

Vienna Database Library

Version 1.0

Reference Manual

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

Generated by Doxygen 1.2.14

Contents

1	Introduction	1
2	Vienna Database Library Module Index	3
3	Vienna Database Library Hierarchical Index	3
4	Vienna Database Library Compound Index	5
5	Vienna Database Library File Index	7
6	Vienna Database Library Module Documentation	9
7	Vienna Database Library Class Documentation	16
8	Vienna Database Library File Documentation	136

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. Its 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_col<T>`).
- `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<T>`). Instead of holding data, its contents are function objects (methods), which are subclasses of `class vdbl::method<T>`. 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 “overlying” 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 Vienna Database Library Module Index

2.1 Vienna Database Library Modules

Here is a list of all modules:

Classes and types for external use	9
Classes and types for internal use	13

3 Vienna Database Library Hierarchical Index

3.1 Vienna Database Library Class Hierarchy

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

<code>--VDBL_colbase</code>	16
<code>_VDBL_colbase< _R ></code>	27
<code>_VDBL_mthdcol< _C, _M, _R ></code>	54
<code>col_base< _R ></code>	95
<code>_VDBL_colbase< _M::return_type ></code>	27
<code>_VDBL_mthdcol< _M::context, _M, _M::return_type ></code>	54
<code>method_col< _M ></code>	112
<code>_VDBL_colbase< _T ></code>	27
<code>_VDBL_stdcol< _T ></code>	68
<code>typed_col< _T ></code>	126
<code>--VDBL_index</code>	
<code>_VDBL_index</code>	
<code>index</code>	
<code>_VDBL_acl</code>	17
<code>_VDBL_aclentry</code>	19
<code>_VDBL_alltype_base</code>	23
<code>_VDBL_alltype< _R ></code>	20
<code>_VDBL_alltype< _T ></code>	20
<code>alltype< _T ></code>	94
<code>_VDBL_col</code>	24
<code>_VDBL_colflags</code>	30
<code>_VDBL_context</code>	31
<code>_VDBL_database</code>	32
<code>database</code>	98
<code>_VDBL_date</code>	36
<code>_VDBL_dateinterval</code>	38
<code>_VDBL_eval_expr</code>	
<code>_VDBL_exprcol</code>	
<code>_VDBL_exprequal</code>	

_VDBL_exprneql	
_VDBL_exprrow	
_VDBL_method< _R >	46
method< _R >	111
_VDBL_mixtype	47
_VDBL_row	57
row	113
_VDBL_table	71
_VDBL_standardtable	59
standard_table	115
_VDBL_viewtable	91
view_table	
table	121
_VDBL_table::col_iterator_base	75
_VDBL_table::col_iterator< _Tp, _Ref, _Ptr >	74
_VDBL_table::row_iterator< _Tp, _Ref, _Ptr >	76
_VDBL_tableflags	76
_VDBL_user	
_VDBL_userflags	78
_VDBL_view	79
_VDBL_hierarchicalview	40
hierarchical_view	104
_VDBL_joinview	
join_view	
_VDBL_standardview	63
view	128
_VDBL_view::col_iterator_base	84
_VDBL_view::col_iterator< _Tp, _Ref, _Ptr >	84
_VDBL_view::default_iterator< _Tp, _Ref, _Ptr >	85
_VDBL_view::row_iterator_base	86
_VDBL_view::row_iterator< _Tp, _Ref, _Ptr >	86
_VDBL_viewdbase	87

viewbase	133
_VDBL_viewflags	89
col_spec	97
triple< _T1, _T2, _T3 >	124

4 Vienna Database Library Compound Index

4.1 Vienna Database Library Compound List

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

_VDBL_colbase (The base class of the internal column structure)	16
_VDBL_acl (Access control list)	17
_VDBL_aclentry (Entry in the access control list)	19
_VDBL_alltype< R > (The templated class for the all_type class)	20
_VDBL_alltype_base (The base class for the all_type class)	23
_VDBL_col (The generic column class (the external structure))	24
_VDBL_colbase< R > (The type dependent base class of the internal column structure)	27
_VDBL_colflags (Additional table information for a column)	30
_VDBL_context (Base class for context objects)	31
_VDBL_database (The database class)	32
_VDBL_date (The VDBL date class)	36
_VDBL_dateinterval (The VDBL date interval class)	38
_VDBL_hierarchicalview (Hierarchical view class)	40
_VDBL_method< R > (Base class for methods usable in _VDBL_mthdcol columns)	46
_VDBL_mixtype (Mixed type)	47
_VDBL_mthdcol< _C, _M, _R > (Generic column class for methods)	54
_VDBL_row (Row class)	57
_VDBL_standardtable (Standard table in databases, constructed from rows and columns)	59
_VDBL_standardview (Standard view onto one table)	63
_VDBL_stdcol< T > (Generic column class for constant values)	68
_VDBL_table (The base class describing database tables)	71

