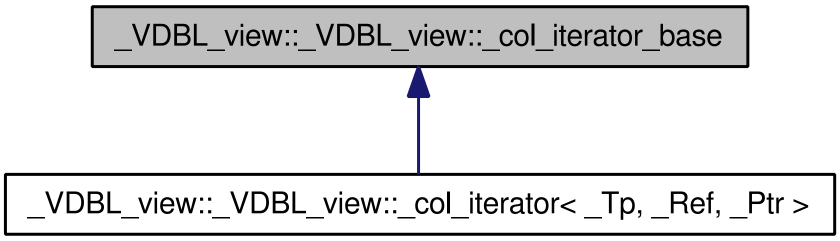
The documentation for this struct was generated from the following file:

- vdbl_view.h

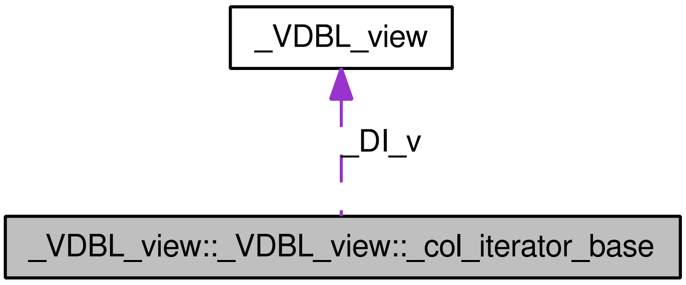
9.29 _VDBL_view::_VDBL_view::_col_iterator_base Class Reference

`#include <vdbl_view.h>`

Inheritance diagram for _VDBL_view::_VDBL_view::_col_iterator_base:


_VDBL_view::_VDBL_view::_col_iterator_base
_VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr >

Collaboration diagram for _VDBL_view::_VDBL_view::_col_iterator_base:



9.29.1 Detailed Description

This is the fundamental class for iterators over all columns, defining basic in(de)crementation for overloading, and basic comparison.

Definition at line 308 of file vdbl_view.h.

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

- vdbl_view.h

9.30 _VDBL_view::_VDBL_view::_default_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

```
#include <vdbl_view.h>
```

9.30.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr> struct _VDBL_view::_default_iterator< _Tp, _Ref, _Ptr >

This is the base class for iterators over all default columns, defining constructors, in(de)crement operations, and dereference operations

Definition at line 251 of file vdbl_view.h.

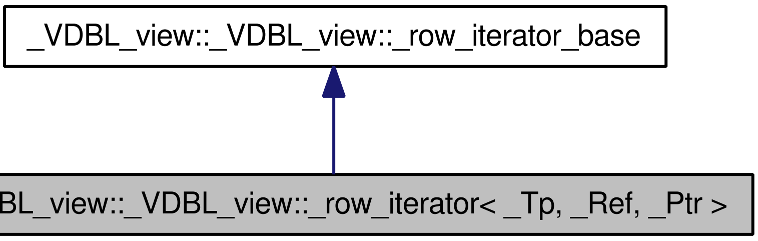
The documentation for this struct was generated from the following file:

- vdbl_view.h

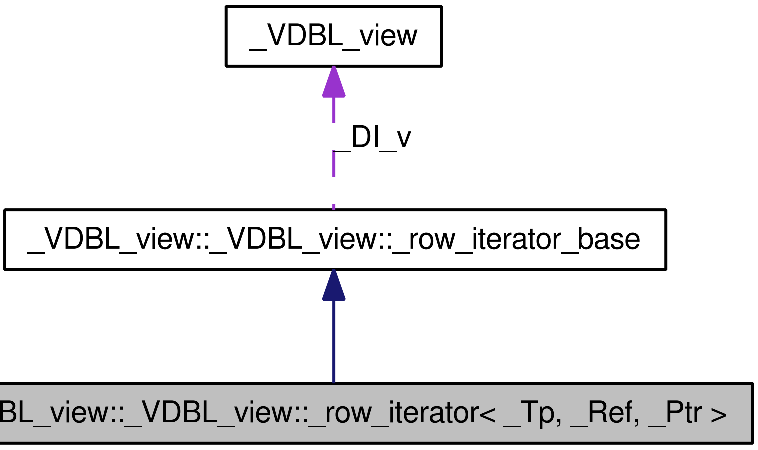
9.31 _VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

`#include <vdbl_view.h>`

Inheritance diagram for _VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr >:


_VDBL_view::_VDBL_view::_row_iterator_base
_VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr >

Collaboration diagram for _VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr >:


_VDBL_view
_DI_v
_VDBL_view::_VDBL_view::_row_iterator_base
_VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr >

9.31.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr> struct _VDBL_view::_row_iterator< _-Tp, _Ref, _Ptr >

This is the base class for iterators over all rows, defining constructors, in(de)crement operations, and dereference operations

Definition at line 520 of file vdbl_view.h.

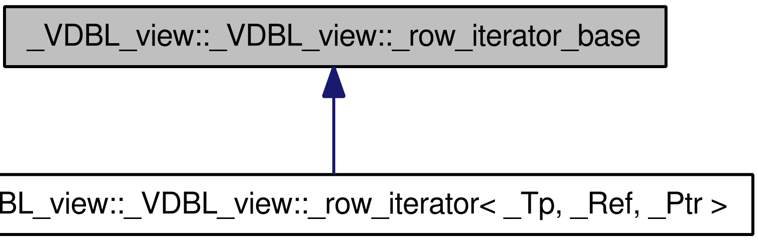
The documentation for this struct was generated from the following file:

- vdbl_view.h

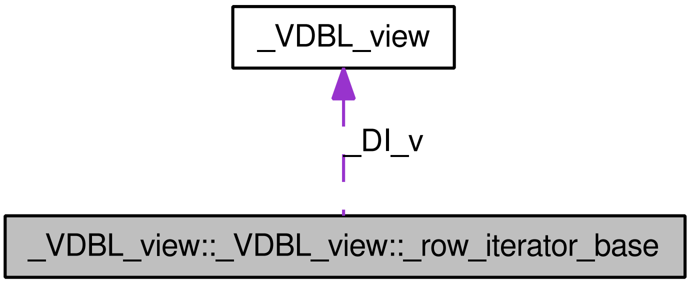
9.32 _VDBL_view::_VDBL_view::_row_iterator_base Class Reference

`#include <vdbl_view.h>`

Inheritance diagram for _VDBL_view::_VDBL_view::_row_iterator_base:


_VDBL_view::_VDBL_view::_row_iterator_base
_VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr >

Collaboration diagram for _VDBL_view::_VDBL_view::_row_iterator_base:



9.32.1 Detailed Description

This is the fundamental class for iterators over all columns, defining basic in(de)crementation for overloading, and basic comparison.

Definition at line 446 of file vdbl_view.h.

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

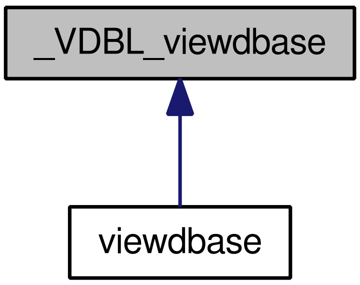
- vdbl_view.h

9.33 _VDBL_viewdbase Class Reference

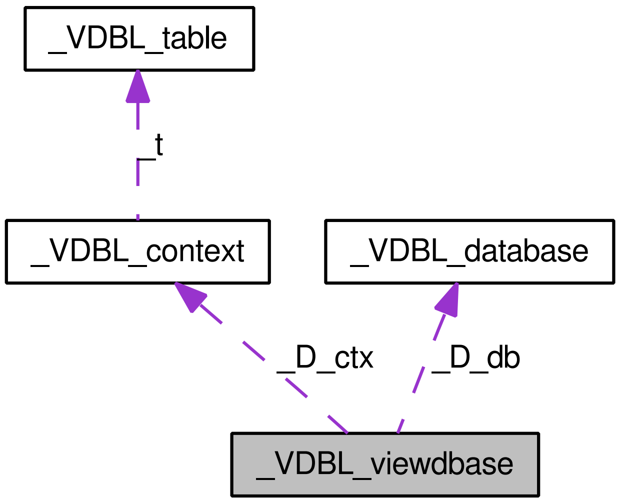
a view to a complete database

`#include <vdbl_viewdbase.h>`

Inheritance diagram for _VDBL_viewdbase:



Collaboration diagram for _VDBL_viewdbase:



Public Member Functions

- _VDBL_tableid get_tableid (const std::string &_C_i) const
- bool has_view (const _VDBL_tableid &_C_i) const
- bool has_view (const std::string &_C_i) const
- _VDBL_view ∗ get_view (const _VDBL_tableid &_C_i) const
- _VDBL_view ∗ get_view (const std::string &_C_i) const
- _VDBL_viewdbase ()
- _VDBL_viewdbase (const _VDBL_database &_db, const _VDBL_userid &uid, const _VDBL_-context &__c, const _V_enum &__e=V_frozen)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  _VDBL_viewdbase (const _VDBL_database &_db, const _VDBL_userid &uid, const _VDBL_-context &__c, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an, const _V_enum &__e=V_frozen)
- virtual ∼_VDBL_viewdbase ()
- bool add_table_view (const _VDBL_tableid &__t, const std::string &__s, const _VDBL_context &__c)

- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  bool add_table_view (const _VDBL_tableid &__t, const std::string &__s, const _VDBL_context &__c, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an)
- bool add_table_view (const std::string &__s, const _VDBL_context &__c)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  bool add_table_view (const std::string &__s, const _VDBL_context &__c, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  bool add_visible (const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  bool rm_visible (const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an)

9.33.1 Detailed Description

This class implements a view onto a complete database. The view names correspond to the names of all existing tables.

Definition at line 51 of file vdbl_viewdbase.h.

9.33.2 Constructor & Destructor Documentation

9.33.2.1 _VDBL_viewdbase::_VDBL_viewdbase () `[inline]`

standard constructor

Definition at line 151 of file vdbl_viewdbase.h.

9.33.2.2 _VDBL_viewdbase::_VDBL_viewdbase (const _VDBL_database & *_db*, const _VDBL_userid & *uid*, const _VDBL_context & *__c*, const _V_enum & *__e* = `V_frozen`) `[inline]`

constructor which builds a view to the database from

- `_db` – the database
- `__c` – the evaluation context for all views

Definition at line 159 of file vdbl_viewdbase.h.

9.33.2.3 template<template< class _TC, class _TA > class _SqCtr, class _Al > _VDBL_viewdbase::_VDBL_viewdbase (const _VDBL_database & *_db*, const _VDBL_userid & *uid*, const _VDBL_context & *__c*, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*, const _V_enum & *__e* = `V_frozen`) `[inline]`

constructor which builds a view to the database from

- `_db` – the database

- ___c – the evaluation context for all views The third argument is any sequential container of table,row pairs to which the view shall be restricted.

Definition at line 184 of file vdbl_viewdbase.h.

9.33.2.4 virtual _VDBL_viewdbase::∼_VDBL_viewdbase () [inline, virtual]

standard destructor

Definition at line 203 of file vdbl_viewdbase.h.

9.33.3 Member Function Documentation

9.33.3.1 template<template< class _TC, class _TA > class _SqCtr, class _Al > bool _VDBL_viewdbase::add_table_view (const std::string & *__s*, const _VDBL_context & *__c*, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*) [inline]

This method adds another view to the viewdb. The table name is given by ___s, and the context by ___c. The sequential container of table,row pairs specifies additional table,row pairs which shall be visible in the view.

Reimplemented in viewdbase.

Definition at line 264 of file vdbl_viewdbase.h.

9.33.3.2 bool _VDBL_viewdbase::add_table_view (const std::string & *__s*, const _VDBL_context & *__c*) [inline]

This method adds another view to the viewdb. The table name is given by ___s, and the context by ___c.

Reimplemented in viewdbase.

Definition at line 249 of file vdbl_viewdbase.h.

9.33.3.3 template<template< class _TC, class _TA > class _SqCtr, class _Al > bool _VDBL_viewdbase::add_table_view (const _VDBL_tableid & *__t*, const std::string & *__s*, const _VDBL_context & *__c*, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*) [inline]

This method adds another view to the viewdb. The table id is given by ___t, the table name is given by ___s, and the context by ___c. The sequential container of table,row pairs specifies additional table,row pairs which shall be visible in the view.

Reimplemented in viewdbase.

Definition at line 232 of file vdbl_viewdbase.h.

9.33.3.4 bool _VDBL_viewdbase::add_table_view (const _VDBL_tableid & *__t*, const std::string & *__s*, const _VDBL_context & *__c*) `[inline]`

This method adds another view to the viewdb. The table id is given by `__t`, the table name is given by `__s`, and the context by `__c`.

Reimplemented in viewdbase.

Definition at line 213 of file vdbl_viewdbase.h.

9.33.3.5 template<template< class _TC, class _TA > class _SqCtr, class _Al > bool _VDBL_viewdbase::add_visible (const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*) `[inline]`

The sequential container of table,row pairs specifies additional table,row pairs which shall be visible in the view.

Reimplemented in viewdbase.

Definition at line 278 of file vdbl_viewdbase.h.

9.33.3.6 _VDBL_tableid _VDBL_viewdbase::get_tableid (const std::string & *_C_i*) const `[inline]`

return the table id (and view id) of table `_C_i`

Definition at line 73 of file vdbl_viewdbase.h.

9.33.3.7 _VDBL_view∗ _VDBL_viewdbase::get_view (const std::string & *_C_i*) const `[inline]`

this method returns a pointer to the view `_C_i`.

Reimplemented in viewdbase.

Definition at line 145 of file vdbl_viewdbase.h.

9.33.3.8 _VDBL_view∗ _VDBL_viewdbase::get_view (const _VDBL_tableid & *_C_i*) const `[inline]`

this method returns a pointer to the view associated to id `_C_i`.

Definition at line 115 of file vdbl_viewdbase.h.

9.33.3.9 bool _VDBL_viewdbase::has_view (const std::string & *_C_i*) const `[inline]`

check whether the view `_C_i` exists

Definition at line 109 of file vdbl_viewdbase.h.

9.33.3.10 bool _VDBL_viewdbase::has_view (const _VDBL_tableid & *_C_i*) const `[inline]`

check whether a given view (associated to table id `_C_i`) exists

Definition at line 92 of file vdbl_viewdbase.h.

9.33.3.11 template<template< class _TC, class _TA > class _SqCtr, class _Al > bool _VDBL_viewdbase::rm_visible (const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*) `[inline]`

The sequential container of table,row pairs specifies table,row pairs which shall no longer be visible in the view.

Reimplemented in viewdbase.

Definition at line 292 of file vdbl_viewdbase.h.

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

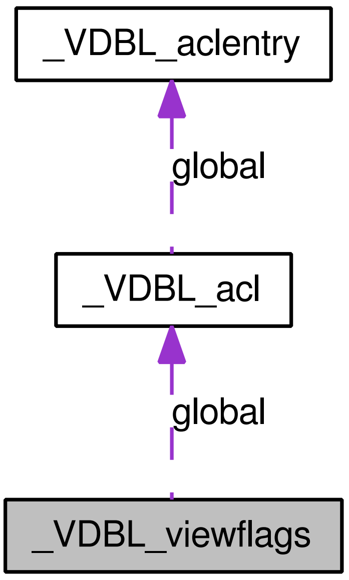
- vdbl_viewdbase.h

9.34 _VDBL_viewflags Class Reference

flags for one view

`#include <vdbl_database.h>`

Collaboration diagram for _VDBL_viewflags:



Public Member Functions

- _VDBL_viewflags ()
- _VDBL_viewflags (const _VDBL_acl &_gacl)
- _VDBL_viewflags (const _VDBL_viewflags &__v)
- virtual ∼_VDBL_viewflags ()
- _VDBL_viewflags & operator= (const _VDBL_viewflags &__v)

Public Attributes

- _VDBL_acl global
- std::map< _VDBL_userid, _VDBL_acl > ACLs

9.34.1 Detailed Description

This class describes the additional information for a view within a database, including access control

Definition at line 287 of file vdbl_database.h.

9.34.2 Constructor & Destructor Documentation

9.34.2.1 _VDBL_viewflags::_VDBL_viewflags () `[inline]`

standard constructor

Definition at line 303 of file vdbl_database.h.

9.34.2.2 _VDBL_viewflags::_VDBL_viewflags (const _VDBL_acl & *_gacl*) `[inline]`

constructor which initializes the global ACL entry

Definition at line 308 of file vdbl_database.h.

9.34.2.3 _VDBL_viewflags::_VDBL_viewflags (const _VDBL_viewflags & *__v*) `[inline]`

copy constructor

Definition at line 313 of file vdbl_database.h.

9.34.2.4 virtual _VDBL_viewflags::∼_VDBL_viewflags () `[inline, virtual]`

standard destructor

Definition at line 318 of file vdbl_database.h.

9.34.3 Member Function Documentation

9.34.3.1 _VDBL_viewflags& _VDBL_viewflags::operator= (const _VDBL_viewflags & *__v*) `[inline]`

assignment operator

Definition at line 323 of file vdbl_database.h.

9.34.4 Member Data Documentation

9.34.4.1 std::map< _VDBL_userid, _VDBL_acl > _VDBL_viewflags::ACLs

this is an access control list for every single user

Definition at line 297 of file vdbl_database.h.

9.34.4.2 _VDBL_acl _VDBL_viewflags::global

this is the global access control list (valid for all users)

Definition at line 293 of file vdbl_database.h.

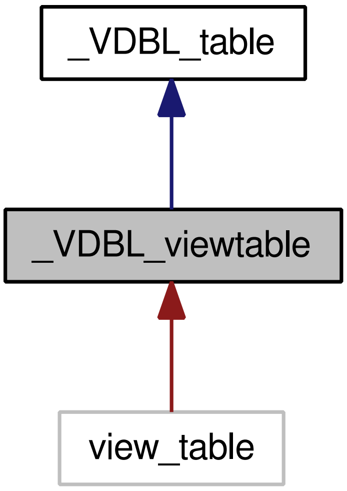
The documentation for this class was generated from the following file:

- vdbl_database.h

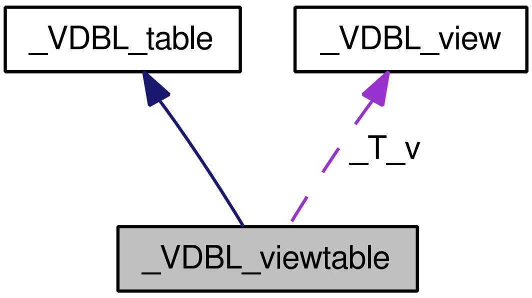
9.35 _VDBL_viewtable Class Reference

`#include <vdbl_vtable.h>`

Inheritance diagram for _VDBL_viewtable:



Collaboration diagram for _VDBL_viewtable:



Public Types

- typedef std::pair< std::string, _VDBL_col > _T_colspec
- typedef std::pair< const std::string ∗, const _VDBL_col ∗ > _T_ptrcolspec
- typedef _col_iterator< std::pair< std::string, _VDBL_colid >, const std::pair< std::string, _VDBL_colid > &, const std::pair< std::string, _VDBL_colid > ∗ > col_const_iterator
- typedef _row_iterator< _VDBL_rowid, const _VDBL_rowid &, const _VDBL_rowid ∗ > row_const_iterator

Public Member Functions

- bool insert (const std::vector< _T_colspec > &_row)
- bool remove (const _VDBL_rowid _ri)
- virtual std::pair< std::string, _VDBL_colid > _next_col (const std::pair< std::string, _VDBL_colid > &_ci) const VDBL_PURE_VIRTUALvirtual std
- virtual _VDBL_rowid _next_row (const _VDBL_rowid &_ci) const VDBL_PURE_VIRTUALvirtual _VDBL_rowid _prev_row(const _VDBL_rowid &_ci) const VDBL_PURE_VIRTUALvirtual row_const_iterator row_begin() const VDBL_PURE_VIRTUALvirtual row_const_iterator row_end() const VDBL_PURE_VIRTUALvirtual row_const_iterator ∗row_iterator_copy(const row_const_iterator &) const VDBL_PURE_VIRTUAL public
- virtual _VDBL_table ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_table ∗t) VDBL_PURE_VIRTUALvirtual bool add_col(const std
- template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn >
  bool insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > &_rows)

Protected Member Functions

- void made_change ()

  - _VDBL_colid get_colid ()
  - _VDBL_rowid get_rowid ()

Friends

- class _VDBL_view

9.35.1 Detailed Description

this is a table on top of a view. The view can be used like a table later. This is especially useful for constructing a hierarchy of tables. Depending on the "transparency" of the view rows are automatically updated or "overshadowed".

Definition at line 47 of file vdbl_vtable.h.

9.35.2 Member Typedef Documentation

9.35.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_viewtable::_T_colspec

specifier of one column, a pair of column name (`string`) and entry (`_VDBL_col`).

Reimplemented from _VDBL_table.

Definition at line 63 of file vdbl_vtable.h.

9.35.2.2 typedef std::pair<const std::string*,const _VDBL_col*> _VDBL_table::_T_ptrcolspec `[inherited]`

specifier of pointers to one column, a pair of column name (`string*`) and entry (`_VDBL_col*`).

Reimplemented in _VDBL_standardtable.

Definition at line 83 of file vdbl_table.h.

9.35.2.3 typedef _col_iterator<std::pair<std::string,_VDBL_colid>, const std::pair<std::string,_VDBL_colid>&, const std::pair<std::string,_VDBL_colid>*> _VDBL_table::col_const_iterator `[inherited]`

const iterator over all columns

Reimplemented in _VDBL_standardtable.

Definition at line 214 of file vdbl_table.h.

9.35.2.4 typedef _row_iterator<_VDBL_rowid, const _VDBL_rowid&, const _VDBL_rowid*> _VDBL_table::row_const_iterator `[inherited]`

const iterator over all rows

Definition at line 321 of file vdbl_table.h.

9.35.3 Member Function Documentation

9.35.3.1 virtual std::pair<std::string,_VDBL_colid> _VDBL_table::_next_col (const std::pair< std::string, _VDBL_colid > & *_ci*) const `[inline, virtual, inherited]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous column for a `_col_iterator`. return iterator to first column return iterator beyond last column This is the fundamental class for iterators over all rows, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented in _VDBL_standardtable.

Definition at line 221 of file vdbl_table.h.

9.35.3.2 virtual _VDBL_rowid _VDBL_table::_next_row (const _VDBL_rowid & *_ci*) const `[inline, virtual, inherited]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next row for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous row for a `_col_iterator`. return iterator to first row return iterator beyond last row construct a new copy of a row iterator standard constructor

Reimplemented in _VDBL_standardtable.

Definition at line 328 of file vdbl_table.h.

9.35.3.3 _VDBL_colid _VDBL_table::get_colid () `[inline, protected, inherited]`

generate new unique id's for rows and columns

Definition at line 99 of file vdbl_table.h.

9.35.3.4 _VDBL_rowid _VDBL_table::get_rowid () `[inline, protected, inherited]`

generate new unique id's for rows and columns

Definition at line 100 of file vdbl_table.h.

9.35.3.5 bool _VDBL_viewtable::insert (const std::vector< _T_colspec > & *_row*) `[inline]`

> must not change this

> insert them into local rows

Definition at line 138 of file vdbl_vtable.h.

9.35.3.6 template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn > bool _VDBL_table::insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > & *_rows*) `[inline, inherited]`

insert a many new rows of specifications `_rows` into the table. The list of rows can be contained in any sequential STL container, which holds any other sequential STL container of column entries. The function returns `true`, if inserting was successful for all rows.

Definition at line 473 of file vdbl_table.h.

9.35.3.7 void _VDBL_table::made_change () `[inline, protected, inherited]`

increment the `last_change` counter.

Definition at line 106 of file vdbl_table.h.

9.35.3.8 virtual _VDBL_table∗ _VDBL_table::new_copy () const `[inline, virtual, inherited]`

clone method clone destructor method add a new column of name `_C_n`, which contains `__c`, and has column flags `__f`. The function returns `true`, if adding the column was successful. modify the column of name `_C_n`, to new contents `__c`, and new column flags `__f`. The function returns `true`, if modifying was successful. modify the contents of column of name `_C_n` The function returns `true`, if modifying was successful. modify the column flags of column of name `_C_n` The function returns `true`, if modifying was successful. remove the column `_C_n` from the table The function returns `true`, if erasing was successful. rename the column `_C_old` to new name _C_new. The function returns `true`, if renaming was successful. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful. insert a new row of specification `_row` into the table. The function returns `true`, if inserting was successful. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. Take any sequential STL container to hold the row entries of the column. The function returns `true`, if inserting was successful.

Definition at line 380 of file vdbl_table.h.

9.35.3.9 bool _VDBL_viewtable::remove (const _VDBL_rowid *_ri*) `[inline]`

can only remove local rows unless we have a window view

Definition at line 210 of file vdbl_vtable.h.

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

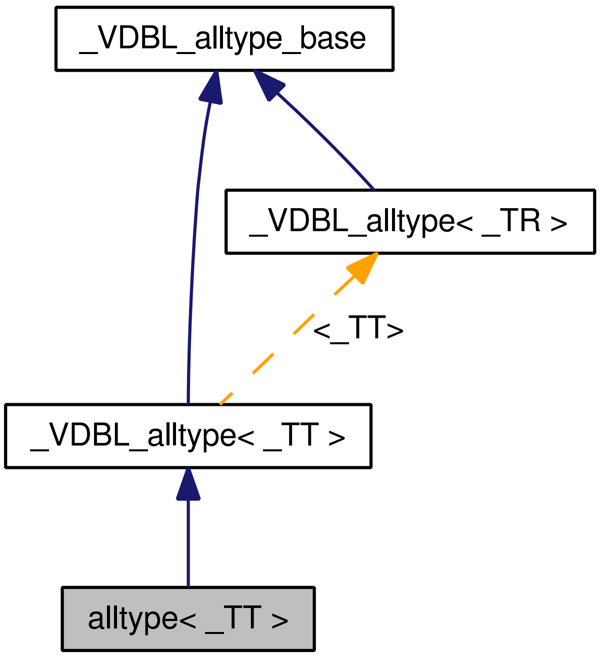
- vdbl_vtable.h

9.36 alltype< _TT > Class Template Reference

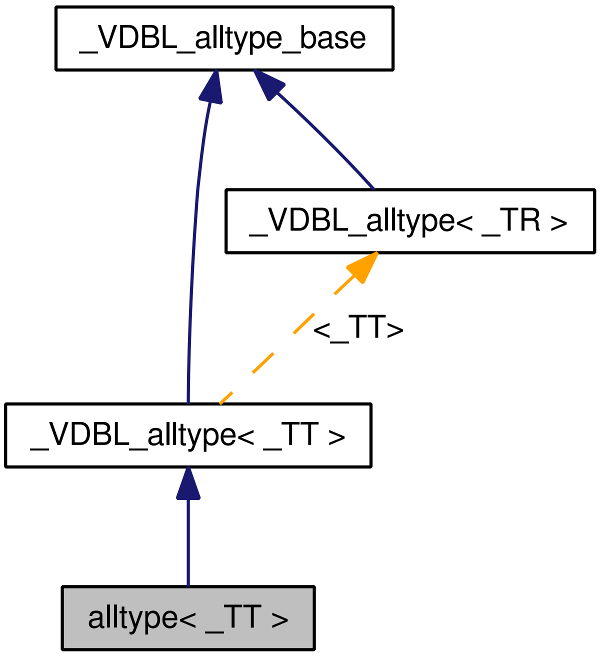
The templated alltype class.

```
#include <vdbl_alltype.h>
```

Inheritance diagram for alltype< _TT >:



Collaboration diagram for alltype< _TT >:



Public Types

- typedef _Base::cont_type cont_type

  the type of the internally stored data

Public Member Functions

- const std::type_info & get_type () const
- const cont_type & content () const

  - bool operator== (const _VDBL_alltype_base &p) const
  - bool operator!= (const _VDBL_alltype_base &p) const

  - bool operator== (const _Self &p) const
  - bool operator!= (const _Self &p) const

9.36.1 Detailed Description

template<class _TT> class alltype< _TT >

This class is used to hold data of arbitrary types. It is manly used as return value.

Data stored in this class has to provide a copy constructor, an assignment operator.

Finaly, it would be useful if the stored type has a '<<'operator.

Definition at line 622 of file vdbl_alltype.h.

9.36.2 Member Function Documentation

9.36.2.1 const cont_type& _VDBL_alltype< _TT >::content () const `[inline, inherited]`

This method returns a const reference to the stored data

Definition at line 140 of file vdbl_alltype.h.

9.36.2.2 const std::type_info& _VDBL_alltype< _TT >::get_type () const `[inline, inherited]`

This member function is used for run-time type checking. It returns the of the .

Definition at line 135 of file vdbl_alltype.h.

9.36.2.3 bool _VDBL_alltype< _TT >::operator!= (const _Self & *p*) const `[inline, inherited]`

The standard comparison operators are mainly used for expressions and selectors.

Definition at line 175 of file vdbl_alltype.h.

9.36.2.4 bool _VDBL_alltype< _TT >::operator!= (const _VDBL_alltype_base & *p*) const `[inline, inherited]`

The standard comparison operators are mainly used for expressions and selectors.

Definition at line 164 of file vdbl_alltype.h.

9.36.2.5 bool _VDBL_alltype< _TT >::operator== (const _Self & *p*) const `[inline, inherited]`

The standard comparison operators are mainly used for expressions and selectors.

Definition at line 174 of file vdbl_alltype.h.

9.36.2.6 bool _VDBL_alltype< _TT >::operator== (const _VDBL_alltype_base & *p*) const `[inline, inherited]`

The standard comparison operators are mainly used for expressions and selectors.

Definition at line 161 of file vdbl_alltype.h.

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

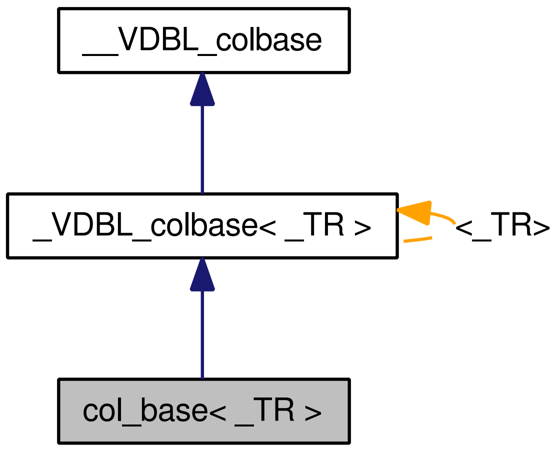
- vdbl_alltype.h

9.37 col_base< _TR > Class Template Reference

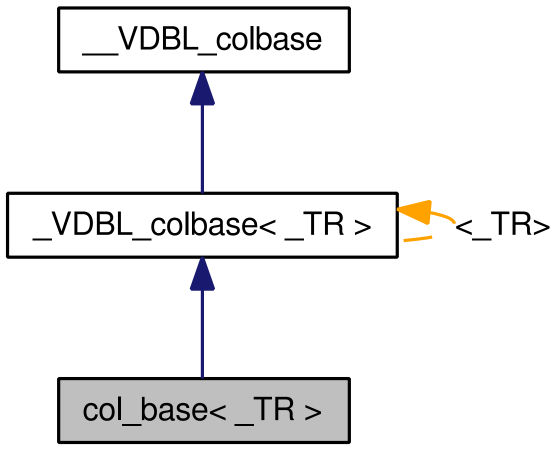
column base class

`#include <vdbl_col.h>`

Inheritance diagram for col_base< _TR >:



Collaboration diagram for col_base< _TR >:



Public Types

- typedef _TR return_type

  `return_type` *is the type of object stored*

Public Member Functions

- virtual _Self ∗ new_copy () const VDBL_PURE_VIRTUALvirtual void setcontext(const context ∗_c
- virtual _Self ∗const _VDBL_row ∗_r virtual VDBL_PURE_VIRTUAL void get (return_type &c) const VDBL_PURE_VIRTUALvirtual void def(return_type &d) const VDBL_PURE_-VIRTUALvirtual void get_ptr(return_type const ∗&c) const VDBL_PURE_VIRTUALvirtual void get_copy(return_type ∗&c) const
- virtual void def_copy (return_type ∗&d) const
- virtual void def_copy (_VDBL_alltype_base ∗&v) const
- virtual void get_copy (_VDBL_alltype_base ∗&v) const
- virtual const std::type_info & return_type_id () const
- virtual std::ostream & print_contents (std::ostream &o) const

9.37.1 Detailed Description

template<class _TR> class col_base< _TR >

this is the external name of the column base class

Definition at line 584 of file vdbl_col.h.

9.37.2 Member Function Documentation

9.37.2.1 virtual void _VDBL_colbase< _TR >::def_copy (_VDBL_alltype_base ∗& *v*) const `[inline, virtual, inherited]`

This version of `get_copy` returns a copy of the columns default value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Definition at line 210 of file vdbl_col.h.

9.37.2.2 virtual void _VDBL_colbase< _TR >::def_copy (return_type ∗& *d*) const `[inline, virtual, inherited]`

This function returns a pointer to a copy of the column's default value. The copy of the value is allocated by `new`. It has to be `delete`d by the user to avoid memory leaks.

Definition at line 188 of file vdbl_col.h.

9.37.2.3 virtual _Self∗ const _VDBL_row∗ _r virtual VDBL_PURE_VIRTUAL void _VDBL_colbase< _TR >::get (return_type & *c*) const `[inline, virtual, inherited]`

This function stores a copy of the column value into `c`. This function stores a copy of the column default value into `d`. This function sets `c` to a const pointer pointing to the column's actual value. Here, no copying is done. This function returns a pointer to a copy of the column's value. The copy of the value is allocated by `new`. It has to be `delete`d by the user to avoid memory leaks.

Reimplemented in _VDBL_mthdcol< _TC, _TM, _TR >.

Definition at line 162 of file vdbl_col.h.

9.37.2.4 virtual void _VDBL_colbase< _TR >::get_copy (_VDBL_alltype_base ∗& *v*) const `[inline, virtual, inherited]`

This version of `get_copy` returns a copy of the columns value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Definition at line 197 of file vdbl_col.h.

9.37.2.5 virtual _Self∗ _VDBL_colbase< _TR >::new_copy () const `[virtual, inherited]`

`new_copy` is the clone operation for copy-constructor overloading. `setcontext` sets the context for value retrieval.

Reimplemented from __VDBL_colbase.

Reimplemented in _VDBL_mthdcol< _TC, _TM, _TR >.

9.37.2.6 virtual std::ostream& _VDBL_colbase< _TR >::print_contents (std::ostream & *o*) const `[inline, virtual, inherited]`

This function is needed for the operator<< for columns of type `return_type`.

Reimplemented in _VDBL_mthdcol< _TC, _TM, _TR >.

Definition at line 228 of file vdbl_col.h.

9.37.2.7 virtual const std::type_info& _VDBL_colbase< _TR >::return_type_id () const `[inline, virtual, inherited]`

This function returns the `type_info` of the column type. This information is used during run-time type checking.

Definition at line 221 of file vdbl_col.h.

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

- vdbl_col.h

9.38 col_spec Class Reference

column specification

`#include <vdbl_table.h>`

Public Member Functions

- col_spec (const col_spec &__c)
- virtual ~col_spec ()

  - col_spec (const std::string &__s, const _VDBL_col &__c)
  - col_spec (const char *__s, const _VDBL_col &__c)

  - template<class _CR >
    col_spec (const std::string &__s, const _CR &__c)
  - template<class _CR >
    col_spec (const char *__s, const _CR &__c)

9.38.1 Detailed Description

This class is used to specify columns of a table.

Definition at line 1268 of file vdbl_table.h.

9.38.2 Constructor & Destructor Documentation

9.38.2.1 col_spec::col_spec (const std::string & *__s*, const _VDBL_col & *__c*) `[inline]`

constructor building a column description with a name (`__s`) and column data (`__c`).

Definition at line 1279 of file vdbl_table.h.

9.38.2.2 col_spec::col_spec (const char * *__s*, const _VDBL_col & *__c*) `[inline]`

constructor building a column description with a name (`__s`) and column data (`__c`).

Definition at line 1282 of file vdbl_table.h.

9.38.2.3 template<class _CR > col_spec::col_spec (const std::string & *__s*, const _CR & *__c*) `[inline]`

constructor building a column description with a name (`__s`) and arbitrary data (`__c`) which is converted to column data.

Definition at line 1292 of file vdbl_table.h.

9.38.2.4 template<class _CR > col_spec::col_spec (const char * *__s*, const _CR & *__c*) `[inline]`

constructor building a column description with a name (`__s`) and arbitrary data (`__c`) which is converted to column data.

Definition at line 1296 of file vdbl_table.h.

9.38.2.5 col_spec::col_spec (const col_spec & *__c*) `[inline]`

copy constructor

Definition at line 1303 of file vdbl_table.h.

9.38.2.6 virtual col_spec::∼col_spec () `[inline, virtual]`

standard destructor

Definition at line 1308 of file vdbl_table.h.

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

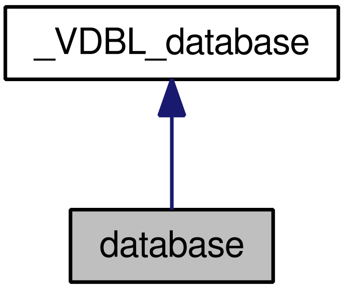
- vdbl_table.h

9.39 database Class Reference

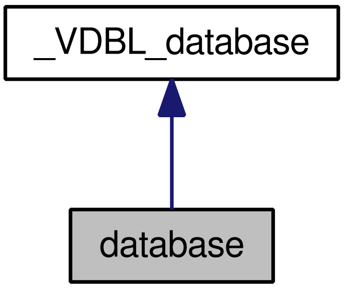
the database class

`#include <vdbl_database.h>`

Inheritance diagram for database:



Collaboration diagram for database:



Public Member Functions

- bool create_table (const std::string &_C_i, const _VDBL_userid &_C_u, const _VDBL_tableflags &__f=_VDBL_tableflags())
- bool create_table (const char *_C_i, const _VDBL_userid &_C_u, const _VDBL_tableflags &__-f=_VDBL_tableflags())
- bool drop_table (const std::string &_C_i, const _VDBL_userid &_C_u)
- bool drop_table (const char *_C_i, const _VDBL_userid &_C_u)
- bool has_table (const std::string &_C_i, const _VDBL_userid &_C_u) const
- bool has_table (const char *_C_i, const _VDBL_userid &_C_u) const
- bool create_view (const std::string &_C_i, const _VDBL_userid &_C_u, const _VDBL_context &_-_c, const std::string &_C_t, const _V_enum &__e)
- bool create_view (const char *_C_i, const _VDBL_userid &_C_u, const _VDBL_context &__c, const std::string &_C_t, const _V_enum &__e)
- bool create_view (const std::string &_C_i, const _VDBL_userid &_C_u, const _VDBL_context &_-_c, const char *_C_t, const _V_enum &__e)
- bool create_view (const char *_C_i, const _VDBL_userid &_C_u, const _VDBL_context &__c, const char *_C_t, const _V_enum &__e)
- bool drop_view (const std::string &_C_i, const _VDBL_userid &_C_u)
- bool drop_view (const char *_C_i, const _VDBL_userid &_C_u)
- bool has_view (const std::string &_C_i, const _VDBL_userid &_C_u) const
- bool has_view (const char *_C_i, const _VDBL_userid &_C_u) const
- viewbase * get_view (const std::string &_C_i, const _VDBL_userid &_C_u) const
- viewbase * get_view (const char *_C_i, const _VDBL_userid &_C_u) const
- table * get_table (const tableid &_C_i, const _VDBL_userid &_C_u) const
- table * get_table (const std::string &_C_i, const _VDBL_userid &_C_u) const
- table * get_table (const char *_C_i, const _VDBL_userid &_C_u) const
- _VDBL_tableid get_tableid (const std::string &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_viewid get_viewid (const std::string &_C_i, const _VDBL_userid &_C_u) const
- bool drop_table (const _D_tables::iterator &__t, const _D_table_names::iterator &__tn, const _-VDBL_userid &_C_u)
- bool drop_table (const _VDBL_tableid &_C_i, const _VDBL_userid &_C_u)
- bool has_table (const _VDBL_tableid &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_table * get_table (const _VDBL_tableid &_C_i, const _VDBL_userid &_C_u) const
- bool drop_view (const _D_views::iterator &__v, const _D_view_names::iterator &__vn, const _-VDBL_userid &_C_u)
- bool drop_view (const _VDBL_viewid &_C_i, const _VDBL_userid &_C_u)
- bool has_view (const _VDBL_viewid &_C_i, const _VDBL_userid &_C_u) const
- _VDBL_view * get_view (const _VDBL_viewid &_C_i, const _VDBL_userid &_C_u) const

Protected Member Functions

- _VDBL_tableid get_tableid ()
- _VDBL_userid get_userid ()
- _VDBL_viewid get_viewid ()

Friends

- class _VDBL_viewdbase

9.39.1 Detailed Description

This is the class describing a whole database including users, tables and views.

Definition at line 831 of file vdbl_database.h.

9.39.2 Member Function Documentation

9.39.2.1 bool database::create_table (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*, const _VDBL_tableflags & *__f* = _VDBL_tableflags`()`) `[inline]`

create a new table

- `_C_i`: name
- `_C_u`: user id
- `__f`: the table flags (if they are not default) return `true`, if creating the table was successful.

Definition at line 859 of file vdbl_database.h.

9.39.2.2 bool database::create_table (const std::string & *_C_i*, const _VDBL_userid & *_C_u*, const _VDBL_tableflags & *__f* = _VDBL_tableflags`()`) `[inline]`

create a new table

- `_C_i`: name
- `_C_u`: user id
- `__f`: the table flags (if they are not default) return `true`, if creating the table was successful.

Reimplemented from _VDBL_database.

Definition at line 849 of file vdbl_database.h.

9.39.2.3 bool database::create_view (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*, const _VDBL_context & *__c*, const char ∗ *_C_t*, const _V_enum & *__e*) `[inline]`

create a new standard view with name `_C_i`, evaluation context `__c`, for table `_C_t`, of type `__e`. return `true` if creating worked, and `false` otherwise.

Definition at line 934 of file vdbl_database.h.

9.39.2.4 bool database::create_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*, const _VDBL_context & *__c*, const char ∗ *_C_t*, const _V_enum & *__e*) `[inline]`

create a new standard view with name `_C_i`, evaluation context `__c`, for table `_C_t`, of type `__e`. return `true` if creating worked, and `false` otherwise.

Definition at line 925 of file vdbl_database.h.

9.39.2.5 bool database::create_view (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*, const _VDBL_context & *__c*, const std::string & *_C_t*, const _V_enum & *__e*) `[inline]`

create a new standard view with name `_C_i`, evaluation context `__c`, for table `_C_t`, of type `__e`. return `true` if creating worked, and `false` otherwise.

Definition at line 916 of file vdbl_database.h.

9.39.2.6 bool database::create_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*, const _VDBL_context & *__c*, const std::string & *_C_t*, const _V_enum & *__e*) `[inline]`

create a new standard view with name `_C_i`, evaluation context `__c`, for table `_C_t`, of type `__e`. return `true` if creating worked, and `false` otherwise.

Reimplemented from _VDBL_database.

Definition at line 907 of file vdbl_database.h.

9.39.2.7 bool _VDBL_database::drop_table (const _VDBL_tableid & *_C_i*, const _VDBL_userid & *_C_u*) `[inline, inherited]`

delete a table, whose table id is provided. return `true`, if deleting the table has worked.

Definition at line 496 of file vdbl_database.h.

9.39.2.8 bool _VDBL_database::drop_table (const _D_tables::iterator & *__t*, const _D_table_names::iterator & *__tn*, const _VDBL_userid & *_C_u*) `[inline, inherited]`

delete a table, internal function. The first argument is the iterator into the table map, the second argument is the iterator into the table names map, return `true`, if deleting the table has worked.

Definition at line 471 of file vdbl_database.h.

9.39.2.9 bool database::drop_table (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*) `[inline]`

delete a table, whose name is provided. return `true`, if deleting the table has worked.

Definition at line 890 of file vdbl_database.h.

9.39.2.10 bool database::drop_table (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) `[inline]`

delete a table, whose name is provided. return `true`, if deleting the table has worked.

Reimplemented from _VDBL_database.

Definition at line 884 of file vdbl_database.h.

9.39.2.11 bool _VDBL_database::drop_view (const _VDBL_viewid & *_C_i*, const _VDBL_userid & *_C_u*) `[inline, inherited]`

delete a view, whose id is provided. return `true`, if deleting the table has worked.

Definition at line 729 of file vdbl_database.h.

9.39.2.12 bool _VDBL_database::drop_view (const _D_views::iterator & *__v*, const _D_view_names::iterator & *__vn*, const _VDBL_userid & *_C_u*) `[inline, inherited]`

delete a view, internal function. The first argument is the iterator into the table map, the second argument is the iterator into the table names map, return `true`, if deleting the table has worked.

Definition at line 704 of file vdbl_database.h.

9.39.2.13 bool database::drop_view (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*) `[inline]`

delete a view, whose name is provided. return `true`, if deleting the view has worked.

Definition at line 953 of file vdbl_database.h.

9.39.2.14 bool database::drop_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) `[inline]`

delete a view, whose name is provided. return `true`, if deleting the view has worked.

Reimplemented from _VDBL_database.

Definition at line 947 of file vdbl_database.h.

9.39.2.15 _VDBL_table∗ _VDBL_database::get_table (const _VDBL_tableid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline, inherited]`

return a pointer to the table with id `_C_i`.

Definition at line 537 of file vdbl_database.h.

9.39.2.16 table∗ database::get_table (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the table with name `_C_i`.

Definition at line 1008 of file vdbl_database.h.

9.39.2.17 table∗ database::get_table (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the table with name `_C_i`.

Reimplemented from _VDBL_database.

Definition at line 1003 of file vdbl_database.h.

9.39.2.18 table∗ database::get_table (const tableid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the table with id `_C_i`.

Definition at line 998 of file vdbl_database.h.

9.39.2.19 _VDBL_tableid _VDBL_database::get_tableid (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline, inherited]`

return the table id for a given name

Definition at line 448 of file vdbl_database.h.

9.39.2.20 _VDBL_tableid _VDBL_database::get_tableid () `[inline, protected, inherited]`

generate a new unique id for tables, views, and users

Definition at line 392 of file vdbl_database.h.

9.39.2.21 _VDBL_userid _VDBL_database::get_userid () `[inline, protected, inherited]`

generate a new unique id for tables, views, and users

Definition at line 393 of file vdbl_database.h.

9.39.2.22 _VDBL_view∗ _VDBL_database::get_view (const _VDBL_viewid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline, inherited]`

return a pointer to the view with id `_C_i`.

Definition at line 769 of file vdbl_database.h.

9.39.2.23 viewbase∗ database::get_view (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the view with name `_C_i`.

Definition at line 973 of file vdbl_database.h.

9.39.2.24 viewbase∗ database::get_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

return a pointer to the view with name `_C_i`.

Reimplemented from _VDBL_database.

Definition at line 968 of file vdbl_database.h.

9.39.2.25 _VDBL_viewid _VDBL_database::get_viewid (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline, inherited]`

return the view id of view `_C_i`.

Definition at line 661 of file vdbl_database.h.

9.39.2.26 _VDBL_viewid _VDBL_database::get_viewid () `[inline, protected, inherited]`

generate a new unique id for tables, views, and users

Definition at line 394 of file vdbl_database.h.

9.39.2.27 bool _VDBL_database::has_table (const _VDBL_tableid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline, inherited]`

check whether the table `_C_i` exists

Definition at line 518 of file vdbl_database.h.

9.39.2.28 bool database::has_table (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

check whether the table `_C_i` exists

Definition at line 900 of file vdbl_database.h.

9.39.2.29 bool database::has_table (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

check whether the table `_C_i` exists

Reimplemented from _VDBL_database.

Definition at line 895 of file vdbl_database.h.

9.39.2.30 bool _VDBL_database::has_view (const _VDBL_viewid & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline, inherited]`

check whether the view with id `_C_i` exists

Definition at line 751 of file vdbl_database.h.

9.39.2.31 bool database::has_view (const char ∗ *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

check whether the view `_C_i` exists

Definition at line 963 of file vdbl_database.h.

9.39.2.32 bool database::has_view (const std::string & *_C_i*, const _VDBL_userid & *_C_u*) const `[inline]`

check whether the view `_C_i` exists

Reimplemented from _VDBL_database.

Definition at line 958 of file vdbl_database.h.

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

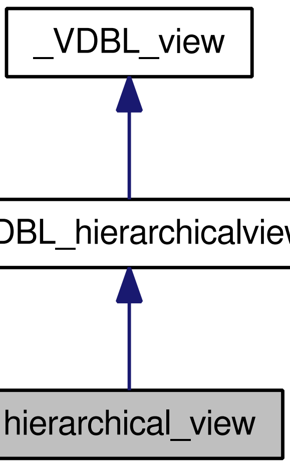
- vdbl_database.h

9.40 hierarchical_view Class Reference

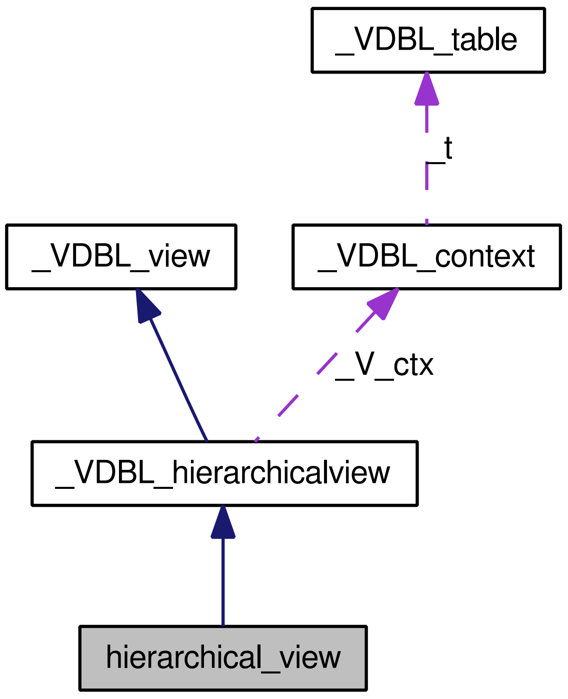
hierarchical view class onto a stack of tables

`#include <vdbl_hrview.h>`

Inheritance diagram for hierarchical_view:



Collaboration diagram for hierarchical_view:



Public Types

- typedef std::pair< std::string, _VDBL_col > _T_colspec

Public Member Functions

- hierarchical_view (const _VDBL_tableid &__ti, _VDBL_table ∗__t, const _VDBL_context &__c, _V_enum __e=V_window)
- hierarchical_view (const hierarchical_view &_v)
- template<class _TR >
  bool get (const tableid &_ti, const rowid &_ri, const colid &_ci, _TR &r) const
- template<class _TR >
  bool get_raw_ptr (const tableid &_ti, const rowid &_ri, const colid &_ci, _TR const ∗&r) const
- void push_table (const _VDBL_tableid &__ti, _VDBL_table ∗__t)
- void push_table (const _VDBL_tableid &__ti, _VDBL_table ∗__t, const std::vector< _VDBL_-rowid > &_rs)
- _VDBL_tableid pop_table ()
- bool insert (const std::vector< _T_colspec > &_row)
- std::ostream & print_col (std::ostream &o, const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, bool &printed) const
- template<class _TR >
  bool get_raw_ptr (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _TR const ∗&r) const
- template<class _TR >
  bool get (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _TR &r) const
- virtual _VDBL_view ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_-view ∗v) VDBL_PURE_VIRTUALvirtual const std

  - template<class _TR >
    bool get (const rowid &_ri, const std::string &_c, _TR &r) const
  - template<class _TR >
    bool get (const rowid &_ri, const char ∗_c, _TR &r) const

Protected Types

- typedef _default_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col ∗ > default_-const_iterator
- typedef _col_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col ∗ > col_const_iterator
- typedef _row_iterator< _VDBL_row, const _VDBL_row &, const _VDBL_row ∗ > row_const_-iterator

Protected Member Functions

- std::triple< _VDBL_tableid, _VDBL_colid, void ∗ > _next_def_col (const _VDBL_tableid &_t, const _VDBL_colid &_c, void ∗_d) const
- void ∗ _copy_def_data (void ∗_d) const
- void ∗ _copy_col_data (void ∗_d) const
- void ∗ _copy_row_data (void ∗_d) const
- void made_change ()

  increment the `change` *counter.*

- unsigned int get_change_ctr () const

  read the change counter

Protected Attributes

- _V_rows _V_r
- _V_cols _V_c
- _V_colxref _V_cx

9.40.1 Detailed Description

This is the hierarchical view. It is an in-memory view onto a stack of VDBL tables.