_VDBL_table::col_iterator< _Tp, _Ref, _Ptr >	74
_VDBL_table::col_iterator_base	75
_VDBL_table::row_iterator< _Tp, _Ref, _Ptr >	76
_VDBL_tableflags (Flags for one table)	76
_VDBL_userflags (The permission flags for a user)	78
_VDBL_view (Base class of all views)	79
_VDBL_view::col_iterator< _Tp, _Ref, _Ptr >	84
_VDBL_view::col_iterator_base	84
_VDBL_view::default_iterator< _Tp, _Ref, _Ptr >	85
_VDBL_view::row_iterator< _Tp, _Ref, _Ptr >	86
_VDBL_view::row_iterator_base	86
_VDBL_viewdbase (A view to a complete database)	87
_VDBL_viewflags (Flags for one view)	89
_VDBL_viewtable	91
alltype< _T > (The templated alltype class)	94
col_base< _R > (Column base class)	95
col_spec (Column specification)	97
database (The database class)	98
hierarchical_view (Hierarchical view class onto a stack of tables)	104
method< _R > (Base class for methods usable in method columns)	111
method_col< _M > (External name for computed columns)	112
row (Class implementing table rows)	113
standard_table (Standard table of a database)	115
table (Base class for tables in a database)	121
triple< _T1, _T2, _T3 > (Triple holds three objects of arbitrary type)	124
typed_col< _T > (External name for constant data columns)	126
view (Standard view class onto a single table)	128
viewdbase (A view to a complete database)	133

5 Vienna Database Library File Index

5.1 Vienna Database Library File List

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

database	136
db_alltype	136
db_col	137
db_context	137
db_expression	??
db_hrview	138
db_index	138
db_joinview	139
db_method	139
db_row	140
db_selector	??
db_table	140
db_user	??
db_view	141
dbs	??
triple	141
vdbl_alltype.h	142
vdbl_col.h	145
vdbl_config.h	147
vdbl_context.h	147
vdbl_database.h	149
vdbl_dbs.h	??
vdbl_expression.h	150
vdbl_extradocu.h	??
vdbl_hrview.h	151
vdbl_index.h	152

vdbl_joinview.h	153
vdbl_method.h	154
vdbl_row.h	155
vdbl_selector.h	??
vdbl_stview.h	156
vdbl_table.h	157
vdbl_triple.h	158
vdbl_types.h	160
vdbl_user.h	162
vdbl_view.h	162
vdbl_viewdbase.h	164
vdbl_vtable.h	165
viewdbase	165

6 Vienna Database Library Module Documentation

6.1 Classes and types for external use

Compounds

- class [alltype](#)
The templated alltype class.
- class [col_base](#)
column base class.
- class [col_spec](#)
column specification.
- class [database](#)
the database class.
- class [hierarchical_view](#)
hierarchical [view](#) class onto a stack of tables.
- class [method](#)
base class for methods usable in `method` columns.
- class [method_col](#)
external name for computed columns.

- class `row`
class implementing `table` rows.
- class `standard_table`
standard `table` of a `database`.
- class `table`
base class for tables in a `database`.
- class `typed_col`
external name for constant data columns.
- class `view`
standard view class onto a single `table`.
- class `viewdbase`
a `view` to a complete `database`.

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.

Enumerations

- enum `_V_enum`
different view properties.

6.1.1 Detailed Description

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

6.1.2 Typedef Documentation

6.1.2.1 typedef `_VDBL_acl acl`

this is the external name for access control lists

Definition at line 957 of file `vdbl_database.h`.

6.1.2.2 typedef `_VDBL_aclentry aclentry`

this is the external name for access control list entries

Definition at line 951 of file `vdbl_database.h`.

6.1.2.3 typedef [_VDBL_alltype_base](#) alltype_base

The base class of the [alltype](#) templated classes

Definition at line 593 of file [vdbl_alltype.h](#).

6.1.2.4 typedef [_VDBL_col](#) col

this is the external name of the column class

Definition at line 570 of file [vdbl_col.h](#).

6.1.2.5 typedef [_VDBL_colflags](#) colflags

This type describes the additional properties of a column in a [table](#).

Definition at line 138 of file [vdbl_types.h](#).

6.1.2.6 typedef [_VDBL_colid](#) colid

The column id type

Definition at line 123 of file [vdbl_types.h](#).

6.1.2.7 typedef [_VDBL_context](#) context

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

Definition at line 87 of file [vdbl_context.h](#).

6.1.2.8 typedef [_VDBL_date](#) date

the 'official' name of the date class

Definition at line 574 of file [vdbl_alltype.h](#).