In a hierachical view the **master table** (the table lowest in the stack) determines the columns valid in this view. The tables upwards in the stack add rows and can change the default values of columns. The upmost definition of a column counts.

Definition at line 738 of file vdbl_hrview.h.

9.40.2 Member Typedef Documentation

9.40.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_view::_T_colspec `[inherited]`

This is the description of one column

Definition at line 61 of file vdbl_view.h.

9.40.2.2 typedef _col_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col*> _VDBL_view::col_const_iterator `[protected, inherited]`

const iterator over all columns

Definition at line 439 of file vdbl_view.h.

9.40.2.3 typedef _default_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col*> _VDBL_view::default_const_iterator `[protected, inherited]`

const iterator over all default columns

Definition at line 301 of file vdbl_view.h.

9.40.2.4 typedef _row_iterator<_VDBL_row, const _VDBL_row&, const _VDBL_row*> _VDBL_view::row_const_iterator `[protected, inherited]`

const iterator over all rows

Definition at line 568 of file vdbl_view.h.

9.40.3 Constructor & Destructor Documentation

9.40.3.1 hierarchical_view::hierarchical_view (const _VDBL_tableid & *__ti*, _VDBL_table ∗ *__t*, const _VDBL_context & *__c*, _V_enum *__e* = `V_window`) `[inline]`

standard constructor which initalizes the `table` and the `tableid` of the master table, the evaluation context, and the view type.

Definition at line 750 of file vdbl_hrview.h.

9.40.3.2 hierarchical_view::hierarchical_view (const hierarchical_view & *_v*) `[inline]`

copy constructor

Definition at line 757 of file vdbl_hrview.h.

9.40.4 Member Function Documentation

9.40.4.1 void∗ _VDBL_hierarchicalview::_copy_col_data (void ∗ *_d*) const `[inline, protected, virtual, inherited]`

This function copies the additional data needed by a _col_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next row for a `_row_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous row for a `_row_iterator`. This function destroys the additional data needed by a _row_iterator

Reimplemented from _VDBL_view.

Definition at line 290 of file vdbl_hrview.h.

9.40.4.2 void∗ _VDBL_hierarchicalview::_copy_def_data (void ∗ *_d*) const `[inline, protected, virtual, inherited]`

This function copies the additional data needed by a _default_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous column for a `_col_iterator`. This function destroys the additional data needed by a _col_iterator

Reimplemented from _VDBL_view.

Definition at line 233 of file vdbl_hrview.h.

9.40.4.3 void∗ _VDBL_hierarchicalview::_copy_row_data (void ∗ *_d*) const `[inline, protected, virtual, inherited]`

This function copies the additional data needed by a _row_iterator This is the fundamental class for iterators over all default columns, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented from _VDBL_view.

Definition at line 327 of file vdbl_hrview.h.

9.40.4.4 std::triple<_VDBL_tableid,_VDBL_colid,void*> _VDBL_hierarchicalview::_next_def_col (const _VDBL_tableid & *_t*, const _VDBL_colid & *_c*, void * *_d*) const `[inline, protected, virtual, inherited]`

This virtual function has to be overloaded by the derived view classes, and it performs the step to the next default of a column a `_default_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous default of a column a `_default_iterator`. This function destroys the additional data needed by a _default_iterator

Reimplemented from _VDBL_view.

Definition at line 181 of file vdbl_hrview.h.

9.40.4.5 template<class _TR > bool _VDBL_hierarchicalview::get (const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, _TR & *r*) const `[inline, inherited]`

get the data from column `_ci` in row `_ri.second` of table `_ri.first`. The data stored in the column must be of type `_TR`.

Definition at line 651 of file vdbl_hrview.h.

9.40.4.6 template<class _TR > bool hierarchical_view::get (const rowid & *_ri*, const char * *_c*, _TR & *r*) const `[inline]`

get the data from column `_c` in row `_ri`. The data stored in the column must be of type `_TR`.

Definition at line 789 of file vdbl_hrview.h.

9.40.4.7 template<class _TR > bool hierarchical_view::get (const rowid & *_ri*, const std::string & *_c*, _TR & *r*) const `[inline]`

get the data from column `_c` in row `_ri`. The data stored in the column must be of type `_TR`.

Definition at line 784 of file vdbl_hrview.h.

9.40.4.8 template<class _TR > bool hierarchical_view::get (const tableid & *_ti*, const rowid & *_ri*, const colid & *_ci*, _TR & *r*) const `[inline]`

get the data from column `_ci` in row `_ri` of table `_ti`. The data stored in the column must be of type `_TR`.

Definition at line 764 of file vdbl_hrview.h.

9.40.4.9 template<class _TR > bool _VDBL_hierarchicalview::get_raw_ptr (const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, _TR const ∗& *r*) const `[inline, inherited]`

get a const ptr to the data from column `_ci` in row `_ri.second` of table `_ri.first`. The data stored in the column must be of type `_TR`. In this function no data copying is done. Note that this function returns a pointer to the columns raw data, so it can only be used to refer to constant columns.

Definition at line 632 of file vdbl_hrview.h.

9.40.4.10 template<class _TR > bool hierarchical_view::get_raw_ptr (const tableid & *_ti*, const rowid & *_ri*, const colid & *_ci*, _TR const ∗& *r*) const `[inline]`

get a const pointer to the data from column `_ci` in row `_ri` of table `_ti`. The data stored in the column must be of type `_TR`. This only works if the column's data is constant. There is no implicit copying performed.

Definition at line 774 of file vdbl_hrview.h.

9.40.4.11 bool _VDBL_hierarchicalview::insert (const std::vector< _T_colspec > & *_row*) `[inline, inherited]`

for now window views can only make changes in the top table in the list of tables

Definition at line 506 of file vdbl_hrview.h.

9.40.4.12 virtual _VDBL_view∗ _VDBL_view::new_copy () const `[inline, virtual, inherited]`

clone operation clone destructor return the column information for column `_C_n`. the return value is a reference to the `type_info` of the column's value type. The reference `_r` contains upon returning whether the function was successful, the column id, and the column flags. return a reference to the `type_-info` of the column's value type. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful. remove the row with it `_ri` from the table return whether the column `_C_n` is defined in the table return table id and column id of column `_C_n` return column name of column `_C_n` return a const reference to the column with column id `_ci` in row `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. Note that in this function the context for the column is NOT set. This method adds the row ids in `_rs` to the visible part of the view. This method removes the row ids in `_rs` from the visible part of the view. print the contents of the column with column id `_ci` in row `_ri.second` of table `ri.first`. If the row is empty and no default is defined, print nothing. return a const reference to the row with row id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return a const reference to the default column with column id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return iterator to first default column return iterator beyond last default column return iterator to first row return iterator beyond last row return iterator to first column return iterator beyond last column return the type of this view

Definition at line 587 of file vdbl_view.h.

9.40.4.13 _VDBL_tableid _VDBL_hierarchicalview::pop_table () `[inline, inherited]`

remove the topmost table from the view, and return its table id.

Definition at line 459 of file vdbl_hrview.h.

9.40.4.14 std::ostream& _VDBL_hierarchicalview::print_col (std::ostream & *o*, const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, bool & *printed*) const `[inline, inherited]`

print the contents od column `_ci` in row `_ri.second` of table `_ri.first`.

Definition at line 606 of file vdbl_hrview.h.

9.40.4.15 void _VDBL_hierarchicalview::push_table (const _VDBL_tableid & *__ti*, _VDBL_table ∗ *__t*, const std::vector< _VDBL_rowid > & *_rs*) `[inline, inherited]`

This pushes a new table onto the top of the hierarchical view stack. Additionally, a subset of the table's rows, which are visible in the view, can be specified.

Definition at line 434 of file vdbl_hrview.h.

9.40.4.16 void _VDBL_hierarchicalview::push_table (const _VDBL_tableid & *__ti*, _VDBL_table ∗ *__t*) `[inline, inherited]`

This pushes a new table onto the top of the hierarchical view stack.

Definition at line 409 of file vdbl_hrview.h.

9.40.5 Member Data Documentation

9.40.5.1 _V_cols _VDBL_hierarchicalview::_V_c `[protected, inherited]`

This contains all columns of the view

Definition at line 90 of file vdbl_hrview.h.

9.40.5.2 _V_colxref _VDBL_hierarchicalview::_V_cx `[protected, inherited]`

This is the cross reference: view col id -> <tableid, real col id>

Definition at line 94 of file vdbl_hrview.h.

9.40.5.3 _V_rows _VDBL_hierarchicalview::_V_r `[protected, inherited]`

This contains all rows of the view

Definition at line 86 of file vdbl_hrview.h.

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

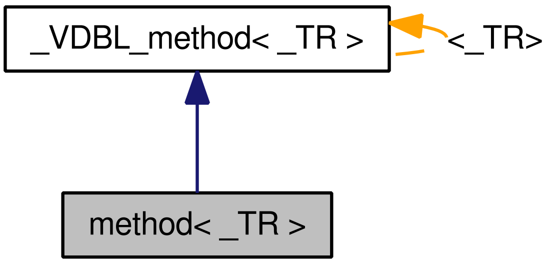
- vdbl_hrview.h

9.41 method< _TR > Class Template Reference

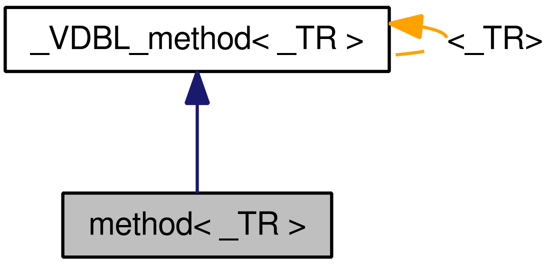
base class for methods usable in `method` columns.

`#include <vdbl_method.h>`

Inheritance diagram for method< _TR >:



Collaboration diagram for method< _TR >:



Public Types

- typedef _TR return_type

Public Member Functions

- virtual return_type operator() () const VDBL_PURE_VIRTUALvirtual return_type def() const VDBL_PURE_VIRTUALvirtual void setcontext(const context ∗_c

9.41.1 Detailed Description

template<class _TR> class method< _TR >

This is the base class, from which all methods should be derived that are used in `method_col` columns. Its virtual methods are those required from a method used for computing column names dynamically. Such a method is a function object with two additional methods described below.

Definition at line 101 of file vdbl_method.h.

9.41.2 Member Typedef Documentation

9.41.2.1 template<class _TR > typedef _TR method< _TR >::return_type

This is the type of the return value of the evaluation methods. Note that this type has to coincide with the column type.

Reimplemented from _VDBL_method< _TR >.

Definition at line 108 of file vdbl_method.h.

9.41.3 Member Function Documentation

9.41.3.1 virtual return_type _VDBL_method< _TR >::operator() () const `[virtual, inherited]`

compute the value compute the default value set the evaluation context and the evaluation row.

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

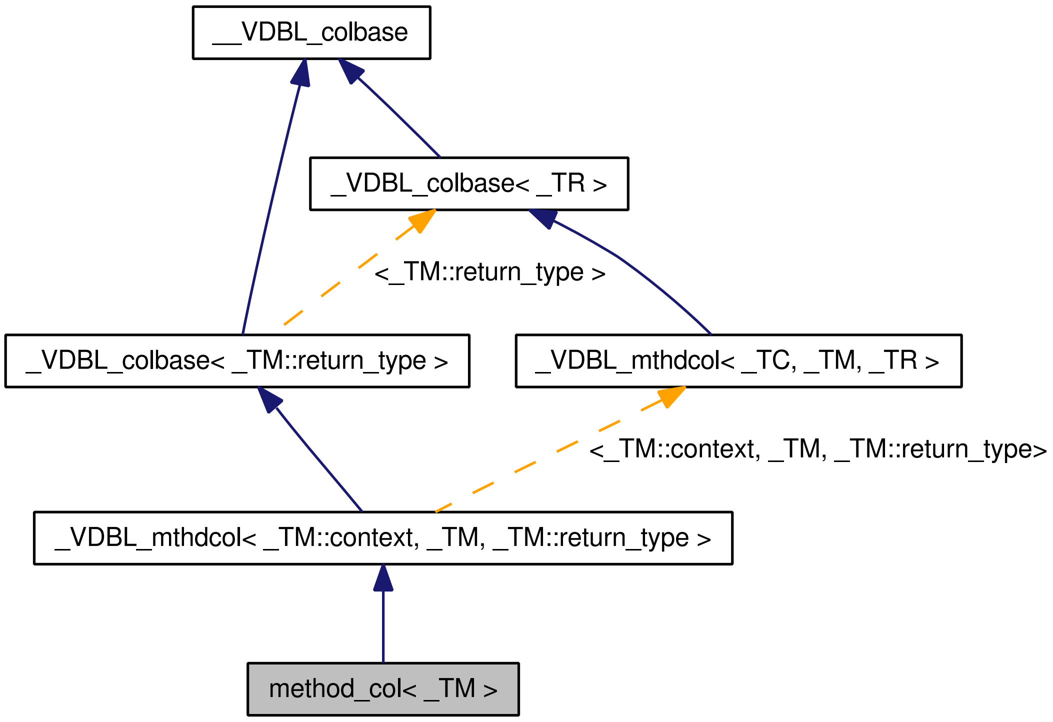
- vdbl_method.h

9.42 method_col< _TM > Class Template Reference

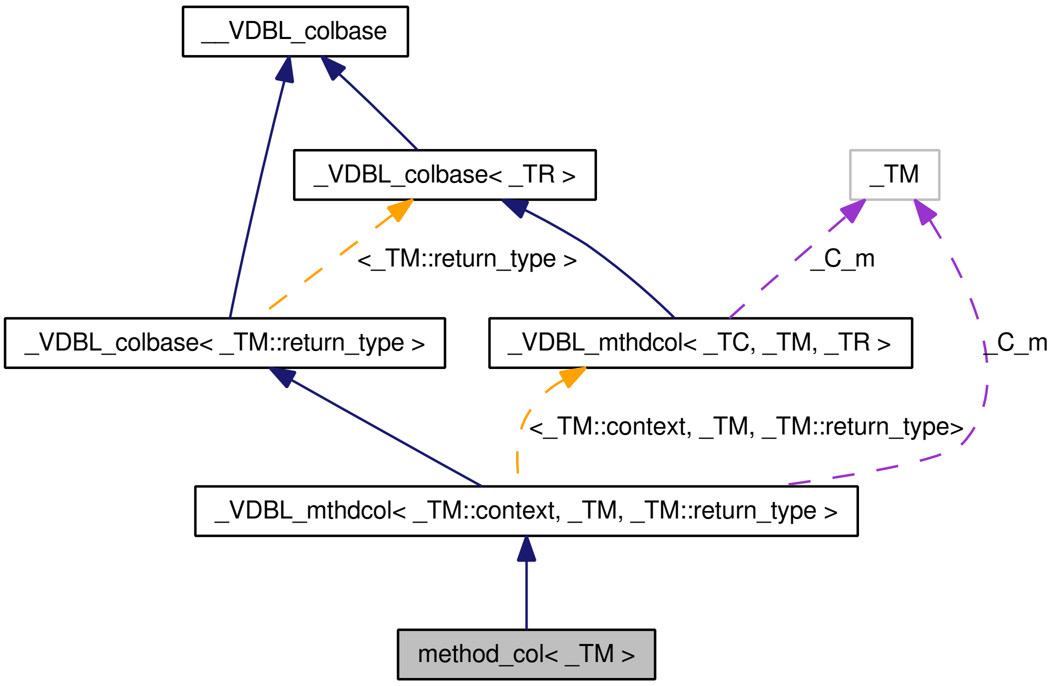
external name for computed columns

```
#include <vdbl_col.h>
```

Inheritance diagram for method_col< _TM >:


__VDBL_colbase
_VDBL_colbase< _TR >
<_TM::return_type >
_VDBL_colbase< _TM::return_type >
_VDBL_mthdcol< _TC, _TM, _TR >
<_TM::context, _TM, _TM::return_type>
_VDBL_mthdcol< _TM::context, _TM, _TM::return_type >
method_col< _TM >

Collaboration diagram for method_col< _TM >:


__VDBL_colbase
_VDBL_colbase< _TR >
_TM
<_TM::return_type >
_C_m
_VDBL_colbase< _TM::return_type >
_VDBL_mthdcol< _TC, _TM, _TR >
_C_m
<_TM::context, _TM, _TM::return_type>
_VDBL_mthdcol< _TM::context, _TM, _TM::return_type >
method_col< _TM >

Public Types

- typedef _TM::return_type return_type

  `return_type` *is the type of object stored*

Public Member Functions

- _Self ∗ new_copy () const
- virtual void setcontext (const context ∗_c, const _VDBL_row ∗_r)
- void get (type &c) const

  the function object provides us with the retrieval method

- void get_ptr (type const ∗&c) const

  there is no way to get a pointer to the method's result properly

- void def (type &d) const

  the default value might be different, and might be computed differently

- virtual void get_copy (_VDBL_alltype_base ∗&v) const
- void def_copy (return_type ∗&d) const
- virtual void def_copy (_VDBL_alltype_base ∗&v) const
- std::ostream & print_contents (std::ostream &o) const
- virtual const std::type_info & return_type_id () const

9.42.1 Detailed Description

template<class _TM> class method_col< _TM >

this is the external name of the standard column for methods.

a method can only be used if it provides at least the following

- type definitions: # context specifies the context class for evaluation # return_type specifies the return value type of the evaluation
- methods: # const return_type& operator()() const for evaluation # const return_type& def() const for evaluation of the default value # void setcontext(const context∗ _c, const _VDBL_row∗ _r) for setting the evaluation context

usually, a method will be a class derived from the `method` base class.

Definition at line 630 of file vdbl_col.h.

9.42.2 Member Function Documentation

9.42.2.1 virtual void _VDBL_colbase< _TM::return_type >::def_copy (_VDBL_alltype_base ∗& *v*) const `[inline, virtual, inherited]`

This version of `get_copy` returns a copy of the columns default value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Definition at line 210 of file vdbl_col.h.

9.42.2.2 void _VDBL_mthdcol< _TM::context , _TM , _TM::return_type >::def_copy (return_type ∗& *d*) const `[inline, virtual, inherited]`

Reimplemented from _VDBL_colbase< _TM::return_type >.

Definition at line 558 of file vdbl_col.h.

9.42.2.3 virtual void _VDBL_colbase< _TM::return_type >::get_copy (_VDBL_alltype_base ∗& *v*) const `[inline, virtual, inherited]`

This version of `get_copy` returns a copy of the columns value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Definition at line 197 of file vdbl_col.h.

9.42.2.4 _Self∗ _VDBL_mthdcol< _TM::context , _TM , _TM::return_type >::new_copy () const `[inline, virtual, inherited]`

`new_copy` is the clone operation for copy-constructor overloading. `setcontext` sets the context for value retrieval.

Reimplemented from _VDBL_colbase< _TM::return_type >.

Definition at line 534 of file vdbl_col.h.

9.42.2.5 std::ostream& _VDBL_mthdcol< _TM::context , _TM , _TM::return_type >::print_contents (std::ostream & *o*) const `[inline, virtual, inherited]`

This function is needed for the operator<< for columns of type `return_type`.

Reimplemented from _VDBL_colbase< _TM::return_type >.

Definition at line 561 of file vdbl_col.h.

9.42.2.6 virtual const std::type_info& _VDBL_colbase< _TM::return_type >::return_type_id () const `[inline, virtual, inherited]`

This function returns the `type_info` of the column type. This information is used during run-time type checking.

Definition at line 221 of file vdbl_col.h.

9.42.2.7 virtual void _VDBL_mthdcol< _TM::context , _TM , _TM::return_type >::setcontext (const context ∗ *_c*, const _VDBL_row ∗ *_r*) `[inline, virtual, inherited]`

for setting the context, the setcontext method of the function object is used.

Definition at line 540 of file vdbl_col.h.

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

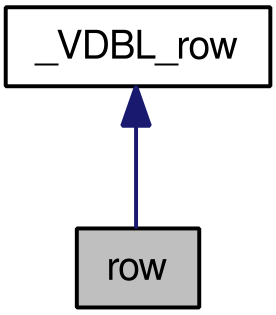
- vdbl_col.h

9.43 row Class Reference

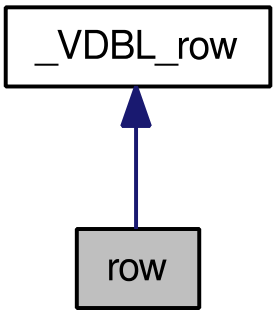
class implementing table rows

`#include <vdbl_row.h>`

Inheritance diagram for row:



Collaboration diagram for row:



Public Member Functions

- row ()
- row (const row &__r)

- row (const _Base &__r)
- row & operator= (const row &_x)
- row & operator= (const _Base &_x)
- const _VDBL_col & get_col (const _VDBL_colid &_id, bool &error) const
- _VDBL_col & get_col (const _VDBL_colid &_id, bool &error)
- bool has_col (const _VDBL_colid &_id) const
- bool insert (const _VDBL_colid &_id, const _VDBL_col &_col)
- bool drop (const _VDBL_colid &_id)
- void update (const _VDBL_colid &_id, const _VDBL_col &_col)

9.43.1 Detailed Description

This class implements the rows of a table

Definition at line 180 of file vdbl_row.h.

9.43.2 Constructor & Destructor Documentation

9.43.2.1 row::row () `[inline]`

standard constructor

Definition at line 191 of file vdbl_row.h.

9.43.2.2 row::row (const row & *__r*) `[inline]`

copy constructor

Definition at line 195 of file vdbl_row.h.

9.43.2.3 row::row (const _Base & *__r*) `[inline]`

copy constructor from internal class

Definition at line 199 of file vdbl_row.h.

9.43.3 Member Function Documentation

9.43.3.1 bool _VDBL_row::drop (const _VDBL_colid & *_id*) `[inline, inherited]`

remove the column with id `_id` from this row. Return `true` if erasing was successful, and `false` if the column does not exist.

Definition at line 149 of file vdbl_row.h.

9.43.3.2 _VDBL_col& _VDBL_row::get_col (const _VDBL_colid & *_id*, bool & *error*) `[inline, inherited]`

get a reference to the column with id `_id`. If the column existed, `error` will be `false`, otherwise `error` will be `true`.

Definition at line 103 of file vdbl_row.h.

9.43.3.3 const _VDBL_col& _VDBL_row::get_col (const _VDBL_colid & *_id*, bool & *error*) const `[inline, inherited]`

get a const reference to the column with id `_id`. If the column existed, `error` will be `false`, otherwise `error` will be `true`.

Definition at line 82 of file vdbl_row.h.

9.43.3.4 bool _VDBL_row::has_col (const _VDBL_colid & *_id*) const `[inline, inherited]`

return whether a column with id `_id` exists in this row.

Definition at line 122 of file vdbl_row.h.

9.43.3.5 bool _VDBL_row::insert (const _VDBL_colid & *_id*, const _VDBL_col & *_col*) `[inline, inherited]`

insert the new column `_col` with id `_id` in this row. If this id exists, return `false`, otherwise return `true`.

Definition at line 132 of file vdbl_row.h.

9.43.3.6 row& row::operator= (const _Base & *_x*) `[inline]`

assignment operator from internal class

Definition at line 212 of file vdbl_row.h.

9.43.3.7 row& row::operator= (const row & *_x*) `[inline]`

assignment operator

Definition at line 204 of file vdbl_row.h.

9.43.3.8 void _VDBL_row::update (const _VDBL_colid & *_id*, const _VDBL_col & *_col*) `[inline, inherited]`

update the column with id `_id` with the value `_col`. If the column does not yet exist, insert it. Otherwise, change its value.

Definition at line 166 of file vdbl_row.h.

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

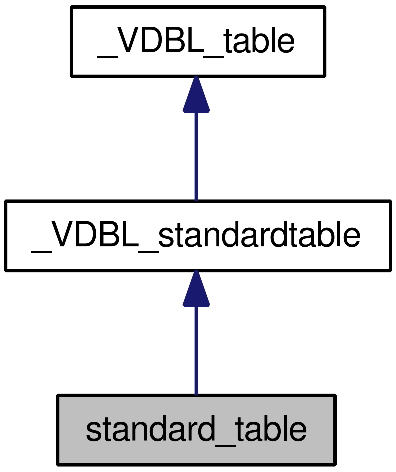
- vdbl_row.h

9.44 standard_table Class Reference

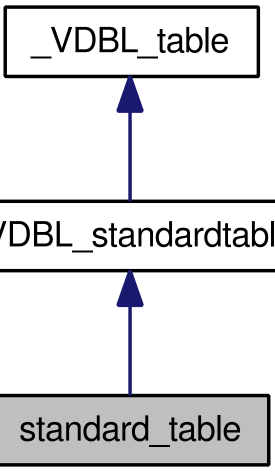
standard table of a database

`#include <vdbl_table.h>`

Inheritance diagram for standard_table:



Collaboration diagram for standard_table:



Public Types

- typedef std::pair< std::string, _VDBL_col > _T_colspec
- typedef std::pair< const std::string ∗, const _VDBL_col ∗ > _T_ptrcolspec
- typedef _Base::col_const_iterator col_const_iterator
- typedef _row_iterator< _VDBL_rowid, const _VDBL_rowid &, const _VDBL_rowid ∗ > row_-const_iterator

Public Member Functions

- standard_table ()
- standard_table (const standard_table &__t)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 >
  standard_table (const __SequenceCtr< std::triple< std::string, _VDBL_col, _VDBL_colflags >, Allocator1 > &__cc)
- virtual ∼standard_table ()
- virtual standard_table ∗ new_copy ()
- virtual void destroy_copy (standard_table ∗t)
- bool insert (const std::vector< _T_ptrcolspec > &_row, rowid &_ri)
- bool insert (const std::vector< _T_ptrcolspec > &_row)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 >
  bool insert_row (const __SequenceCtr< col_spec, Allocator1 > &_row, rowid &_ri)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 >
  bool insert_row (const __SequenceCtr< col_spec, Allocator1 > &_row)
- template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn >
  bool insert_row (const __SequenceCtrOut< __SequenceCtrIn< col_spec, AllocatorIn >, AllocatorOut > &_rows)
- bool update (const std::vector< _T_ptrcolspec > &_row, _VDBL_rowid &_ri)
- template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 >
  bool update_row (const __SequenceCtr< col_spec, Allocator1 > &_row, rowid &_ri)
- const row & get_row (const rowid &_ri, bool &error) const

- const row ∗ get_row_ptr (const rowid &_ri) const
- row & get_row (const rowid &_ri, bool &error)
- std::string get_colname (const colid &_C_i) const
- row_const_iterator row_begin () const
- row_const_iterator row_end () const
- const std::type_info & get_colinfo (const std::string &_C_n, std::triple< bool, _VDBL_colid, _-VDBL_colflags > &_r) const
- _VDBL_colid get_col_id (const std::string &_C_n) const
- std::string get_col_name (const _VDBL_colid &_C_id) const
- virtual _VDBL_table ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_-table ∗t) VDBL_PURE_VIRTUALvirtual bool add_col(const std
- virtual void destroy_copy (_VDBL_table ∗t)
- template<class _VALclass >
  bool retrieve (const _VDBL_rowid &_r, const _VDBL_colid &_c, const _VDBL_context ∗_ctx, _VALclass ∗&_val) const
- std::pair< std::string, _VDBL_colid > _next_col (const std::pair< std::string, _VDBL_colid > &_-ci) const
- _VDBL_rowid _next_row (const _VDBL_rowid &_ci) const
- template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn >
  bool insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > &_rows)

  - template<class _CB >
    bool add_col (const char ∗_C_n, const _CB &__c, const _VDBL_colflags &__f)
  - template<class _CB >
    bool add_col (const std::string &_C_n, const _CB &__c, const _VDBL_colflags &__f)

  - colid get_colid (const std::string &_C_n) const
  - colid get_colid (const char ∗_C_n) const

Protected Member Functions

- void made_change ()

  - _VDBL_colid get_colid ()
  - _VDBL_rowid get_rowid ()

9.44.1 Detailed Description

This class describes the standard table of a database, consisting of rows and columns.

Definition at line 1317 of file vdbl_table.h.

9.44.2 Member Typedef Documentation

9.44.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_standardtable::_T_colspec [inherited]

specifier of one column, a pair of column name (`string`) and entry (`_VDBL_col`).

Reimplemented from _VDBL_table.

Definition at line 591 of file vdbl_table.h.

9.44.2.2 typedef std::pair<const std::string∗,const _VDBL_col∗> _VDBL_standardtable::_T_-ptrcolspec `[inherited]`

specifier of pointers to one column, a pair of column name (`string∗`) and entry (`_VDBL_col∗`).

Reimplemented from _VDBL_table.

Definition at line 596 of file vdbl_table.h.

9.44.2.3 typedef _Base::col_const_iterator _VDBL_standardtable::col_const_iterator `[inherited]`

const iterator over all columns

Reimplemented from _VDBL_table.

Definition at line 601 of file vdbl_table.h.

9.44.2.4 typedef _row_iterator<_VDBL_rowid, const _VDBL_rowid&, const _VDBL_rowid∗> _VDBL_table::row_const_iterator `[inherited]`

const iterator over all rows

Definition at line 321 of file vdbl_table.h.

9.44.3 Constructor & Destructor Documentation

9.44.3.1 standard_table::standard_table () `[inline]`

standard constructor

Definition at line 1330 of file vdbl_table.h.

9.44.3.2 standard_table::standard_table (const standard_table & *__t*) `[inline]`

copy constructor

Definition at line 1334 of file vdbl_table.h.

9.44.3.3 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 > standard_table::standard_table (const __SequenceCtr< std::triple< std::string, _VDBL_col, _VDBL_colflags >, Allocator1 > & *__cc*) `[inline]`

constructor defining a table using a list of columns. This list can be contained in any STL sequence container.

Definition at line 1342 of file vdbl_table.h.

9.44.3.4 virtual standard_table::~standard_table () `[inline, virtual]`

standard destructor

Definition at line 1348 of file vdbl_table.h.

9.44.4 Member Function Documentation

9.44.4.1 std::pair<std::string,_VDBL_colid> _VDBL_standardtable::_next_col (const std::pair< std::string, _VDBL_colid > & *_ci*) const `[inline, virtual, inherited]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous column for a `_col_iterator`. return iterator to first column return iterator beyond last column This is the fundamental class for iterators over all rows, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented from _VDBL_table.

Definition at line 1174 of file vdbl_table.h.

9.44.4.2 _VDBL_rowid _VDBL_standardtable::_next_row (const _VDBL_rowid & *_ci*) const `[inline, virtual, inherited]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next row for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous row for a `_col_iterator`. return iterator to first row return iterator beyond last row construct a new copy of a row iterator standard constructor

Reimplemented from _VDBL_table.

Definition at line 1209 of file vdbl_table.h.

9.44.4.3 template<class _CB > bool standard_table::add_col (const std::string & *_C_n*, const _CB & *__c*, const _VDBL_colflags & *__f*) `[inline]`

add a new column of name `_C_n`, with data `__c`, and column flags `__f`. The function returns `true`, if adding the column was successful.

Definition at line 1371 of file vdbl_table.h.

9.44.4.4 template<class _CB > bool standard_table::add_col (const char $*$ *_C_n*, const _CB & *__c*, const _VDBL_colflags & *__f*) `[inline]`

add a new column of name `_C_n`, with data `__c`, and column flags `__f`. The function returns `true`, if adding the column was successful.

Definition at line 1367 of file vdbl_table.h.

9.44.4.5 virtual void _VDBL_standardtable::destroy_copy (_VDBL_table $*$ *t*) `[inline, virtual, inherited]`

clone destructor method

Definition at line 717 of file vdbl_table.h.

9.44.4.6 virtual void standard_table::destroy_copy (standard_table $*$ *t*) `[inline, virtual]`

clone destructor method

Definition at line 1358 of file vdbl_table.h.

9.44.4.7 _VDBL_colid _VDBL_standardtable::get_col_id (const std::string & *_C_n*) const `[inline, inherited]`

return the column id of column `_C_n`

Definition at line 637 of file vdbl_table.h.

9.44.4.8 std::string _VDBL_standardtable::get_col_name (const _VDBL_colid & *_C_id*) const `[inline, inherited]`

return the column name of column `_C_id`

Definition at line 650 of file vdbl_table.h.

9.44.4.9 _VDBL_colid _VDBL_table::get_colid () `[inline, protected, inherited]`

generate new unique id's for rows and columns

Definition at line 99 of file vdbl_table.h.

9.44.4.10 colid standard_table::get_colid (const char ∗ *_C_n*) const [inline]

return the column id of column _C_n

Definition at line 1506 of file vdbl_table.h.

9.44.4.11 colid standard_table::get_colid (const std::string & *_C_n*) const [inline]

return the column id of column _C_n

Definition at line 1503 of file vdbl_table.h.

9.44.4.12 const std::type_info& _VDBL_standardtable::get_colinfo (const std::string & *_C_n*, std::triple< bool, _VDBL_colid, _VDBL_colflags > & *_r*) const [inline, virtual, inherited]

return the column information for column _C_n. the return value is a referenc to the type_info of the column's value type. The reference _r contains upon returning whether the function was successful, the column id, and the column flags. what was the id of the last change to the table

Reimplemented from _VDBL_table.

Definition at line 614 of file vdbl_table.h.

9.44.4.13 std::string standard_table::get_colname (const colid & *_C_i*) const [inline]

return the column name of column _C_n

Definition at line 1513 of file vdbl_table.h.

9.44.4.14 row& standard_table::get_row (const rowid & *_ri*, bool & *error*) [inline]

return a reference to the row with id _ri. If an error occurs, set error to true, otherwise to false.

Definition at line 1496 of file vdbl_table.h.

9.44.4.15 const row& standard_table::get_row (const rowid & *_ri*, bool & *error*) const [inline]

return a const reference to the row with id _ri. If an error occurs, set error to true, otherwise to false.

Definition at line 1482 of file vdbl_table.h.

9.44.4.16 const row∗ standard_table::get_row_ptr (const rowid & *_ri*) const `[inline]`

return a const pointer to the row with id `_ri`. If an error occurs, return NULL

Definition at line 1489 of file vdbl_table.h.

9.44.4.17 _VDBL_rowid _VDBL_table::get_rowid () `[inline, protected, inherited]`

generate new unique id's for rows and columns

Definition at line 100 of file vdbl_table.h.

9.44.4.18 bool standard_table::insert (const std::vector< _T_ptrcolspec > & *_row*) `[inline]`

insert a new row of specification `_row` into the table. The function returns `true`, if inserting was successful.

Reimplemented from _VDBL_standardtable.

Definition at line 1388 of file vdbl_table.h.

9.44.4.19 bool standard_table::insert (const std::vector< _T_ptrcolspec > & *_row*, rowid & *_ri*) `[inline]`

insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful.

Definition at line 1381 of file vdbl_table.h.

9.44.4.20 template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn > bool _VDBL_table::insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > & *_rows*) `[inline, inherited]`

insert a many new rows of specifications `_rows` into the table. The list of rows can be contained in any sequential STL container, which holds any other sequential STL container of column entries. The function returns `true`, if inserting was successful for all rows.

Definition at line 473 of file vdbl_table.h.

9.44.4.21 template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn > bool standard_table::insert_row (const __SequenceCtrOut< __SequenceCtrIn< col_spec, AllocatorIn >, AllocatorOut > & *_rows*) `[inline]`

insert a many new rows of specifications `_rows` into the table. The list of rows can be contained in any sequential STL container, which holds any other sequential STL container of column entries. The function returns `true`, if inserting was successful for all rows.

Definition at line 1434 of file vdbl_table.h.

9.44.4.22 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 > bool standard_table::insert_row (const __SequenceCtr< col_spec, Allocator1 > & *_row*) `[inline]`

insert a new row of specification `_row` into the table. Take any sequential STL container to hold the row entries of the column. The function returns `true`, if inserting was successful.

Definition at line 1418 of file vdbl_table.h.

9.44.4.23 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 > bool standard_table::insert_row (const __SequenceCtr< col_spec, Allocator1 > & *_row*, rowid & *_ri*) `[inline]`

insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. Take any sequential STL container to hold the row entries of the column. The function returns `true`, if inserting was successful.

Definition at line 1400 of file vdbl_table.h.

9.44.4.24 void _VDBL_table::made_change () `[inline, protected, inherited]`

increment the `last_change` counter.

Definition at line 106 of file vdbl_table.h.

9.44.4.25 virtual _VDBL_table∗ _VDBL_table::new_copy () const `[inline, virtual, inherited]`

clone method clone destructor method add a new column of name `_C_n`, which contains `__c`, and has column flags `__f`. The function returns `true`, if adding the column was successful. modify the column of name `_C_n`, to new contents `__c`, and new column flags `__f`. The function returns `true`, if modifying was successful. modify the contents of column of name `_C_n` The function returns `true`, if modifying was successful. modify the column flags of column of name `_C_n` The function returns `true`, if modifying was successful. remove the column `_C_n` from the table The function returns `true`, if erasing was successful.

rename the column _C_old to new name _C_new. The function returns true, if renaming was successful. insert a new row of specification _row into the table, and return the row id of the newly created row in _r. The function returns true, if inserting was successful. insert a new row of specification _row into the table. The function returns true, if inserting was successful. insert a new row of specification _row into the table, and return the row id of the newly created row in _r. Take any sequential STL container to hold the row entries of the column. The function returns true, if inserting was successful.

Definition at line 380 of file vdbl_table.h.

9.44.4.26 virtual standard_table∗ standard_table::new_copy () [inline, virtual]

clone method

Reimplemented from _VDBL_standardtable.

Definition at line 1353 of file vdbl_table.h.

9.44.4.27 template<class _VALclass > bool _VDBL_standardtable::retrieve (const _VDBL_rowid & *_r*, const _VDBL_colid & *_c*, const _VDBL_context ∗ *_ctx*, _VALclass ∗& *_val*) const [inline, inherited]

retrieve the value of column (id = _c) in row (id = _r) under evaluation context _ctx to _val. Return whether retrieval was successful. If there is no defined value in row _r for column _c, then return the default value of column _c, if defined.

Definition at line 1105 of file vdbl_table.h.

9.44.4.28 row_const_iterator standard_table::row_begin () const [inline]

return iterator to first row

Reimplemented from _VDBL_standardtable.

Definition at line 1519 of file vdbl_table.h.

9.44.4.29 row_const_iterator standard_table::row_end () const [inline]

return iterator beyond last row

Reimplemented from _VDBL_standardtable.

Definition at line 1523 of file vdbl_table.h.

9.44.4.30 bool standard_table::update (const std::vector< _T_ptrcolspec > & *_row*, _VDBL_rowid & *_ri*) [inline]

update the row with row id `_r` to the new of specification `_row` The function returns `true`, if updating was successful. The function returns `true` if updating was successful.

Reimplemented from _VDBL_standardtable.

Definition at line 1453 of file vdbl_table.h.

9.44.4.31 template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1 > bool standard_table::update_row (const __SequenceCtr< col_spec, Allocator1 > & *_row*, rowid & *_ri*) `[inline]`

update the row with row id `_r` to the new of specification `_row` The function returns `true`, if updating was successful. The function returns `true` if updating was successful. Take any sequential STL container to hold the row entries of the column. The function returns `true`, if inserting was successful.

Definition at line 1468 of file vdbl_table.h.

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

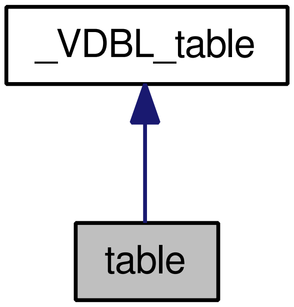
- vdbl_table.h

9.45 table Class Reference

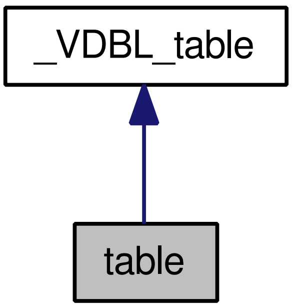
base class for tables in a database

`#include <vdbl_table.h>`

Inheritance diagram for table:



Collaboration diagram for table:



Public Types

- typedef std::pair< std::string, _VDBL_col > _T_colspec
- typedef std::pair< const std::string ∗, const _VDBL_col ∗ > _T_ptrcolspec
- typedef _col_iterator< std::pair< std::string, _VDBL_colid >, const std::pair< std::string, _VDBL_colid > &, const std::pair< std::string, _VDBL_colid > ∗ > col_const_iterator
- typedef _row_iterator< _VDBL_rowid, const _VDBL_rowid &, const _VDBL_rowid ∗ > row_const_iterator

Public Member Functions

- virtual const std::type_info & get_colinfo (const std::string &_C_n, std::triple< bool, _VDBL_colid, _VDBL_colflags > &_r) const VDBL_PURE_VIRTUAL public
- virtual std::pair< std::string, _VDBL_colid > _next_col (const std::pair< std::string, _VDBL_colid > &_ci) const VDBL_PURE_VIRTUALvirtual std
- virtual _VDBL_rowid _next_row (const _VDBL_rowid &_ci) const VDBL_PURE_VIRTUALvirtual _VDBL_rowid _prev_row(const _VDBL_rowid &_ci) const VDBL_PURE_VIRTUALvirtual row_const_iterator row_begin() const VDBL_PURE_VIRTUALvirtual row_const_iterator row_end() const VDBL_PURE_VIRTUALvirtual row_const_iterator ∗row_iterator_copy(const row_const_iterator &) const VDBL_PURE_VIRTUAL public
- virtual _VDBL_table ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_table ∗t) VDBL_PURE_VIRTUALvirtual bool add_col(const std
- template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn >
  bool insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > &_rows)

Protected Member Functions

- void made_change ()

  - _VDBL_colid get_colid ()
  - _VDBL_rowid get_rowid ()

9.45.1 Detailed Description

This is the base class for all tables in a database

Definition at line 1251 of file vdbl_table.h.

9.45.2 Member Typedef Documentation

9.45.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_table::_T_colspec `[inherited]`

specifier of one column, a pair of column name (`string`) and entry (`_VDBL_col`).

Reimplemented in _VDBL_standardtable, and _VDBL_viewtable.

Definition at line 78 of file vdbl_table.h.

9.45.2.2 typedef std::pair<const std::string*,const _VDBL_col*> _VDBL_table::_T_ptrcolspec `[inherited]`

specifier of pointers to one column, a pair of column name (`string*`) and entry (`_VDBL_col*`).

Reimplemented in _VDBL_standardtable.

Definition at line 83 of file vdbl_table.h.

9.45.2.3 typedef _col_iterator<std::pair<std::string,_VDBL_colid>, const std::pair<std::string,_VDBL_colid>&, const std::pair<std::string,_VDBL_colid>*> _VDBL_table::col_const_iterator `[inherited]`

const iterator over all columns

Reimplemented in _VDBL_standardtable.

Definition at line 214 of file vdbl_table.h.

9.45.2.4 typedef _row_iterator<_VDBL_rowid, const _VDBL_rowid&, const _VDBL_rowid*> _VDBL_table::row_const_iterator `[inherited]`

const iterator over all rows

Definition at line 321 of file vdbl_table.h.

9.45.3 Member Function Documentation

9.45.3.1 virtual std::pair<std::string,_VDBL_colid> _VDBL_table::_next_col (const std::pair< std::string, _VDBL_colid > & *_ci*) const `[inline, virtual, inherited]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous column for a `_col_iterator`. return iterator to first column return iterator beyond last column This is the fundamental class for iterators over all rows, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented in _VDBL_standardtable.

Definition at line 221 of file vdbl_table.h.

9.45.3.2 virtual _VDBL_rowid _VDBL_table::_next_row (const _VDBL_rowid & *_ci*) const `[inline, virtual, inherited]`

This virtual function has to be overloaded by the derived table classes, and it performs the step to the next row for a `_col_iterator`. This virtual function has to be overloaded by the derived table classes, and it performs the step to the previous row for a `_col_iterator`. return iterator to first row return iterator beyond last row construct a new copy of a row iterator standard constructor

Reimplemented in _VDBL_standardtable.

Definition at line 328 of file vdbl_table.h.

9.45.3.3 _VDBL_colid _VDBL_table::get_colid () `[inline, protected, inherited]`

generate new unique id's for rows and columns

Definition at line 99 of file vdbl_table.h.

9.45.3.4 virtual const std::type_info& _VDBL_table::get_colinfo (const std::string & *_C_n*, std::triple< bool, _VDBL_colid, _VDBL_colflags > & *_r*) const `[inline, virtual, inherited]`

return the column information for column `_C_n`. the return value is a referenc to the `type_info` of the column's value type. The reference `_r` contains upon returning whether the function was successful, the column id, and the column flags. what was the id of the last change to the table

Reimplemented in _VDBL_standardtable.

Definition at line 116 of file vdbl_table.h.

9.45.3.5 _VDBL_rowid _VDBL_table::get_rowid () `[inline, protected, inherited]`

generate new unique id's for rows and columns

Definition at line 100 of file vdbl_table.h.

9.45.3.6 template<template< class __Tp1, class __AllocTp1 > class __SequenceCtrOut, template< class __Tp2, class __AllocTp2 > class __SequenceCtrIn, class AllocatorOut , class AllocatorIn > bool _VDBL_table::insert_row (const __SequenceCtrOut< __SequenceCtrIn< _T_colspec, AllocatorIn >, AllocatorOut > & *_rows*) `[inline, inherited]`

insert a many new rows of specifications `_rows` into the table. The list of rows can be contained in any sequential STL container, which holds any other sequential STL container of column entries. The function returns `true`, if inserting was successful for all rows.

Definition at line 473 of file vdbl_table.h.

9.45.3.7 void _VDBL_table::made_change () `[inline, protected, inherited]`

increment the `last_change` counter.

Definition at line 106 of file vdbl_table.h.

9.45.3.8 virtual _VDBL_table∗ _VDBL_table::new_copy () const `[inline, virtual, inherited]`

clone method clone destructor method add a new column of name `_C_n`, which contains `__c`, and has column flags `__f`. The function returns `true`, if adding the column was successful. modify the column of name `_C_n`, to new contents `__c`, and new column flags `__f`. The function returns `true`, if modifying was successful. modify the contents of column of name `_C_n` The function returns `true`, if modifying was successful. modify the column flags of column of name `_C_n` The function returns `true`, if modifying was successful. remove the column `_C_n` from the table The function returns `true`, if erasing was successful. rename the column `_C_old` to new name _C_new. The function returns `true`, if renaming was successful. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful. insert a new row of specification `_row` into the table. The function returns `true`, if inserting was successful. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. Take any sequential STL container to hold the row entries of the column. The function returns `true`, if inserting was successful.

Definition at line 380 of file vdbl_table.h.

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

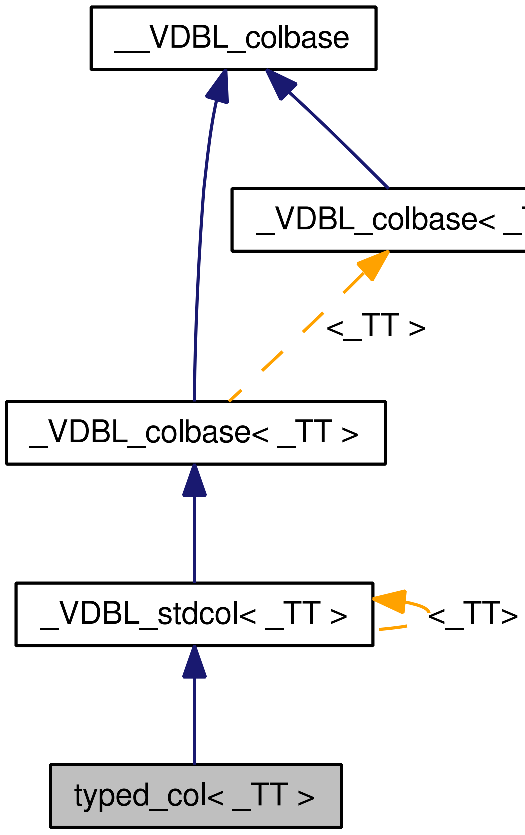
- vdbl_table.h

9.46 typed_col< _TT > Class Template Reference

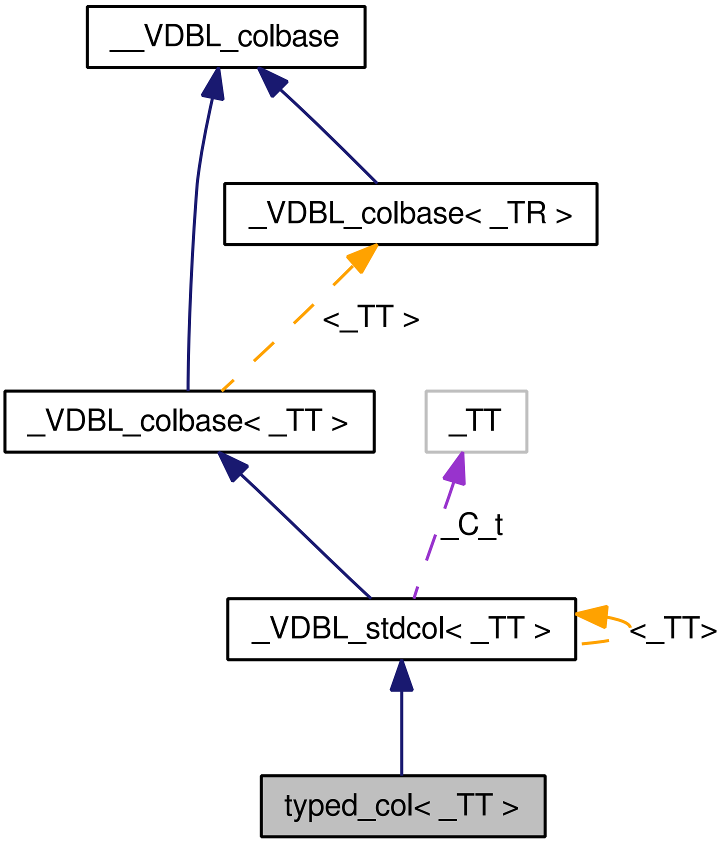
external name for constant data columns

`#include <vdbl_col.h>`

Inheritance diagram for typed_col< _TT >:



Collaboration diagram for typed_col< _TT >:



Public Types

- typedef _TT return_type

  `return_type` *is the type of object stored*

Public Member Functions

- _Self ∗ new_copy () const
- void set (const _Self &_p)
- void set (const type &__t)
- void setcontext (const context ∗_c, const _VDBL_row ∗_r)
- void get (type &c) const
- void def (type &d) const
- void get_copy (_VDBL_alltype_base ∗&v) const
- void def_copy (return_type ∗&d) const
- void def_copy (_VDBL_alltype_base ∗&v) const
- void set_default (const type &__t)
- const type & get_val () const
- std::ostream & print_contents (std::ostream &o) const
- virtual const std::type_info & return_type_id () const

9.46.1 Detailed Description

template<class _TT> class typed_col< _TT >

this is the external name of the standard column for constant data all methods are implemented in class `_VDBL_stdcol<_TT>`.

Definition at line 597 of file vdbl_col.h.

9.46.2 Member Function Documentation

9.46.2.1 void _VDBL_stdcol< _TT >::def (type & *d*) const `[inline, inherited]`

the default for the constant value coincides with the value, since in the table definition the reference object of this class will hold the default, then. There have to be different access methods `get` and `def` for more complicated column types

Definition at line 454 of file vdbl_col.h.

9.46.2.2 void _VDBL_stdcol< _TT >::def_copy (_VDBL_alltype_base *& *v*) const `[inline, virtual, inherited]`

This version of `get_copy` returns a copy of the columns default value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 469 of file vdbl_col.h.

9.46.2.3 void _VDBL_stdcol< _TT >::def_copy (return_type *& *d*) const `[inline, virtual, inherited]`

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 460 of file vdbl_col.h.

9.46.2.4 void _VDBL_stdcol< _TT >::get (type & *c*) const `[inline, virtual, inherited]`

This function stores a copy of the column value into `c`. This function stores a copy of the column default value into `d`. This function sets `c` to a const pointer pointing to the column's actual value. Here, no copying is done. This function returns a pointer to a copy of the column's value. The copy of the value is allocated by `new`. It has to be `deleted` by the user to avoid memory leaks.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 446 of file vdbl_col.h.

9.46.2.5 void _VDBL_stdcol< _TT >::get_copy (_VDBL_alltype_base *& *v*) const `[inline, virtual, inherited]`

This version of `get_copy` returns a copy of the columns value within an `alltype`. This is useful for passing on column values. It can also be used to circumvent the strict run-time type checking. The user is, however, DISCOURAGED to do so.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 462 of file vdbl_col.h.