6.1.2.9 typedef [_VDBL_dateinterval](#) dateinterval

the 'official' name of the dateinterval class

Definition at line 580 of file [vdbl_alltype.h](#).

6.1.2.10 typedef [_VDBL_mixtype](#) mixtype

the official name of the mixtype class

Definition at line 586 of file [vdbl_alltype.h](#).

6.1.2.11 typedef [_VDBL_rowid](#) rowid

The [row](#) id type

Definition at line 128 of file [vdbl_types.h](#).

6.1.2.12 typedef [_VDBL_stdcol](#)<[mixtype](#)> standard_col

the standard column holds constant data of type [mixtype](#)

Definition at line 577 of file vdbl_col.h.

6.1.2.13 typedef `_VDBL_tableflags` `tableflags`

this is the external name for `table` flags

Definition at line 944 of file vdbl_database.h.

6.1.2.14 typedef `_VDBL_tableid` `tableid`

The `table` id type

Definition at line 133 of file vdbl_types.h.

6.1.2.15 typedef `_VDBL_userflags` `userflags`

this is the external name for user flags

Definition at line 932 of file vdbl_database.h.

6.1.2.16 typedef `_VDBL_userid` `userid`

The user id type

Definition at line 113 of file vdbl_types.h.

6.1.2.17 typedef `_VDBL_viewflags` `viewflags`

this is the external name for `view` flags

Definition at line 938 of file vdbl_database.h.

6.1.2.18 typedef `_VDBL_viewid` `viewid`

The `view` id type

Definition at line 118 of file vdbl_types.h.

6.1.3 Enumeration Type Documentation

6.1.3.1 enum `_V_enum`

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

- `V_window`: This `view` looks through to a `table`. It is possible 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 65 of file vdbl_view.h.

6.2 Classes and types for internal use

Compounds

- class `__VDBL_colbase`
The base class of the internal column structure.
- class `_VDBL_acl`
Access control list.
- class `_VDBL_aclentry`
entry in the access control list.
- class `_VDBL_alltype`
The templated class for the all_type class.
- class `_VDBL_alltype_base`
The base class for the all_type class.
- class `_VDBL_col`
The generic column class (the external structure).
- class `_VDBL_colbase`
The type dependent base class of the internal column structure.
- class `_VDBL_colflags`
additional [table](#) information for a column.
- class `_VDBL_context`
base class for context objects.
- class `_VDBL_database`
the [database](#) class.
- class `_VDBL_date`
The VDBL date class.
- class `_VDBL_dateinterval`
The VDBL date interval class.
- class `_VDBL_hierarchicalview`
hierarchical [view](#) class.
- class `_VDBL_method`
base class for methods usable in `_VDBL_mthdcol` columns.
- class `_VDBL_mixtype`
mixed type.
- class `_VDBL_mthdcol`

generic column class for methods.

- class `_VDBL_row`
row class.
- class `_VDBL_standardtable`
standard table in databases, constructed from rows and columns.
- class `_VDBL_standardview`
standard view onto one table.
- class `_VDBL_stdcol`
generic column class for constant values.
- class `_VDBL_table`
the base class describing database tables.
- class `_VDBL_tableflags`
flags for one table.
- class `_VDBL_userflags`
The permission flags for a user.
- class `_VDBL_view`
base class of all views.
- class `_VDBL_viewdbase`
a view to a complete database.
- class `_VDBL_viewflags`
flags for one view.

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`

Functions

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

6.2.1 Detailed Description

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

6.2.2 Typedef Documentation

6.2.2.1 typedef uint64_t _VDBL_colid

The column id type

Definition at line 92 of file vdbl_types.h.

6.2.2.2 typedef uint64_t _VDBL_rowid

The [row](#) id type

Definition at line 96 of file vdbl_types.h.

6.2.2.3 typedef uint32_t _VDBL_tableid

The [table](#) id type

Definition at line 100 of file vdbl_types.h.

6.2.2.4 typedef uint32_t _VDBL_userid

The user id type

Definition at line 84 of file vdbl_types.h.

6.2.2.5 typedef uint32_t _VDBL_viewid

The [view](#) id type

Definition at line 88 of file vdbl_types.h.

6.2.3 Function Documentation

6.2.3.1 template<class _C> std::ostream& print_it (std::ostream & o, const _C & 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 _C did not work properly with g++ 3.2.

Definition at line 54 of file vdbl_col.h.