9.46.2.6 const type& _VDBL_stdcol< _TT >::get_val () const `[inline, inherited]`

get a const reference to the column value

Definition at line 491 of file vdbl_col.h.

9.46.2.7 _Self* _VDBL_stdcol< _TT >::new_copy () const `[inline, virtual, inherited]`

`new_copy` is the clone operation for copy-constructor overloading. `setcontext` sets the context for value retrieval.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 433 of file vdbl_col.h.

9.46.2.8 std::ostream& _VDBL_stdcol< _TT >::print_contents (std::ostream & *o*) const `[inline, virtual, inherited]`

This function is needed for the operator<< for columns of type `return_type`.

Reimplemented from _VDBL_colbase< _TT >.

Definition at line 493 of file vdbl_col.h.

9.46.2.9 virtual const std::type_info& _VDBL_colbase< _TT >::return_type_id () const `[inline, virtual, inherited]`

This function returns the `type_info` of the column type. This information is used during run-time type checking.

Definition at line 221 of file vdbl_col.h.

9.46.2.10 void _VDBL_stdcol< _TT >::set (const type & *__t*) `[inline, inherited]`

set the column value

Definition at line 479 of file vdbl_col.h.

9.46.2.11 void _VDBL_stdcol< _TT >::set (const _Self & *_p*) `[inline, inherited]`

explicit copy operation

Definition at line 438 of file vdbl_col.h.

9.46.2.12 void _VDBL_stdcol< _TT >::set_default (const type & *__t*) `[inline, inherited]`

set the default value for this column. This is actually equivalent to `set`, since default and standard columns coincide for constant values.

Definition at line 486 of file vdbl_col.h.

9.46.2.13 void _VDBL_stdcol< _TT >::setcontext (const context ∗ *_c*, const _VDBL_row ∗ *_r*) `[inline, inherited]`

this method is empty, since constant values are independend of the context.

Definition at line 444 of file vdbl_col.h.

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

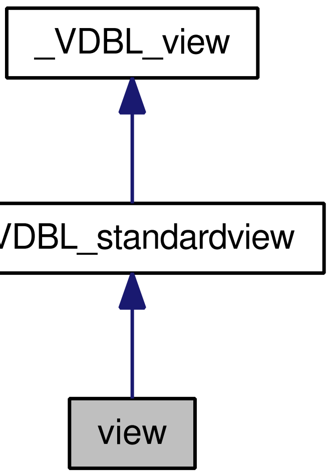
- vdbl_col.h

9.47 view Class Reference

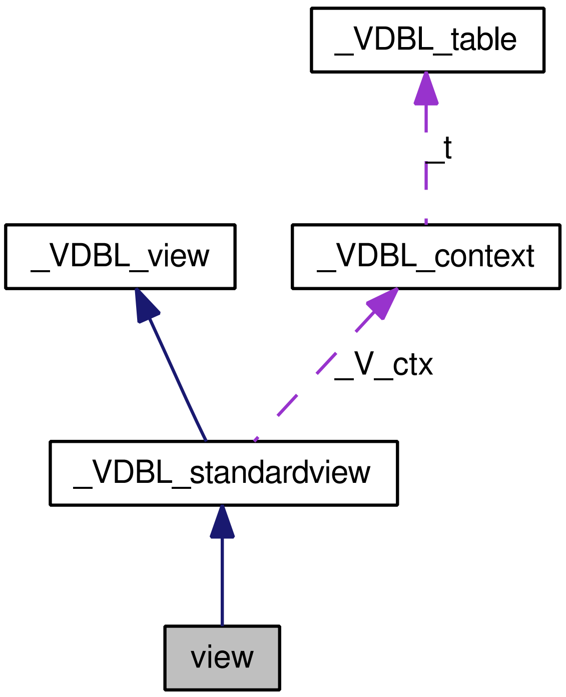
standard view class onto a single table

`#include <vdbl_stview.h>`

Inheritance diagram for view:



Collaboration diagram for view:



Public Types

- typedef _Base::default_const_iterator defaults_const_iterator

  iterator over all default columns

- typedef std::pair< std::string, _VDBL_col > _T_colspec

Public Member Functions

- view (const _VDBL_tableid &__ti, _VDBL_table ∗__t, const _VDBL_context &__c, _V_enum __-e=V_hole)
- view (const view &_v)
- virtual view ∗ new_copy ()
- virtual void destroy_copy (view ∗v)
- template<class _TR >
  bool get (const tableid &_ti, const rowid &_ri, const colid &_ci, _TR &r) const
- template<class _TR >
  bool get_raw_ptr (const tableid &_ti, const rowid &_ri, const colid &_ci, _TR const ∗&r) const
- const col & get_raw_col (const std::pair< tableid, rowid > &_ri, const colid &_ci, row const ∗&_rr, bool &error) const
- virtual _VDBL_view ∗ new_copy () VDBL_PURE_VIRTUALvirtual void destroy_copy(_VDBL_-view ∗v) VDBL_PURE_VIRTUALvirtual const std
- virtual void destroy_copy (_VDBL_view ∗v)
- bool insert (const std::vector< _T_colspec > &_row)
- bool add_visible (const std::vector< _VDBL_rowid > &_rs)
- bool rm_visible (const std::vector< _VDBL_rowid > &_rs)
- std::ostream & print_col (std::ostream &o, const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, bool &printed) const
- template<class _TR >
  bool get (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _TR &r) const
- template<class _TR >
  bool get_raw_ptr (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _TR const ∗&r) const

  - template<class _TR >
    bool get (const rowid &_ri, const std::string &_c, _TR &r) const
  - template<class _TR >
    bool get (const rowid &_ri, const char ∗_c, _TR &r) const

Protected Types

- typedef _default_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col ∗ > default_-const_iterator
- typedef _col_iterator< _VDBL_col, const _VDBL_col &, const _VDBL_col ∗ > col_const_iterator
- typedef _row_iterator< _VDBL_row, const _VDBL_row &, const _VDBL_row ∗ > row_const_-iterator

Protected Member Functions

- std::triple< _VDBL_tableid, _VDBL_colid, void ∗ > _next_def_col (const _VDBL_tableid &_t, const _VDBL_colid &_c, void ∗_d) const
- void ∗ _copy_def_data (void ∗_d) const
- void ∗ _copy_col_data (void ∗_d) const
- void ∗ _copy_row_data (void ∗_d) const
- void made_change ()

  increment the `change` *counter.*

- unsigned int get_change_ctr () const

  read the change counter

Protected Attributes

- _V_rows _V_r
- _V_cols _V_c

9.47.1 Detailed Description

This is the standard view. It is an in-memory view onto a single VDBL table.

Definition at line 704 of file vdbl_stview.h.

9.47.2 Member Typedef Documentation

9.47.2.1 typedef std::pair<std::string,_VDBL_col> _VDBL_view::_T_colspec `[inherited]`

This is the description of one column

Definition at line 61 of file vdbl_view.h.

9.47.2.2 typedef _col_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col*> _VDBL_view::col_const_iterator `[protected, inherited]`

const iterator over all columns

Definition at line 439 of file vdbl_view.h.

9.47.2.3 typedef _default_iterator<_VDBL_col, const _VDBL_col&, const _VDBL_col*> _VDBL_view::default_const_iterator `[protected, inherited]`

const iterator over all default columns

Definition at line 301 of file vdbl_view.h.

9.47.2.4 typedef _row_iterator<_VDBL_row, const _VDBL_row&, const _VDBL_row*> _VDBL_view::row_const_iterator `[protected, inherited]`

const iterator over all rows

Definition at line 568 of file vdbl_view.h.

9.47.3 Constructor & Destructor Documentation

9.47.3.1 view::view (const _VDBL_tableid & __*ti*, _VDBL_table ∗ __*t*, const _VDBL_context & __*c*, _V_enum __*e* = V_hole) [inline]

standard constructor which initalizes the `table` and the `tableid`, the evaluation context, and the view type.

Definition at line 716 of file vdbl_stview.h.

9.47.3.2 view::view (const view & _*v*) [inline]

copy constructor

Definition at line 723 of file vdbl_stview.h.

9.47.4 Member Function Documentation

9.47.4.1 void∗ _VDBL_standardview::_copy_col_data (void ∗ _*d*) const [inline, protected, virtual, inherited]

This function copies the additional data needed by a _col_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next row for a `_row_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous row for a `_row_iterator`. This function destroys the additional data needed by a _row_iterator

Reimplemented from _VDBL_view.

Definition at line 196 of file vdbl_stview.h.

9.47.4.2 void∗ _VDBL_standardview::_copy_def_data (void ∗ _*d*) const [inline, protected, virtual, inherited]

This function copies the additional data needed by a _default_iterator This virtual function has to be overloaded by the derived view classes, and it performs the step to the next column for a `_col_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous column for a `_col_iterator`. This function destroys the additional data needed by a _col_iterator

Reimplemented from _VDBL_view.

Definition at line 146 of file vdbl_stview.h.

9.47.4.3 void∗ _VDBL_standardview::_copy_row_data (void ∗ _*d*) const [inline, protected, virtual, inherited]

This function copies the additional data needed by a _row_iterator This is the fundamental class for iterators over all default columns, defining basic in(de)crementation for overloading, and basic comparison.

Reimplemented from _VDBL_view.

Definition at line 294 of file vdbl_stview.h.

9.47.4.4 std::triple< _VDBL_tableid,_VDBL_colid,void∗> _VDBL_standardview::_next_def_col (const _VDBL_tableid & *_t*, const _VDBL_colid & *_c*, void ∗ *_d*) const `[inline, protected, virtual, inherited]`

This virtual function has to be overloaded by the derived view classes, and it performs the step to the next default of a column a `_default_iterator`. This virtual function has to be overloaded by the derived view classes, and it performs the step to the previous default of a column a `_default_iterator`. This function destroys the additional data needed by a _default_iterator

Reimplemented from _VDBL_view.

Definition at line 102 of file vdbl_stview.h.

9.47.4.5 bool _VDBL_standardview::add_visible (const std::vector< _VDBL_rowid > & *_rs*) `[inline, inherited]`

This method adds the row ids in `_rs` to the visible part of the view.

Definition at line 497 of file vdbl_stview.h.

9.47.4.6 virtual void _VDBL_standardview::destroy_copy (_VDBL_view ∗ *v*) `[inline, virtual, inherited]`

clone destructor

Definition at line 391 of file vdbl_stview.h.

9.47.4.7 virtual void view::destroy_copy (view ∗ *v*) `[inline, virtual]`

clone destructor

Definition at line 732 of file vdbl_stview.h.

9.47.4.8 template<class _TR > bool _VDBL_standardview::get (const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, _TR & *r*) const `[inline, inherited]`

get the data from column `_ci` in row `_ri.second` of table `_ri.first`. The data stored in the column must be of type `_TR`.

Definition at line 559 of file vdbl_stview.h.

9.47.4.9 template<class _TR > bool view::get (const rowid & *_ri*, const char ∗ *_c*, _TR & *r*) const `[inline]`

get the data from column `_c` in row `_ri`. The data stored in the column must be of type `_TR`.

Definition at line 764 of file vdbl_stview.h.

9.47.4.10 template<class _TR > bool view::get (const rowid & *_ri*, const std::string & *_c*, _TR & *r*) const `[inline]`

get the data from column `_c` in row `_ri`. The data stored in the column must be of type `_TR`.

Definition at line 759 of file vdbl_stview.h.

9.47.4.11 template<class _TR > bool view::get (const tableid & *_ti*, const rowid & *_ri*, const colid & *_ci*, _TR & *r*) const `[inline]`

get the data from column `_ci` in row `_ri` of table `_ti`. The data stored in the column must be of type `_TR`.

Definition at line 739 of file vdbl_stview.h.

9.47.4.12 const col& view::get_raw_col (const std::pair< tableid, rowid > & *_ri*, const colid & *_ci*, row const ∗& *_rr*, bool & *error*) const `[inline]`

get a const reference to column `_ci` in row `_ri.second` of table `_ri.first`. If the retrieval was successful, set `error` to , otherwise to `true`.

Definition at line 773 of file vdbl_stview.h.

9.47.4.13 template<class _TR > bool _VDBL_standardview::get_raw_ptr (const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, _TR const ∗& *r*) const `[inline, inherited]`

get a const ptr to the data from column `_ci` in row `_ri.second` of table `_ri.first`. The data stored in the column must be of type `_TR`. In this function no data copying is done. Note that this function returns a pointer to the columns raw data, so it can only be used to refer to constant columns.

Definition at line 581 of file vdbl_stview.h.

9.47.4.14 template<class _TR > bool view::get_raw_ptr (const tableid & *_ti*, const rowid & *_ri*, const colid & *_ci*, _TR const ∗& *r*) const `[inline]`

get a const pointer to the data from column `_ci` in row `_ri` of table `_ti`. The data stored in the column must be of type `_TR`. This only works if the column's data is constant. There is no implicit copying performed.

Definition at line 749 of file vdbl_stview.h.

9.47.4.15 bool _VDBL_standardview::insert (const std::vector< _T_colspec > & *_row*) `[inline, inherited]`

for now window views can only have one table in the list of tables

Definition at line 412 of file vdbl_stview.h.

9.47.4.16 virtual _VDBL_view* _VDBL_view::new_copy () const `[inline, virtual, inherited]`

clone operation clone destructor return the column information for column `_C_n`. the return value is a reference to the `type_info` of the column's value type. The reference `_r` contains upon returning whether the function was successful, the column id, and the column flags. return a reference to the `type_info` of the column's value type. insert a new row of specification `_row` into the table, and return the row id of the newly created row in `_r`. The function returns `true`, if inserting was successful. remove the row with it `_ri` from the table return whether the column `_C_n` is defined in the table return table id and column id of column `_C_n` return column name of column `_C_n` return a const reference to the column with column id `_ci` in row `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. Note that in this function the context for the column is NOT set. This method adds the row ids in `_rs` to the visible part of the view. This method removes the row ids in `_rs` from the visible part of the view. print the contents of the column with column id `_ci` in row `_ri.second` of table `ri.first`. If the row is empty and no default is defined, print nothing. return a const reference to the row with row id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return a const reference to the default column with column id `_ri.second` of table `ri.first`. If successful, set `error` to `false`, and to `true` otherwise. return iterator to first default column return iterator beyond last default column return iterator to first row return iterator beyond last row return iterator to first column return iterator beyond last column return the type of this view

Definition at line 587 of file vdbl_view.h.

9.47.4.17 virtual view* view::new_copy () `[inline, virtual]`

clone operation

Reimplemented from _VDBL_standardview.

Definition at line 728 of file vdbl_stview.h.

9.47.4.18 std::ostream& _VDBL_standardview::print_col (std::ostream & *o*, const std::pair< _VDBL_tableid, _VDBL_rowid > & *_ri*, const _VDBL_colid & *_ci*, bool & *printed*) const `[inline, inherited]`

print the contents of column `_ci` in row `_ri.second` of table `_ri.first`.

Definition at line 536 of file vdbl_stview.h.

9.47.4.19 bool _VDBL_standardview::rm_visible (const std::vector< _VDBL_rowid > & *_rs*) `[inline, inherited]`

This method removes the row ids in `_rs` from the visible part of the view.

Definition at line 514 of file vdbl_stview.h.

9.47.5 Member Data Documentation

9.47.5.1 _V_cols _VDBL_standardview::_V_c `[protected, inherited]`

This contains all columns of the view

Definition at line 88 of file vdbl_stview.h.

9.47.5.2 _V_rows _VDBL_standardview::_V_r `[protected, inherited]`

This contains all rows of the view unless it is a window view

Definition at line 84 of file vdbl_stview.h.

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

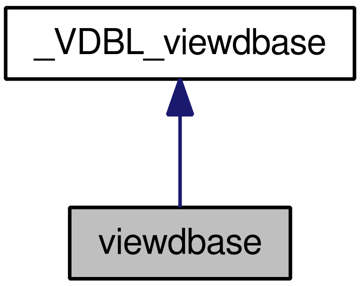
- vdbl_stview.h

9.48 viewdbase Class Reference

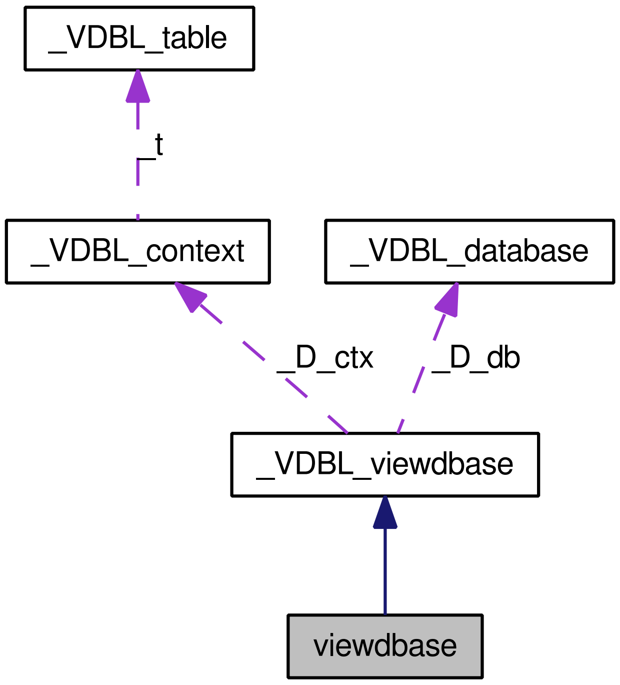
a view to a complete database

`#include <vdbl_viewdbase.h>`

Inheritance diagram for viewdbase:



Collaboration diagram for viewdbase:



Public Member Functions

- bool has_view (const char ∗_C_i) const
- viewbase ∗ get_view (const char ∗_C_i) const
- viewbase ∗ get_view (const std::string &_C_i) const
- viewdbase ()
- viewdbase (const database &db, const userid &uid, const context &c, view_enum e=V_frozen)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  viewdbase (const database &db, const userid &uid, const context &c, const _SqCtr< std::pair< tableid, rowid >, _Al > &__an, view_enum e=V_frozen)
- virtual ∼viewdbase ()
- bool add_table_view (const _VDBL_tableid &__t, const std::string &__s, const _VDBL_context &__c)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  bool add_table_view (const _VDBL_tableid &__t, const std::string &__s, const _VDBL_context &__c, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an)
- bool add_table_view (const std::string &__s, const _VDBL_context &__c)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  bool add_table_view (const std::string &__s, const _VDBL_context &__c, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  bool add_visible (const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an)
- template<template< class _TC, class _TA > class _SqCtr, class _Al >
  bool rm_visible (const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > &__an)
- _VDBL_tableid get_tableid (const std::string &_C_i) const
- bool has_view (const _VDBL_tableid &_C_i) const

- bool has_view (const std::string &_C_i) const
- _VDBL_view ∗ get_view (const _VDBL_tableid &_C_i) const

9.48.1 Detailed Description

This class implements a view onto a complete database. The view names correspond to the names of all existing tables. Optionally, the views can be restricted to subsets of the tables. All constructed views are standard views.

Definition at line 310 of file vdbl_viewdbase.h.

9.48.2 Constructor & Destructor Documentation

9.48.2.1 viewdbase::viewdbase () `[inline]`

standard constructor

Definition at line 338 of file vdbl_viewdbase.h.

9.48.2.2 viewdbase::viewdbase (const database & *db*, const userid & *uid*, const context & *c*, view_enum *e* = `V_frozen`) `[inline]`

constructor which builds a view to the database from

- `db` – the database
- `c` – the evaluation context for all views
- `e` – the view type for all views

Definition at line 346 of file vdbl_viewdbase.h.

9.48.2.3 template<template< class _TC, class _TA > class _SqCtr, class _Al > viewdbase::viewdbase (const database & *db*, const userid & *uid*, const context & *c*, const _SqCtr< std::pair< tableid, rowid >, _Al > & *__an*, view_enum *e* = `V_frozen`) `[inline]`

constructor which builds a view to the database from

- `db` – the database
- `c` – the evaluation context for all views
- `uid` – the user id of the user who owns the view
- `e` – the view type for all views The fourth argument is any sequential container of table,row pairs to which the view shall be restricted.

Definition at line 360 of file vdbl_viewdbase.h.

9.48.2.4 virtual viewdbase::∼viewdbase () `[inline, virtual]`

standard destructor

Definition at line 368 of file vdbl_viewdbase.h.

9.48.3 Member Function Documentation

9.48.3.1 template<template< class _TC, class _TA > class _SqCtr, class _Al > bool viewdbase::add_table_view (const std::string & *__s*, const _VDBL_context & *__c*, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*) `[inline]`

This method adds another view to the viewdb. The table name is given by `__s`, and the context by `__c`. The sequential container of table,row pairs specifies additional table,row pairs which shall be visible in the view.

Reimplemented from _VDBL_viewdbase.

Definition at line 410 of file vdbl_viewdbase.h.

9.48.3.2 bool viewdbase::add_table_view (const std::string & *__s*, const _VDBL_context & *__c*) `[inline]`

This method adds another view to the viewdb. The table name is given by `__s`, and the context by `__c`.

Reimplemented from _VDBL_viewdbase.

Definition at line 398 of file vdbl_viewdbase.h.

9.48.3.3 template<template< class _TC, class _TA > class _SqCtr, class _Al > bool viewdbase::add_table_view (const _VDBL_tableid & *__t*, const std::string & *__s*, const _VDBL_context & *__c*, const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*) `[inline]`

This method adds another view to the viewdb. The table id is given by `__t`, the table name by `__s`, and the context by `__c`. The sequential container of table,row pairs specifies additional table,row pairs which shall be visible in the view.

Reimplemented from _VDBL_viewdbase.

Definition at line 387 of file vdbl_viewdbase.h.

9.48.3.4 bool viewdbase::add_table_view (const _VDBL_tableid & *__t*, const std::string & *__s*, const _VDBL_context & *__c*) `[inline]`

This method adds another view to the viewdb. The table id is given by `__t`, the table name by `__s`, and the context by `__c`.

Reimplemented from _VDBL_viewdbase.

Definition at line 374 of file vdbl_viewdbase.h.

9.48.3.5 template<template< class _TC, class _TA > class _SqCtr, class _Al > bool viewdbase::add_visible (const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*) `[inline]`

The sequential container of table,row pairs specifies additional table,row pairs which shall be visible in the view.

Reimplemented from _VDBL_viewdbase.

Definition at line 421 of file vdbl_viewdbase.h.

9.48.3.6 _VDBL_tableid _VDBL_viewdbase::get_tableid (const std::string & *_C_i*) const `[inline, inherited]`

return the table id (and view id) of table `_C_i`

Definition at line 73 of file vdbl_viewdbase.h.

9.48.3.7 _VDBL_view* _VDBL_viewdbase::get_view (const _VDBL_tableid & *_C_i*) const `[inline, inherited]`

this method returns a pointer to the view associated to id `_C_i`.

Definition at line 115 of file vdbl_viewdbase.h.

9.48.3.8 viewbase* viewdbase::get_view (const std::string & *_C_i*) const `[inline]`

this method returns a pointer to the view `_C_i`.

Reimplemented from _VDBL_viewdbase.

Definition at line 332 of file vdbl_viewdbase.h.

9.48.3.9 viewbase* viewdbase::get_view (const char * *_C_i*) const `[inline]`

this method returns a pointer to the view `_C_i`.

Definition at line 327 of file vdbl_viewdbase.h.

9.48.3.10 bool _VDBL_viewdbase::has_view (const std::string & *_C_i*) const `[inline, inherited]`

check whether the view `_C_i` exists

Definition at line 109 of file vdbl_viewdbase.h.

9.48.3.11 bool _VDBL_viewdbase::has_view (const _VDBL_tableid & *_C_i*) const `[inline, inherited]`

check whether a given view (associated to table id `_C_i`) exists

Definition at line 92 of file vdbl_viewdbase.h.

9.48.3.12 bool viewdbase::has_view (const char ∗ *_C_i*) const `[inline]`

check whether the view `_C_i` exists

Definition at line 322 of file vdbl_viewdbase.h.

9.48.3.13 template<template< class _TC, class _TA > class _SqCtr, class _Al > bool viewdbase::rm_visible (const _SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _Al > & *__an*) `[inline]`

The sequential container of table,row pairs specifies table,row pairs which shall no longer be visible in the view.

Reimplemented from _VDBL_viewdbase.

Definition at line 431 of file vdbl_viewdbase.h.

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

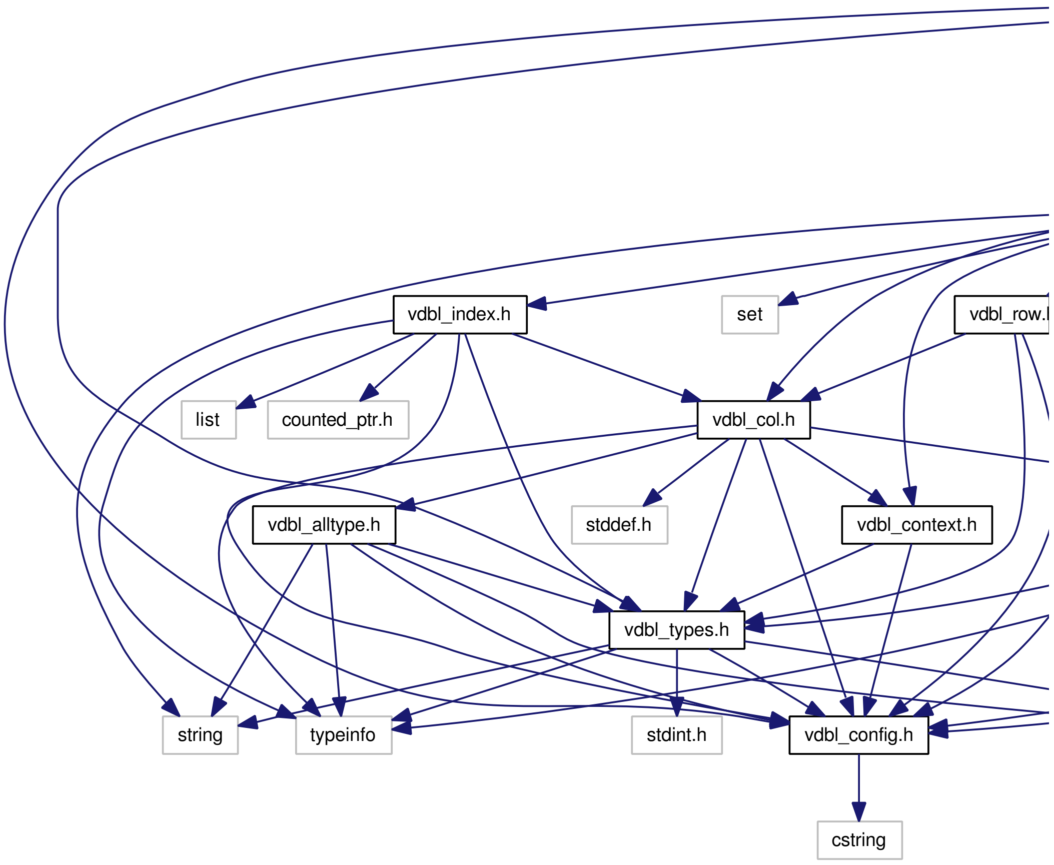
- vdbl_viewdbase.h

10 File Documentation

10.1 database File Reference

`#include <vdbl_database.h>`

Include dependency graph for database:



10.1.1 Detailed Description

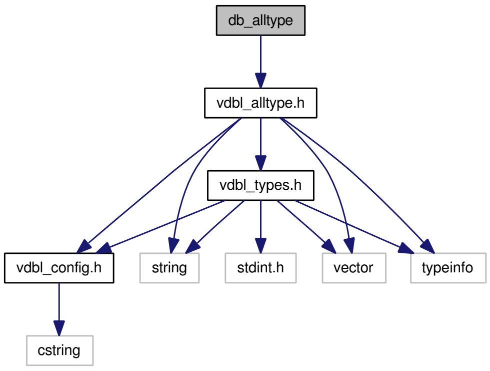
This is the external header file intended for direct use.

Definition in file database.

10.2 db_alltype File Reference

`#include <vdbl_alltype.h>`

Include dependency graph for db_alltype:



10.2.1 Detailed Description

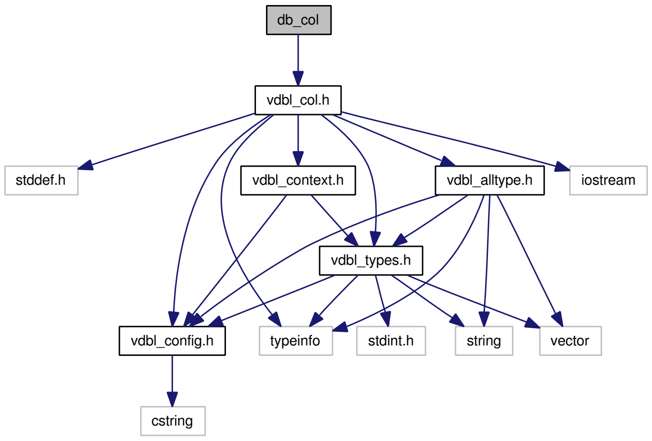
This is the external header file intended for direct use.

Definition in file db_alltype.

10.3 db_col File Reference

`#include <vdbl_col.h>`

Include dependency graph for db_col:



10.3.1 Detailed Description

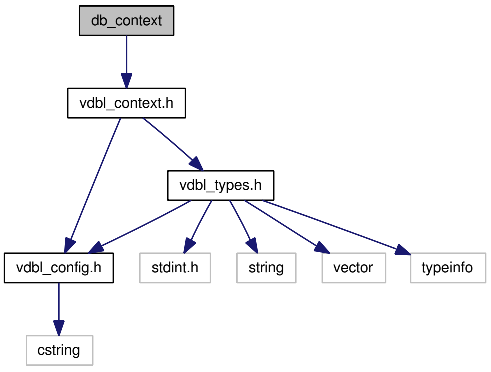
This is the external header file intended for direct use.

Definition in file db_col.

10.4 db_context File Reference

```
#include <vdbl_context.h>
```

Include dependency graph for db_context:



10.4.1 Detailed Description

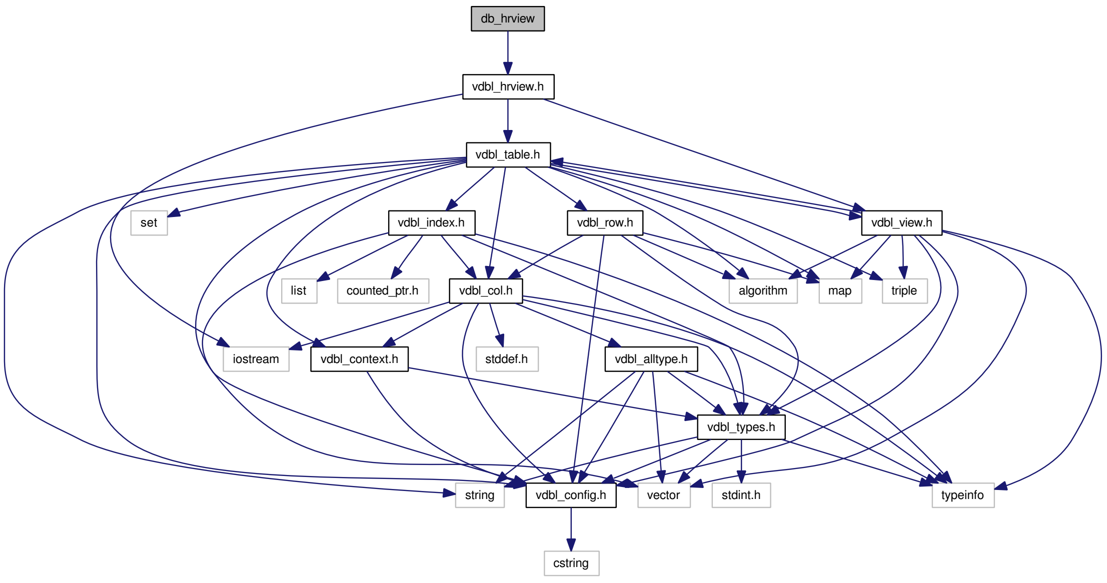
This is the external header file intended for direct use.

Definition in file db_context.

10.5 db_hrview File Reference

```
#include <vdbl_hrview.h>
```

Include dependency graph for db_hrview:



10.5.1 Detailed Description

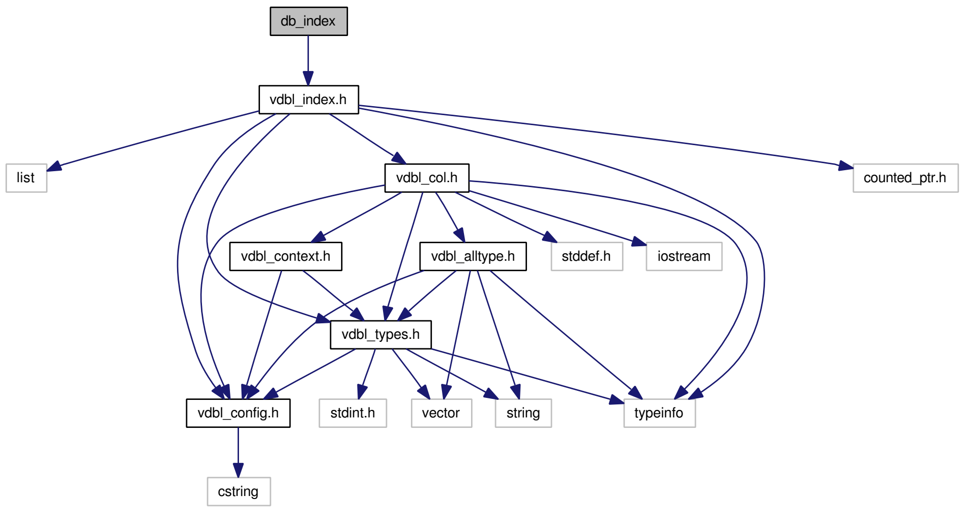
This is the external header file intended for direct use.

Definition in file db_hrview.

10.6 db_index File Reference

```
#include <vdbl_index.h>
```

Include dependency graph for db_index:



10.6.1 Detailed Description

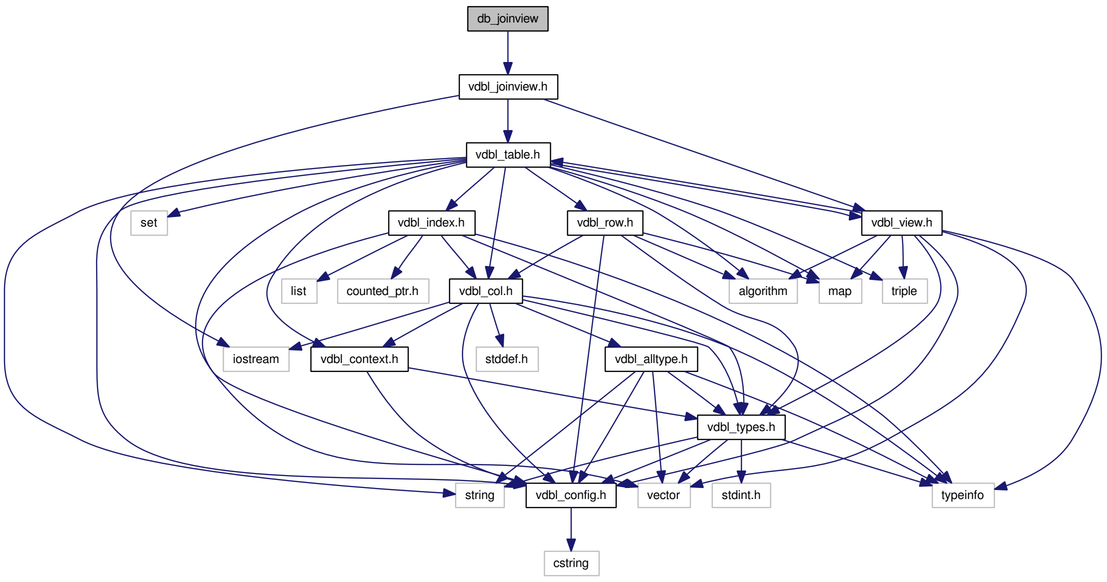
This is the external header file intended for direct use.

Definition in file db_index.

10.7 db_joinview File Reference

```
#include <vdbl_joinview.h>
```

Include dependency graph for db_joinview:



10.7.1 Detailed Description

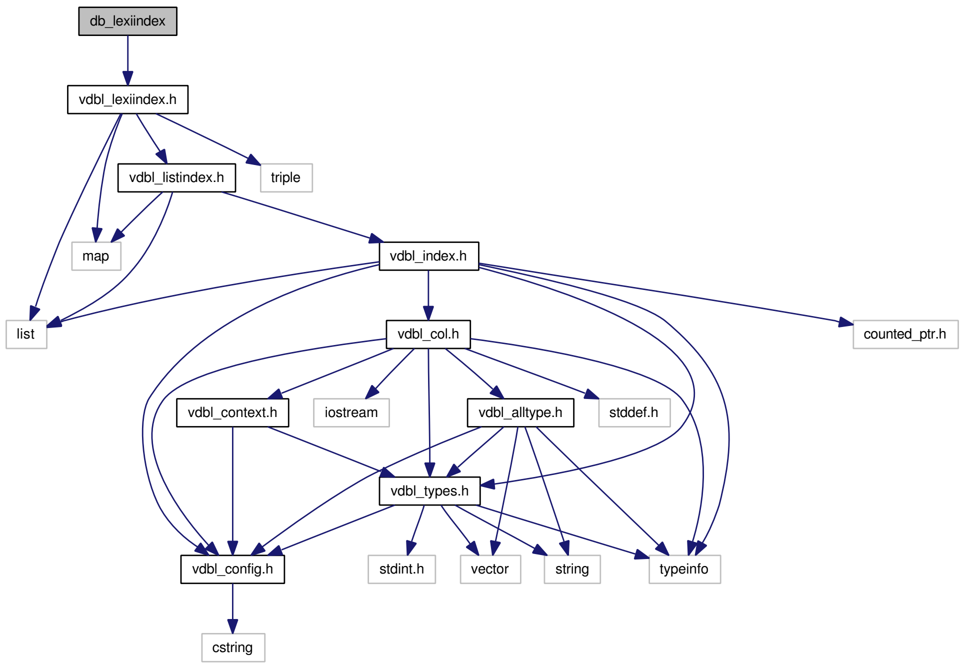
This is the external header file intended for direct use.

Definition in file db_joinview.

10.8 db_lexiindex File Reference

```
#include <vdbl_lexiindex.h>
```

Include dependency graph for db_lexiindex:



10.8.1 Detailed Description

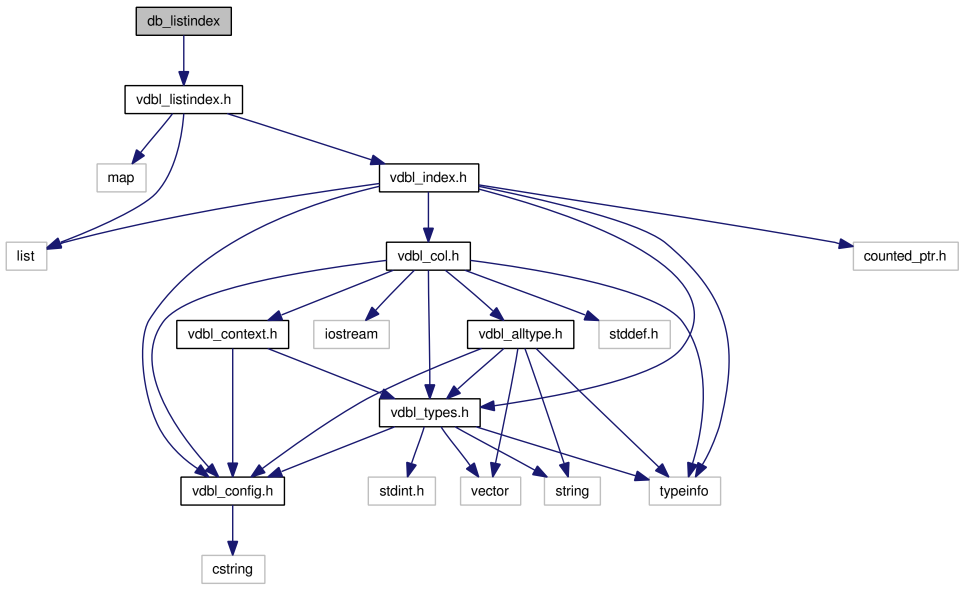
This is the external header file intended for direct use.

Definition in file db_lexiindex.

10.9 db_listindex File Reference

```
#include <vdbl_listindex.h>
```

Include dependency graph for db_listindex:



10.9.1 Detailed Description

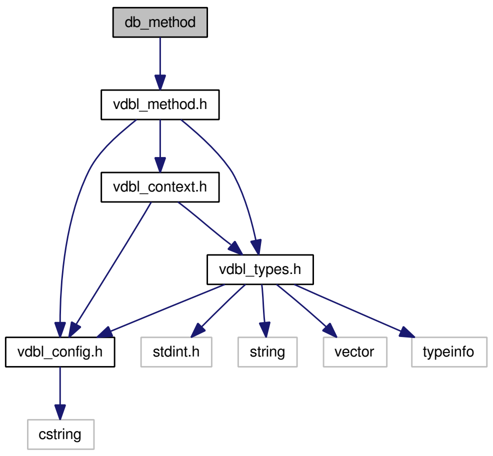
This is the external header file intended for direct use.

Definition in file db_listindex.

10.10 db_method File Reference

`#include <vdbl_method.h>`

Include dependency graph for db_method:



10.10.1 Detailed Description

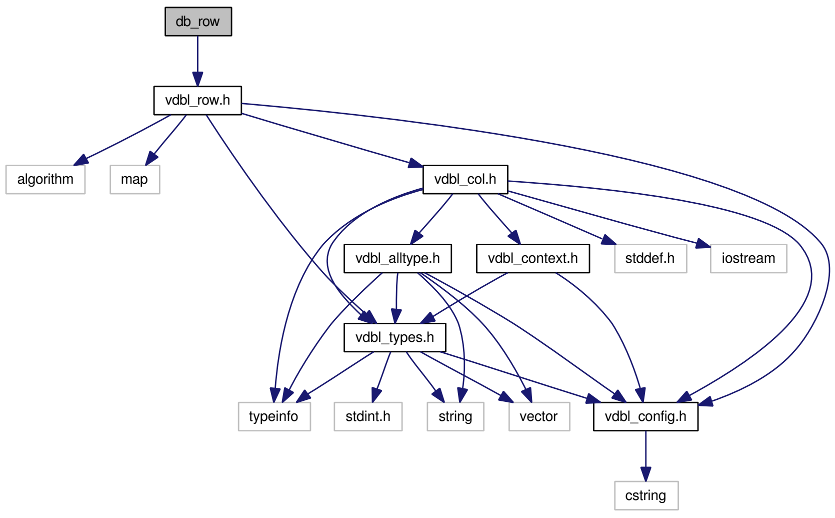
This is the external header file intended for direct use.

Definition in file db_method.

10.11 db_row File Reference

```
#include <vdbl_row.h>
```

Include dependency graph for db_row:



10.11.1 Detailed Description

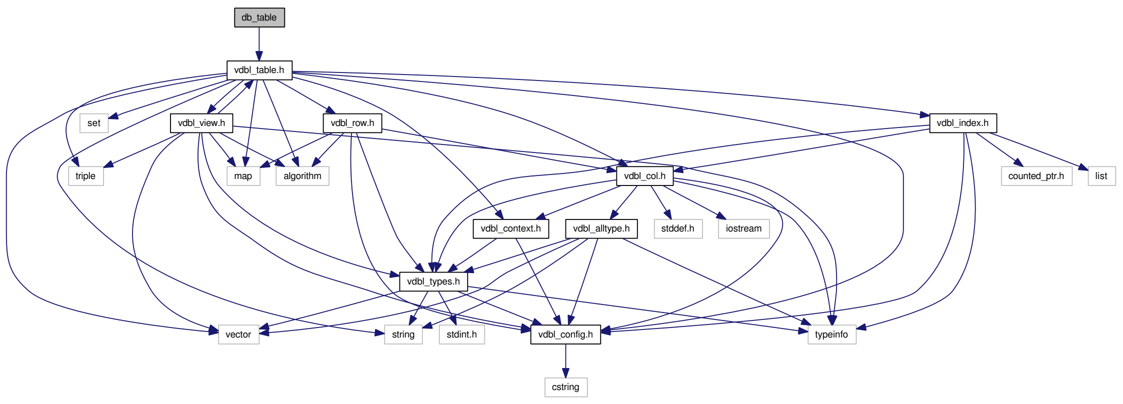
This is the external header file intended for direct use.

Definition in file db_row.

10.12 db_table File Reference

```
#include <vdbl_table.h>
```

Include dependency graph for db_table:



10.12.1 Detailed Description

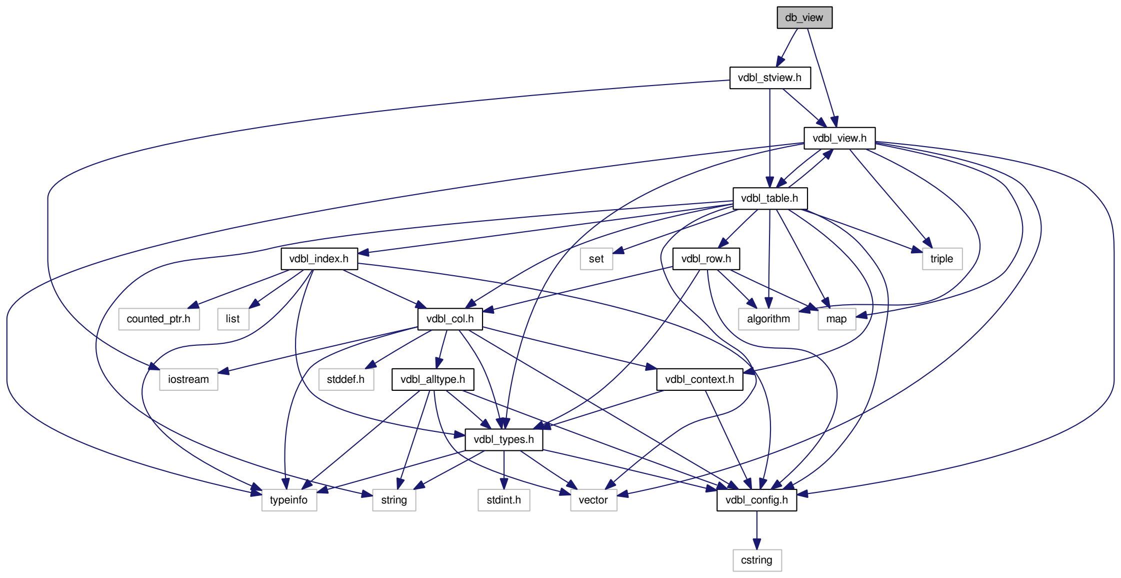
This is the external header file intended for direct use.

Definition in file db_table.

10.13 db_view File Reference

```
#include <vdbl_view.h>
#include <vdbl_stview.h>
```

Include dependency graph for db_view:



10.13.1 Detailed Description

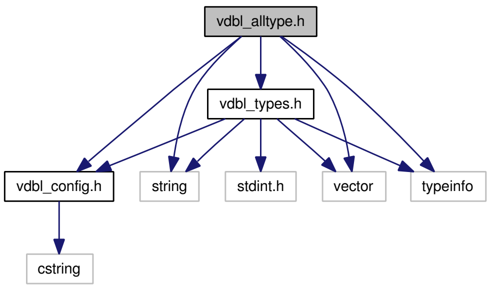
This is the external header file intended for direct use.

Definition in file db_view.

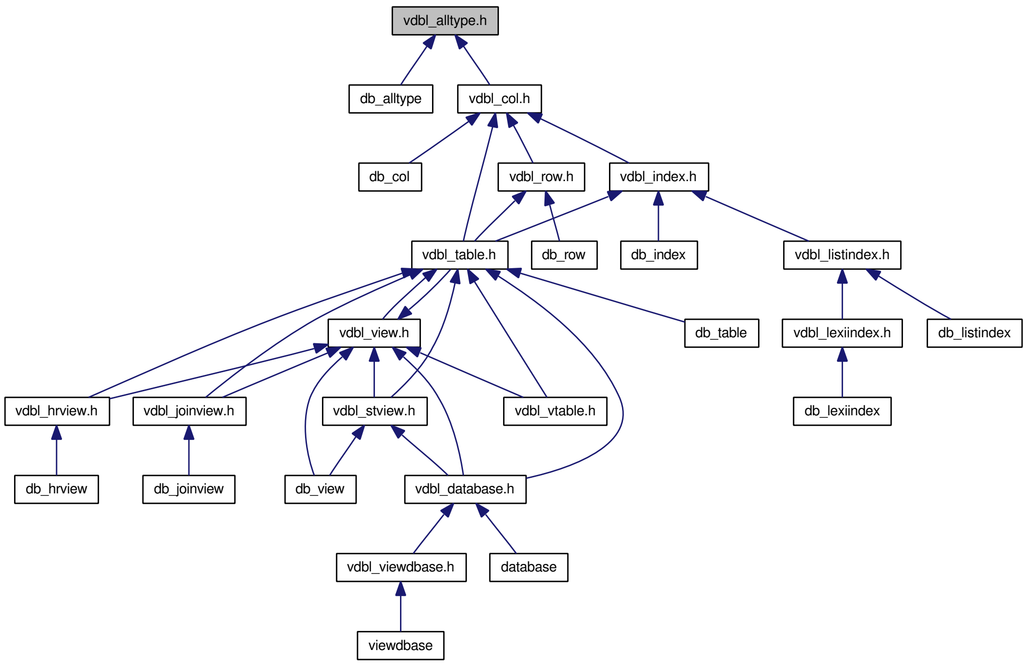
10.14 vdbl_alltype.h File Reference

```
#include <vdbl_config.h>
#include <vdbl_types.h>
#include <string>
#include <vector>
#include <typeinfo>
```

Include dependency graph for vdbl_alltype.h:



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



Classes

- class _VDBL_alltype_base

The base class for the all_type class.

- class _VDBL_alltype< _TR >

    The templated class for the all_type class.

- class _VDBL_date

    The VDBL date class.

- class _VDBL_dateinterval

    The VDBL date interval class.

- class _VDBL_mixtype

    mixed type

- class alltype< _TT >

    The templated alltype class.

Defines

- #define VDBL_MIXTYPE_EMPTY -1
- #define VDBL_MIXTYPE_ALLOCED_NONE 0
- #define VDBL_MIXTYPE_ALLOCED_B 1
- #define VDBL_MIXTYPE_ALLOCED_N 2
- #define VDBL_MIXTYPE_ALLOCED_D 3
- #define VDBL_MIXTYPE_ALLOCED_U 4
- #define VDBL_MIXTYPE_ALLOCED_S 5

Typedefs

- typedef _VDBL_date date

    the date type

- typedef _VDBL_dateinterval dateinterval

    the dateinterval type

- typedef _VDBL_mixtype mixtype

    a mixed type of various scalars and vectors

- typedef _VDBL_alltype_base alltype_base

    the base class of the alltype

10.14.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_alltype.h.

10.14.2 Define Documentation

10.14.2.1 #define VDBL_MIXTYPE_ALLOCED_B 1

These defines internally describe which kind of vector has been allocated for a mixtype, or whether it is scalar or empty.

Definition at line 337 of file vdbl_alltype.h.

10.14.2.2 #define VDBL_MIXTYPE_ALLOCED_D 3

These defines internally describe which kind of vector has been allocated for a mixtype, or whether it is scalar or empty.

Definition at line 339 of file vdbl_alltype.h.

10.14.2.3 #define VDBL_MIXTYPE_ALLOCED_N 2

These defines internally describe which kind of vector has been allocated for a mixtype, or whether it is scalar or empty.

Definition at line 338 of file vdbl_alltype.h.

10.14.2.4 #define VDBL_MIXTYPE_ALLOCED_NONE 0

These defines internally describe which kind of vector has been allocated for a mixtype, or whether it is scalar or empty.

Definition at line 336 of file vdbl_alltype.h.

10.14.2.5 #define VDBL_MIXTYPE_ALLOCED_S 5

These defines internally describe which kind of vector has been allocated for a mixtype, or whether it is scalar or empty.

Definition at line 341 of file vdbl_alltype.h.

10.14.2.6 #define VDBL_MIXTYPE_ALLOCED_U 4

These defines internally describe which kind of vector has been allocated for a mixtype, or whether it is scalar or empty.

Definition at line 340 of file vdbl_alltype.h.

10.14.2.7 #define VDBL_MIXTYPE_EMPTY -1

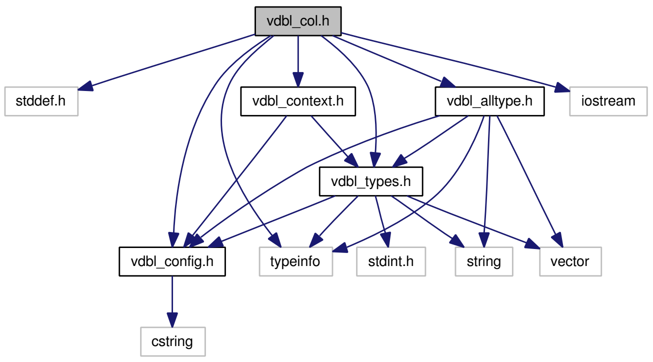
These defines internally describe which kind of vector has been allocated for a mixtype, or whether it is scalar or empty.

Definition at line 335 of file vdbl_alltype.h.

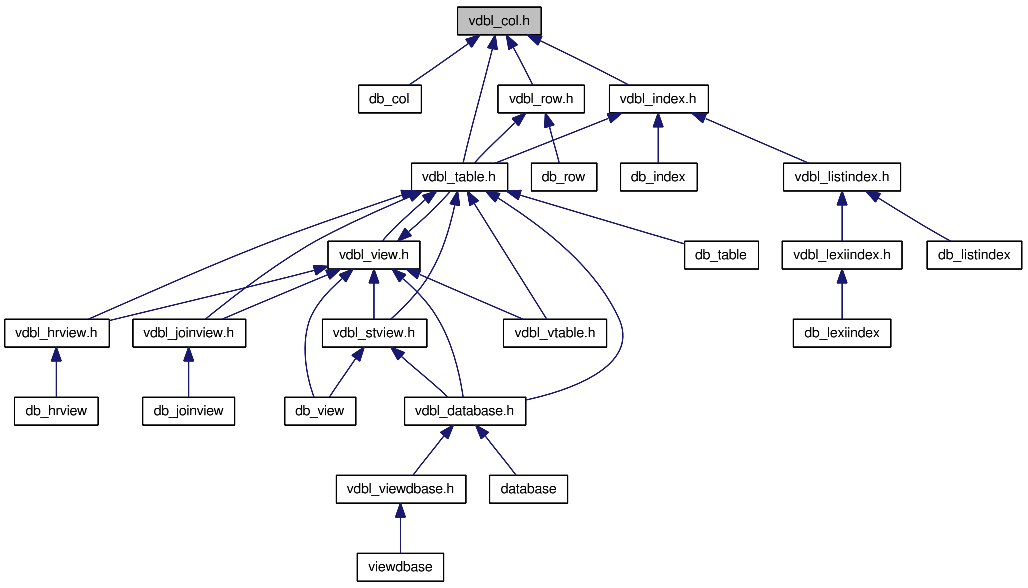
10.15 vdbl_col.h File Reference

```
#include <stddef.h>
#include <typeinfo>
#include <iostream>
#include <vdbl_config.h>
#include <vdbl_types.h>
#include <vdbl_context.h>
#include <vdbl_alltype.h>
```

Include dependency graph for vdbl_col.h:



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



Classes

- class __VDBL_colbase

    The base class of the internal column structure.

- class _VDBL_colbase< _TR >

    The type dependent base class of the internal column structure.

- class _VDBL_col

    The generic column class (the external structure).

- class _VDBL_stdcol< _TT >

    generic column class for constant values

- class _VDBL_mthdcol< _TC, _TM, _TR >

    generic column class for methods

- class col_base< _TR >

    column base class

- class typed_col< _TT >

    external name for constant data columns

- class method_col< _TM >

    external name for computed columns

Typedefs

- typedef _VDBL_col col

  the column class

- typedef _VDBL_stdcol< mixtype > standard_col

  the standard column class with constant `mixtype` *data*

Functions

- template<class _TC >
  std::ostream & print_it (std::ostream &o, const _TC &t)
- std::ostream & operator<< (std::ostream &o, const _VDBL_col &c)

10.15.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_col.h.

10.15.2 Function Documentation

10.15.2.1 std::ostream& operator<< (std::ostream & *o*, const _VDBL_col & *c*) `[inline]`

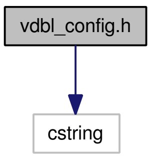
The print operation for generic columns. This implicitely calls operator<< for the columns type. So it is necessary that this operator is indeed defined.

Definition at line 360 of file vdbl_col.h.

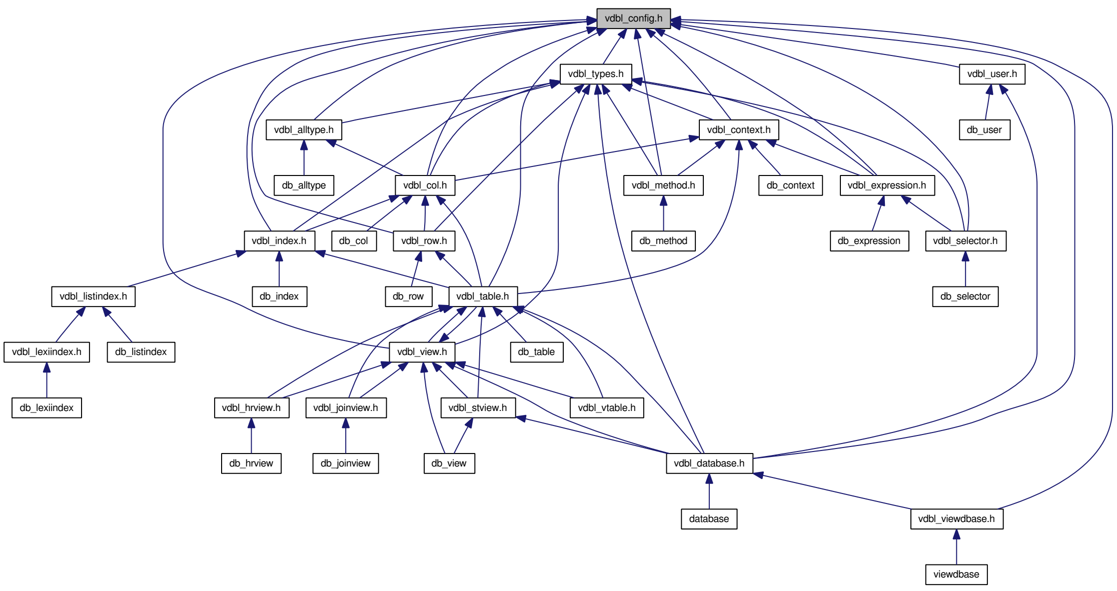
10.16 vdbl_config.h File Reference

`#include <cstring>`

Include dependency graph for vdbl_config.h:



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



10.16.1 Detailed Description

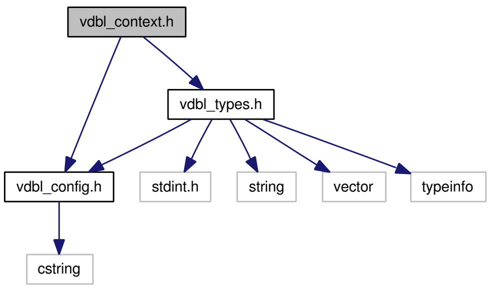
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_config.h.

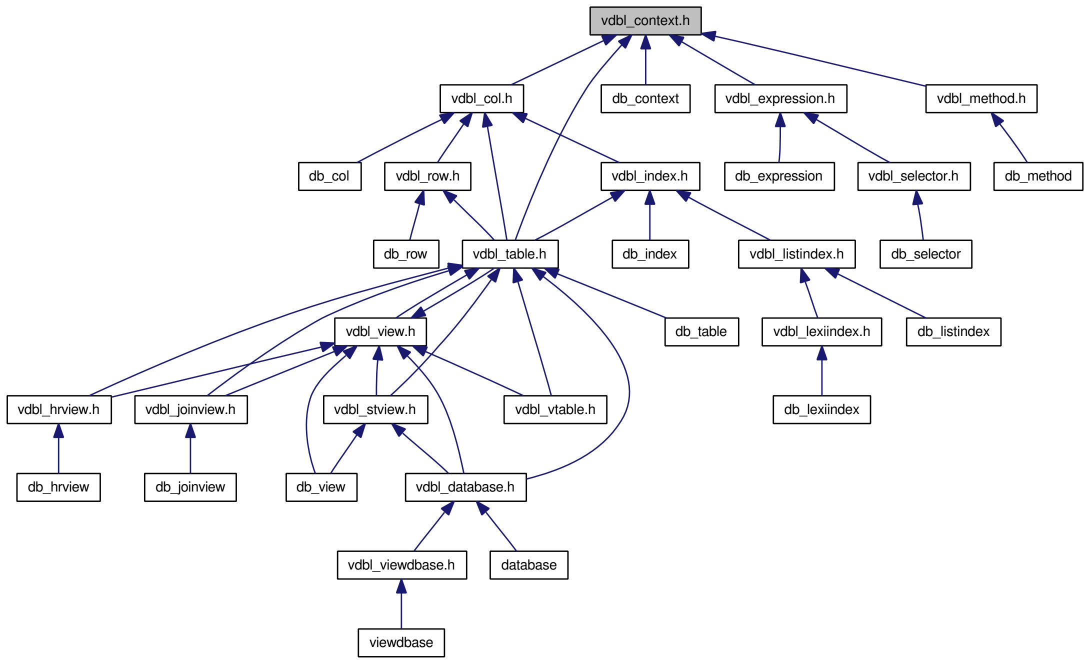
10.17 vdbl_context.h File Reference

```
#include <vdbl_config.h>
#include <vdbl_types.h>
```

Include dependency graph for vdbl_context.h:



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



Classes

- class _VDBL_context

  base class for context objects

Typedefs

- typedef _VDBL_context context

  evaluation context base class

10.17.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

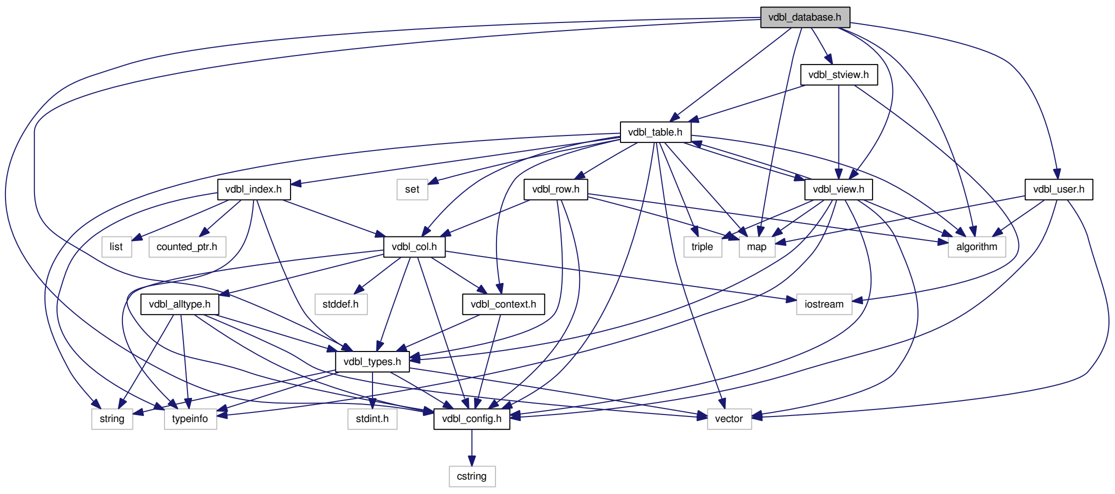
Definition in file vdbl_context.h.

10.18 vdbl_database.h File Reference

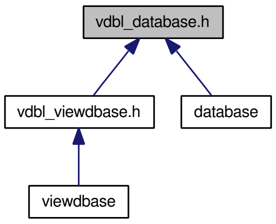
```
#include <algorithm>
#include <map>
#include <vdbl_config.h>
#include <vdbl_types.h>
#include <vdbl_table.h>
```

```
#include <vdbl_user.h>
#include <vdbl_view.h>
#include <vdbl_stview.h>
```

Include dependency graph for vdbl_database.h:



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



Classes

- class _VDBL_userflags

  The permission flags for a user.

- class _VDBL_aclentry

  entry in the access control list

- class _VDBL_acl

  Access control list.

- class _VDBL_tableflags

  flags for one table

- class _VDBL_viewflags

  flags for one view

- class _VDBL_database

    the database class

- class database

    the database class

Typedefs

- typedef _VDBL_userflags userflags

    user flags and permissions

- typedef _VDBL_viewflags viewflags

    view flags and ACLs

- typedef _VDBL_tableflags tableflags

    table flags and ACLs

- typedef _VDBL_aclentry aclentry

    entry in the access control list (ACL)

- typedef _VDBL_acl acl

    ACL for one user.

10.18.1 Detailed Description

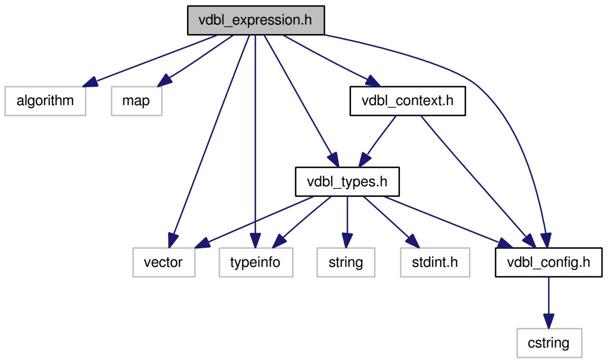
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_database.h.

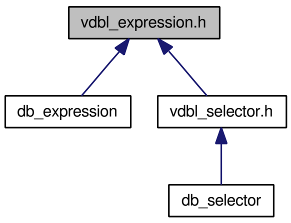
10.19 vdbl_expression.h File Reference

```
#include <algorithm>
#include <map>
#include <vector>
#include <typeinfo>
#include <vdbl_config.h>
#include <vdbl_types.h>
#include <vdbl_context.h>
```

Include dependency graph for vdbl_expression.h:



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



Classes

- class **_VDBL_eval_expr**
- class **_VDBL_exprrow**
- class **_VDBL_exprcol**
- class **_VDBL_exprequal**
- class **_VDBL_exprneql**

10.19.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_expression.h.

10.20 vdbl_extradocu.h File Reference

Namespaces

- namespace vdbl

    Main namespace of the VDBL.

10.20.1 Detailed Description

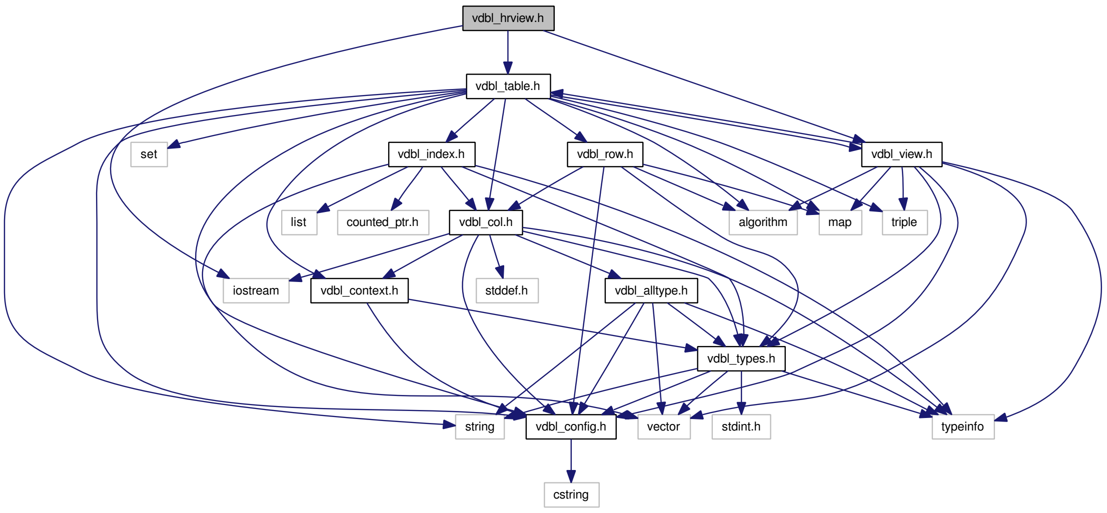
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_extradocu.h.

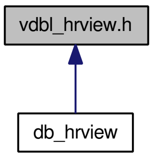
10.21 vdbl_hrview.h File Reference

```
#include <iostream>
#include <vdbl_table.h>
#include <vdbl_view.h>
```

Include dependency graph for vdbl_hrview.h:



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



Classes

- class _VDBL_hierarchicalview

    hierarchical view class

- class hierarchical_view

    hierarchical view class onto a stack of tables

10.21.1 Detailed Description

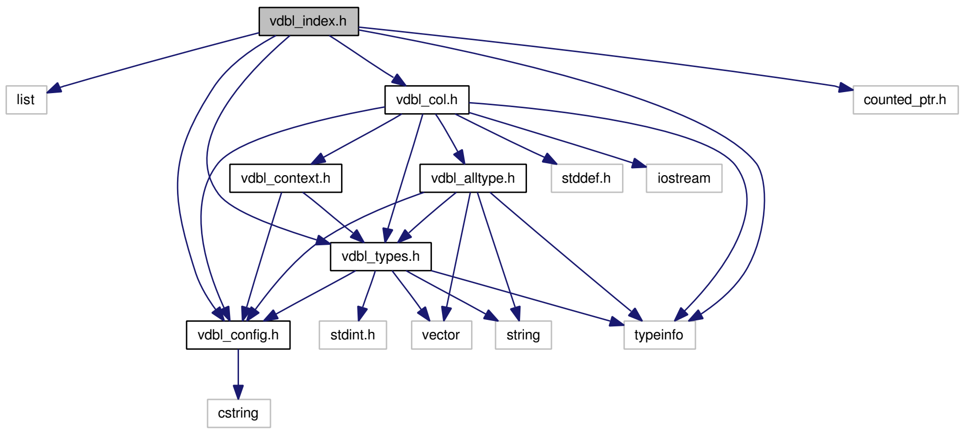
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_hrview.h.

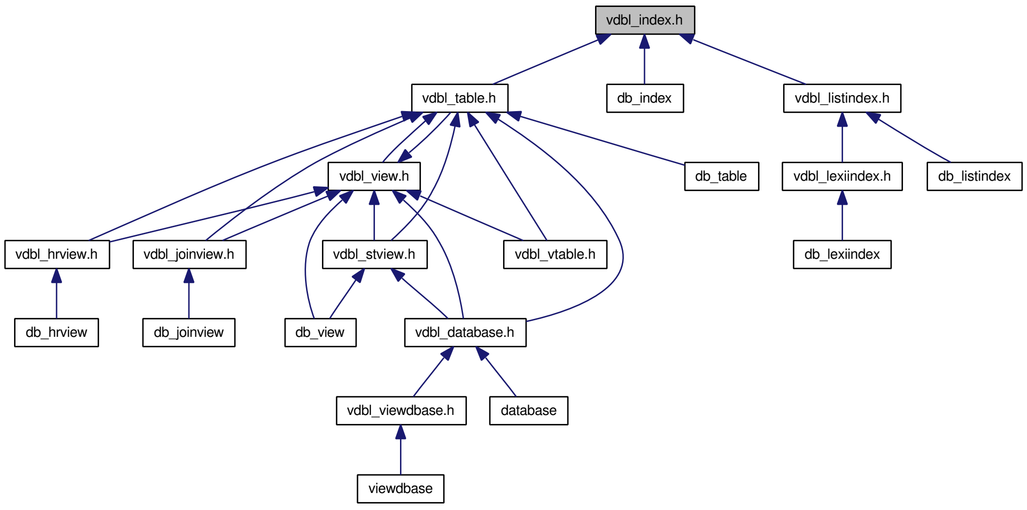
10.22 vdbl_index.h File Reference

```
#include <list>
#include <typeinfo>
#include <counted_ptr.h>
#include <vdbl_config.h>
#include <vdbl_types.h>
#include <vdbl_col.h>
```

Include dependency graph for vdbl_index.h:



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



Classes

- class **__VDBL_index**
- class **index**

10.22.1 Detailed Description

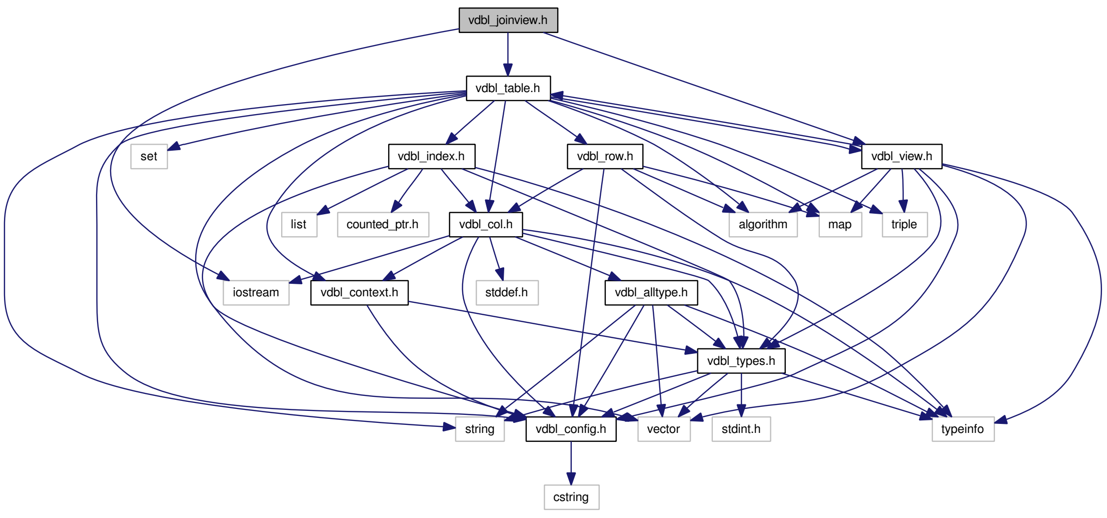
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_index.h.

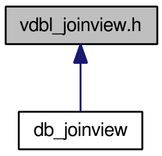
10.23 vdbl_joinview.h File Reference

```
#include <iostream>
#include <vdbl_table.h>
#include <vdbl_view.h>
```

Include dependency graph for vdbl_joinview.h:



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



Classes

- class **_VDBL_joinview**
- class **join_view**

10.23.1 Detailed Description

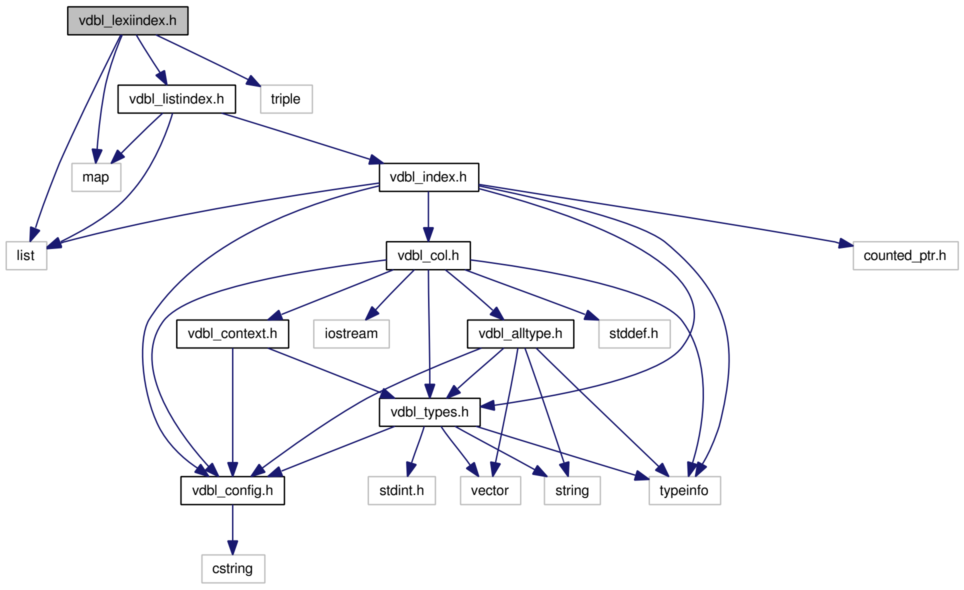
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_joinview.h.

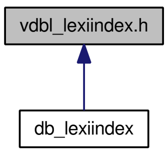
10.24 vdbl_lexiindex.h File Reference

```
#include <map>
#include <list>
#include <triple>
#include <vdbl_listindex.h>
```

Include dependency graph for vdbl_lexiindex.h:



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



Classes

- class **_VDBL_index_lex2**< **_TR, _TS** >
- class **lexi_pair_index**< **_TR, _TS** >
- class **lexi_pair_index_def**< **_TR, _TS** >
- class **_VDBL_index_lex3**< **_TR, _TS, _TT** >
- class **lexi_triple_index**< **_TR, _TS, _TT** >

- class **lexi_triple_index_def**< **_TR, _TS, _TT** >

10.24.1 Detailed Description

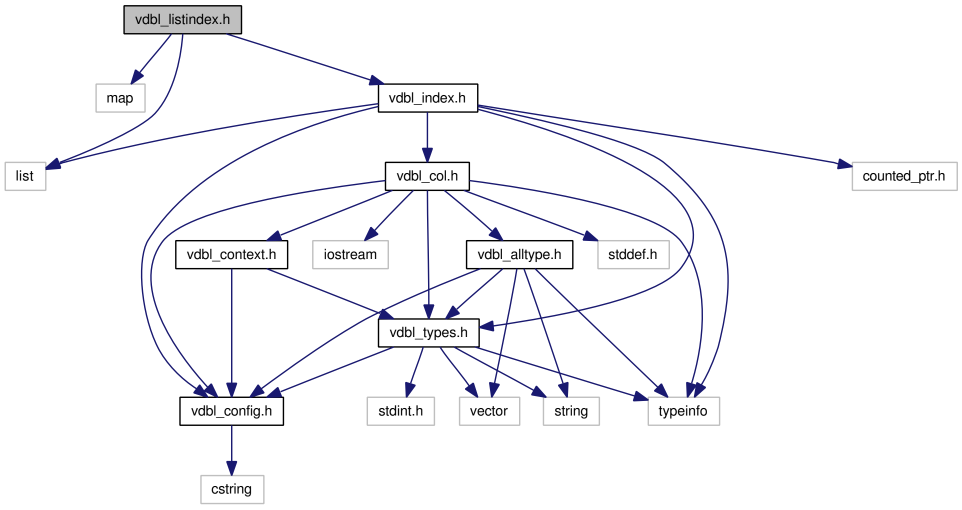
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_lexiindex.h.

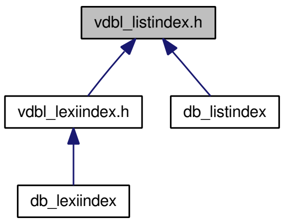
10.25 vdbl_listindex.h File Reference

```
#include <map>
#include <list>
#include <vdbl_index.h>
```

Include dependency graph for vdbl_listindex.h:



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



Classes

- class **_VDBL_index_l**< **_TR** >
- class **list_index**< **_TR** >
- class **list_index_def**< **_TR** >

10.25.1 Detailed Description

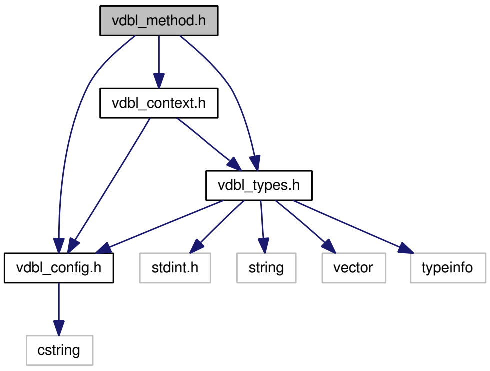
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_listindex.h.

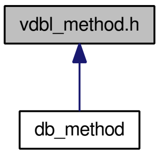
10.26 vdbl_method.h File Reference

```
#include <vdbl_config.h>
#include <vdbl_types.h>
#include <vdbl_context.h>
```

Include dependency graph for vdbl_method.h:



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



Classes

- class _VDBL_method< _TR >

    base class for methods usable in `_VDBL_mthdcol` *columns.*

- class method< _TR >

    base class for methods usable in `method` *columns.*

10.26.1 Detailed Description

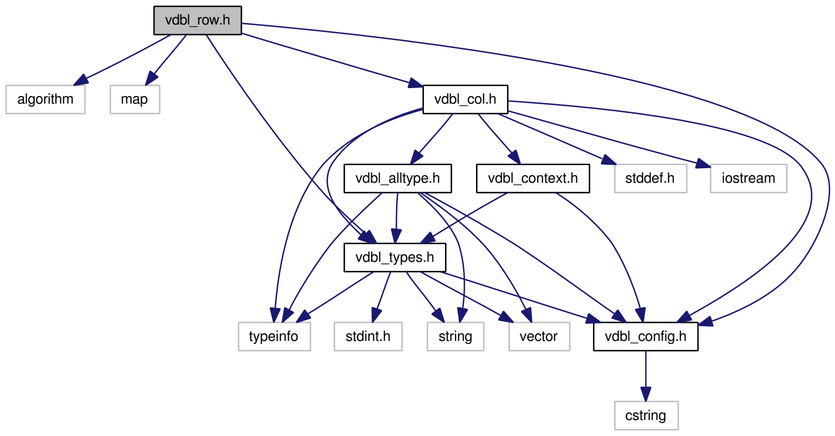
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_method.h.

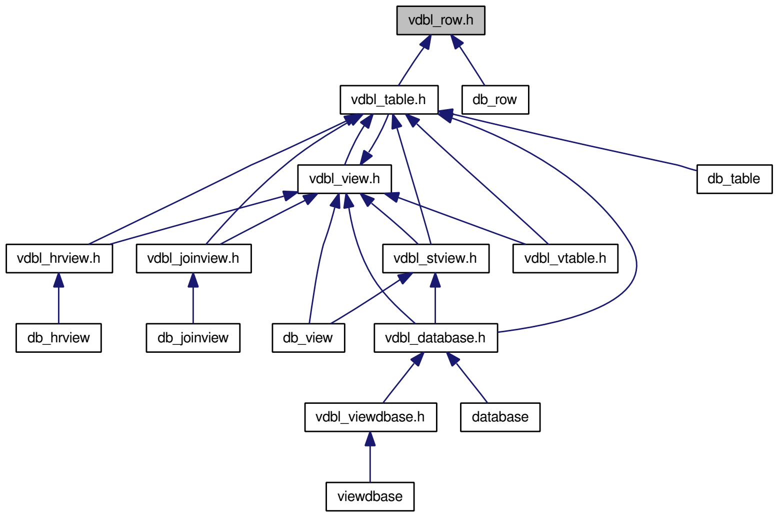
10.27 vdbl_row.h File Reference

```
#include <algorithm>
#include <map>
#include <vdbl_config.h>
#include <vdbl_types.h>
#include <vdbl_col.h>
```

Include dependency graph for vdbl_row.h:



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



Classes

- class _VDBL_row

  row class

- class row

    class implementing table rows

10.27.1 Detailed Description

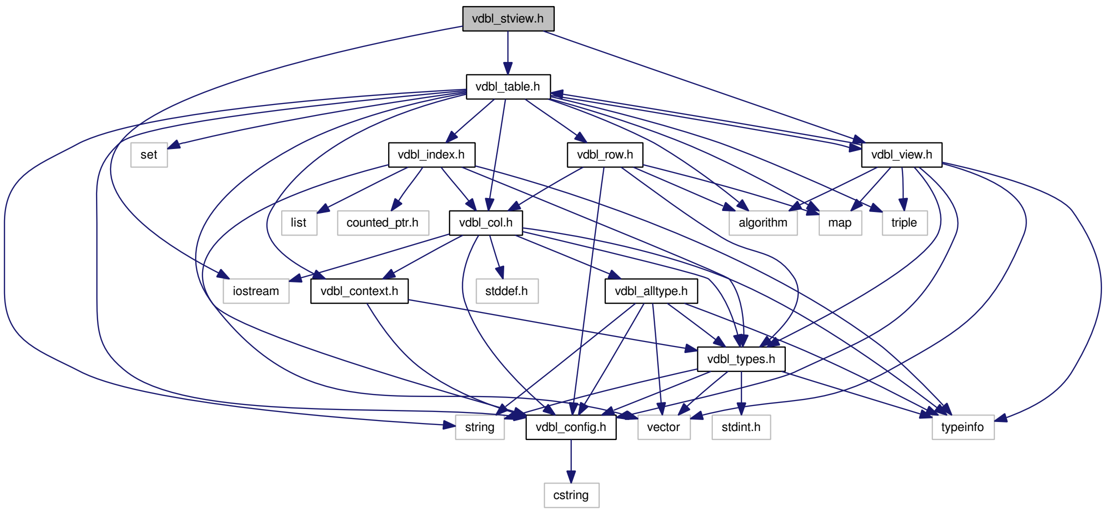
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_row.h.

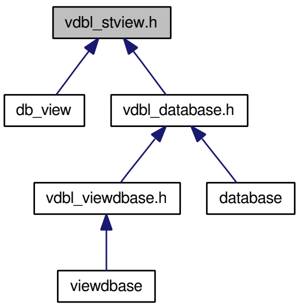
10.28 vdbl_stview.h File Reference

```
#include <iostream>
#include <vdbl_table.h>
#include <vdbl_view.h>
```

Include dependency graph for vdbl_stview.h:



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



Classes

- class _VDBL_standardview

    standard view onto **one** *table*

- class view

    standard view class onto a single table

10.28.1 Detailed Description

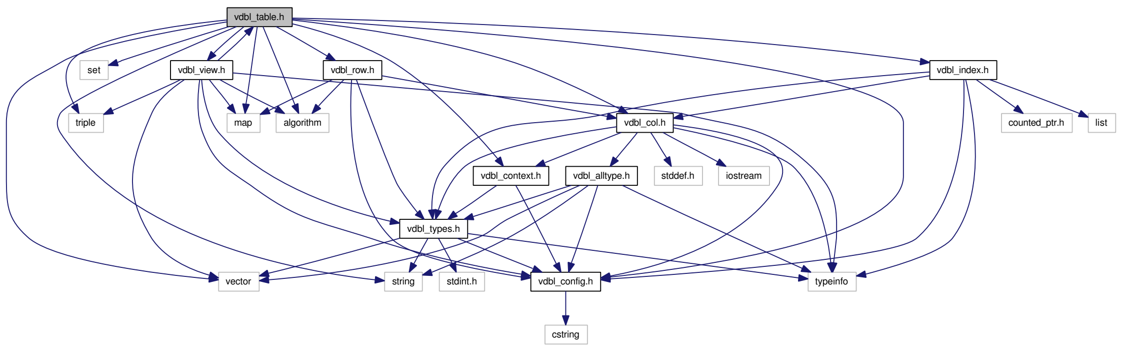
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_stview.h.

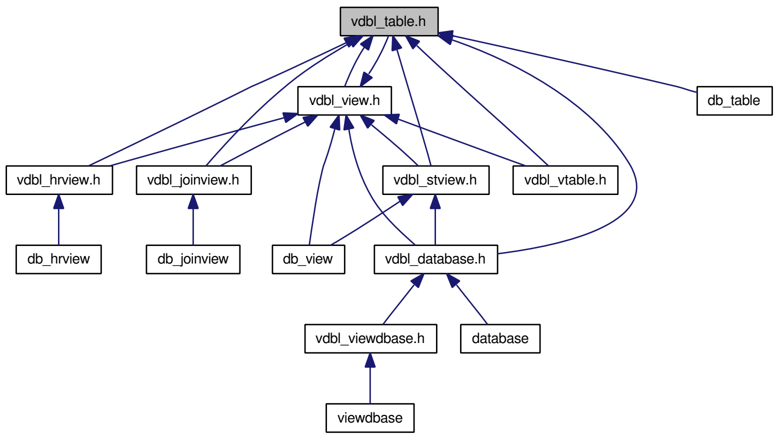
10.29 vdbl_table.h File Reference

```
#include <algorithm>
#include <map>
#include <set>
#include <vector>
#include <string>
#include <triple>
#include <vdbl_config.h>
#include <vdbl_row.h>
#include <vdbl_col.h>
#include <vdbl_index.h>
#include <vdbl_context.h>
#include <vdbl_view.h>
```

Include dependency graph for vdbl_table.h:



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



Classes

- class _VDBL_table

  the base class describing database tables

- class _VDBL_table::_VDBL_table::_col_iterator_base
- struct _VDBL_table::_VDBL_table::_col_iterator< _Tp, _Ref, _Ptr >
- struct _VDBL_table::_VDBL_table::_row_iterator< _Tp, _Ref, _Ptr >
- class _VDBL_standardtable

  standard table in databases, constructed from rows and columns

- class table

  base class for tables in a database

- class col_spec

  column specification

- class standard_table

  standard table of a database

10.29.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_table.h.

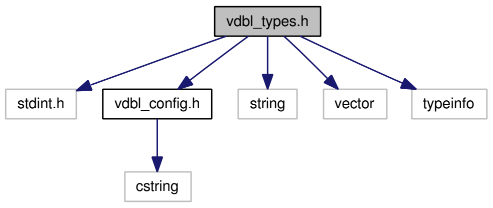
10.30 vdbl_types.h File Reference

```
#include <stdint.h>
#include <vdbl_config.h>
#include <string>
```

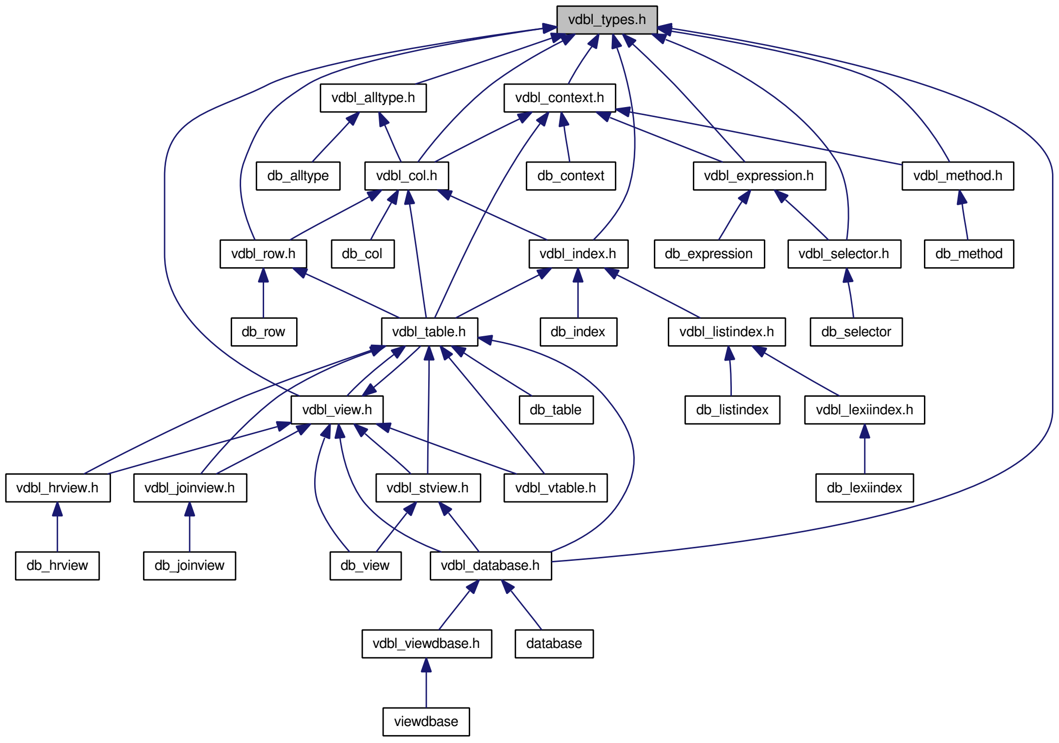
`#include <vector>`

`#include <typeinfo>`

Include dependency graph for vdbl_types.h:



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



Classes

- class _VDBL_colflags

  additional table information for a column

Defines

- #define VDBL_MAXUSERID ((__VDBL::_VDBL_userid)-1)
- #define VDBL_MAXVIEWID ((__VDBL::_VDBL_viewid)-1)
- #define VDBL_MAXCOLID ((__VDBL::_VDBL_colid)-1)

- #define VDBL_MAXROWID ((__VDBL::_VDBL_rowid)-1)
- #define VDBL_MAXTABLEID ((__VDBL::_VDBL_tableid)-1)

Typedefs

- typedef uint32_t _VDBL_userid
- typedef uint32_t _VDBL_viewid
- typedef uint64_t _VDBL_colid
- typedef uint64_t _VDBL_rowid
- typedef uint32_t _VDBL_tableid
- typedef _VDBL_userid userid

  user id

- typedef _VDBL_viewid viewid

  view id

- typedef _VDBL_colid colid

  column id

- typedef _VDBL_rowid rowid

  row id

- typedef _VDBL_tableid tableid

  table id

- typedef _VDBL_colflags colflags

  additional column properties

- typedef _V_enum view_enum

  the view type

Enumerations

- enum _V_enum

  different view properties

10.30.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

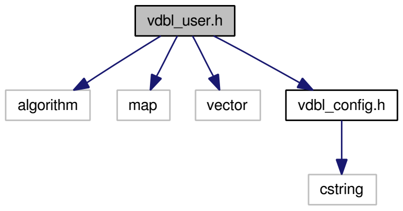
Definition in file vdbl_types.h.

10.31 vdbl_user.h File Reference

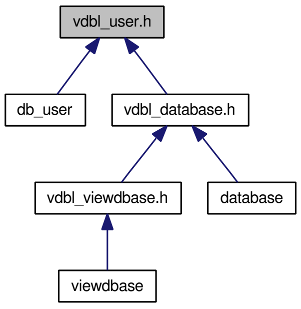
```
#include <algorithm>
#include <map>
```

```
#include <vector>
#include <vdbl_config.h>
```

Include dependency graph for vdbl_user.h:



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



Classes

- class **_VDBL_user**

10.31.1 Detailed Description

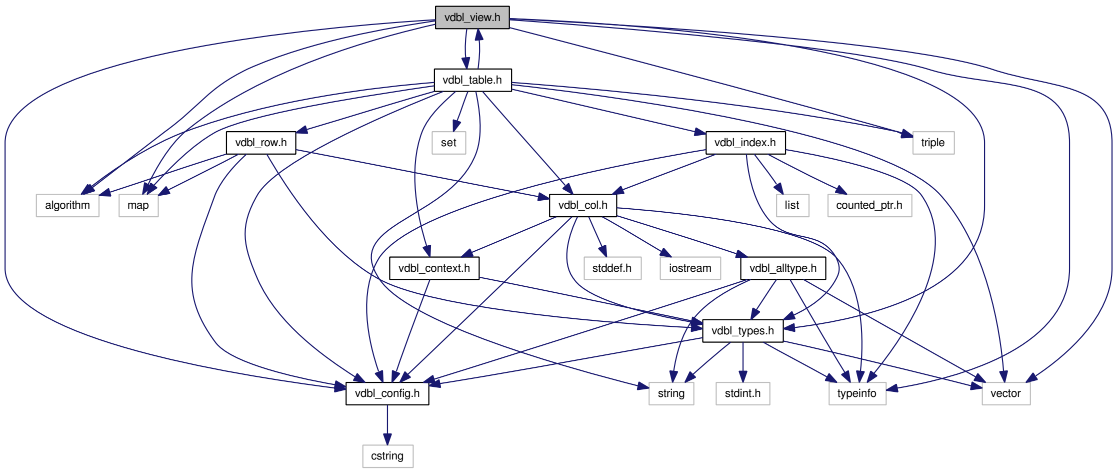
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_user.h.

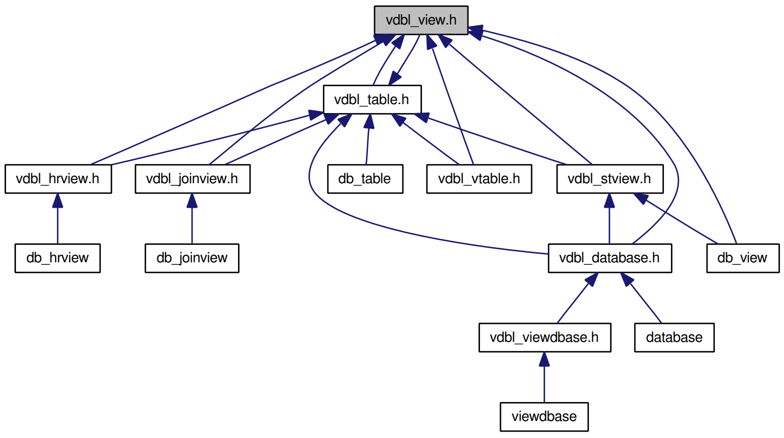
10.32 vdbl_view.h File Reference

```
#include <algorithm>
#include <map>
#include <vector>
#include <typeinfo>
#include <triple>
#include <vdbl_table.h>
#include <vdbl_config.h>
#include <vdbl_types.h>
```

Include dependency graph for vdbl_view.h:



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



Classes

- class _VDBL_view

  base class of all views.

- struct _VDBL_view::_VDBL_view::_default_iterator< _Tp, _Ref, _Ptr >
- class _VDBL_view::_VDBL_view::_col_iterator_base
- struct _VDBL_view::_VDBL_view::_col_iterator< _Tp, _Ref, _Ptr >
- class _VDBL_view::_VDBL_view::_row_iterator_base
- struct _VDBL_view::_VDBL_view::_row_iterator< _Tp, _Ref, _Ptr >

10.32.1 Detailed Description

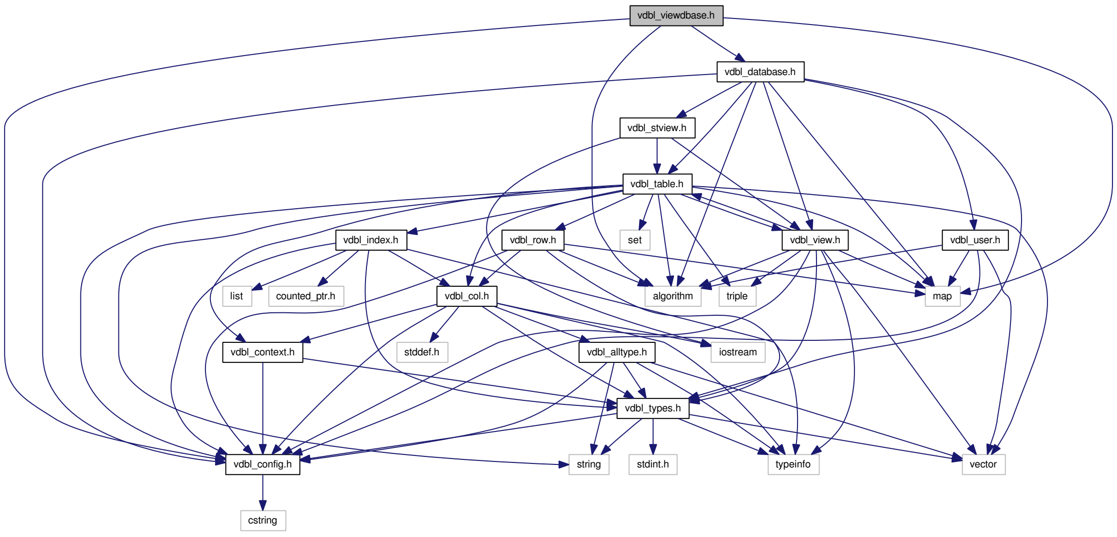
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_view.h.

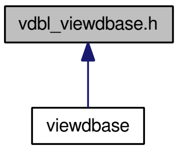
10.33 vdbl_viewdbase.h File Reference

```
#include <algorithm>
#include <map>
#include <vdbl_config.h>
#include <vdbl_database.h>
```

Include dependency graph for vdbl_viewdbase.h:



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



Classes

- class _VDBL_viewdbase

  a view to a complete database

- class viewdbase

  a view to a complete database

10.33.1 Detailed Description

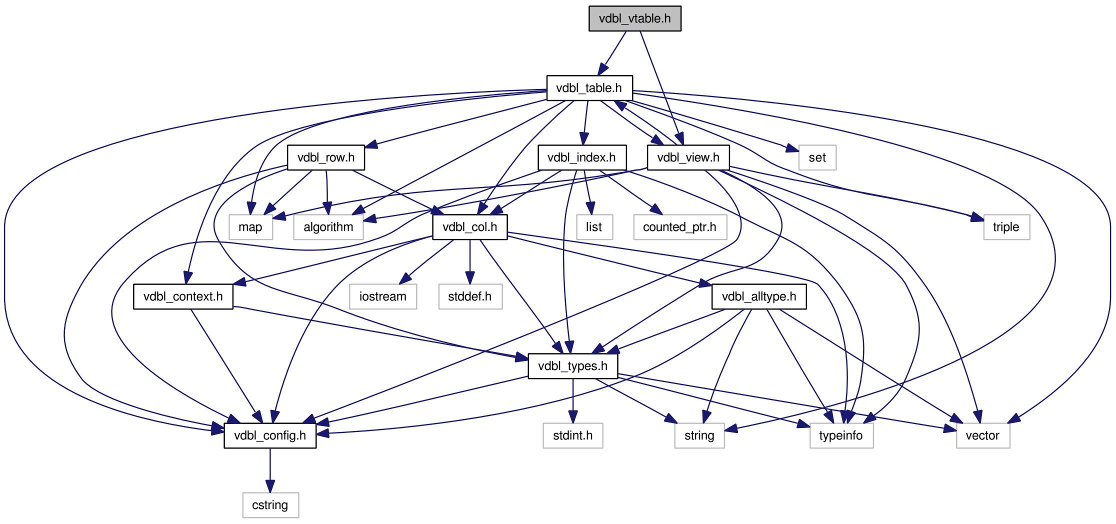
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_viewdbase.h.

10.34 vdbl_vtable.h File Reference

```
#include <vdbl_table.h>
#include <vdbl_view.h>
```

Include dependency graph for vdbl_vtable.h:



Classes

- class _VDBL_viewtable
- class **view_table**

10.34.1 Detailed Description

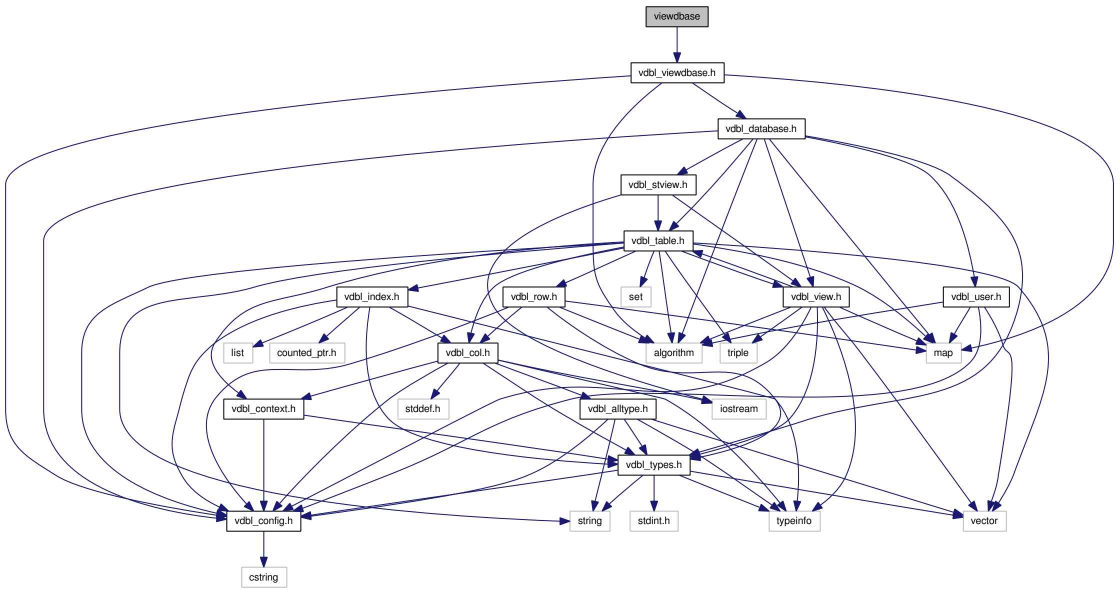
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file vdbl_vtable.h.

10.35 viewdbase File Reference

```
#include <vdbl_viewdbase.h>
```

Include dependency graph for viewdbase:



10.35.1 Detailed Description

This is the external header file intended for direct use.

Definition in file viewdbase.