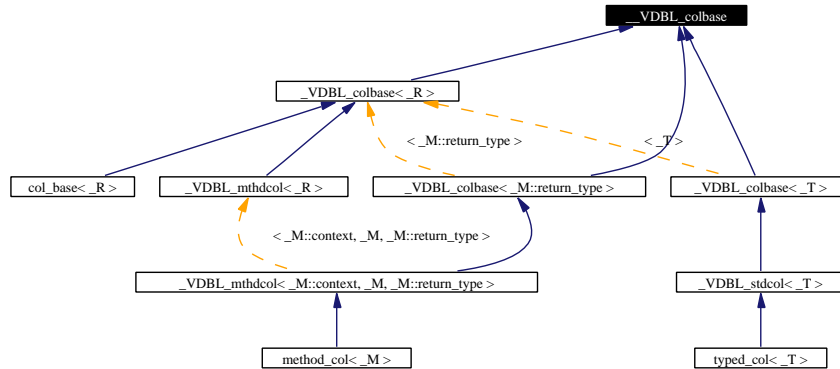
7 Vienna Database Library Class Documentation

7.1 __VDBL_colbase Class Reference

The base class of the internal column structure.

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

Inheritance diagram for __VDBL_colbase:



Public Methods

- virtual `__VDBL_colbase * new_copy () const`
- `__VDBL_colbase ()`
- `__VDBL_colbase (const __VDBL_colbase & __v)`
- virtual `~__VDBL_colbase ()`

7.1.1 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 72 of file `vdbl_col.h`.

7.1.2 Constructor & Destructor Documentation

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

Standard constructor, copy constructor, and destructor

Definition at line 79 of file `vdbl_col.h`.

7.1.2.2 `__VDBL_colbase::__VDBL_colbase (const __VDBL_colbase & __v) [inline]`

Standard constructor, copy constructor, and destructor

Definition at line 80 of file `vdbl_col.h`.

7.1.2.3 `virtual __VDBL_colbase::~__VDBL_colbase () [inline, virtual]`

Standard constructor, copy constructor, and destructor

Definition at line 81 of file `vdbl_col.h`.

7.1.3 Member Function Documentation

7.1.3.1 `virtual __VDBL_colbase* __VDBL_colbase::new_copy () const [inline, virtual]`

This function is used to overload the copy constructor.

Reimplemented in `_VDBL_colbase`.

Definition at line 87 of file `vdbl_col.h`.

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

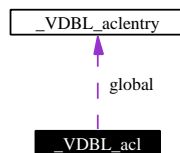
- `vdbl_col.h`

7.2 `_VDBL_acl` Class Reference

Access control list.

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

Collaboration diagram for `_VDBL_acl`:



Public Methods

- `_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`

7.2.1 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 169 of file `vdbl_database.h`.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 `_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 185 of file `vdbl_database.h`.

7.2.2.2 `_VDBL_acl::_VDBL_acl (const _VDBL_acl & a)` [inline]

copy constructor

Definition at line 192 of file `vdbl_database.h`.

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

standard destructor

Definition at line 198 of file `vdbl_database.h`.

7.2.3 Member Function Documentation**7.2.3.1** `_VDBL_acl& _VDBL_acl::operator= (const _VDBL_acl & a)` [inline]

assignment operator

Definition at line 203 of file `vdbl_database.h`.

7.2.4 Member Data Documentation**7.2.4.1** `std::map<_VDBL_colid,_VDBL_aclentry> _VDBL_acl::colwise`

this defines permissions for single columns

Definition at line 179 of file `vdbl_database.h`.

7.2.4.2 `_VDBL_aclentry _VDBL_acl::global`

this defines permissions for all columns

Definition at line 175 of file `vdbl_database.h`.

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

- [vdbl_database.h](#)

7.3 `_VDBL_aclentry` Class Reference

entry in the access control list.

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

Public Methods

- [_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`

7.3.1 Detailed Description

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

Definition at line 111 of file `vdbl_database.h`.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 `_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 129 of file `vdbl_database.h`.

7.3.2.2 `_VDBL_aclentry::_VDBL_aclentry (const _VDBL_aclentry & a)` [`inline`]

copy constructor

Definition at line 138 of file `vdbl_database.h`.

7.3.2.3 `virtual _VDBL_aclentry::~_VDBL_aclentry ()` [`inline`, `virtual`]

standard destructor

Definition at line 146 of file `vdbl_database.h`.

7.3.3 Member Function Documentation

7.3.3.1 `_VDBL_aclentry& _VDBL_aclentry::operator= (const _VDBL_aclentry & a)` [`inline`]

assignment operator

Definition at line 151 of file `vdbl_database.h`.

7.3.4 Member Data Documentation

7.3.4.1 `bool _VDBL_aclentry::append`

These flags describe the permissions

Definition at line 120 of file `vdbl_database.h`.

7.3.4.2 `bool _VDBL_aclentry::commit_to_parent`

These flags describe the permissions

Definition at line 122 of file `vdbl_database.h`.

7.3.4.3 `bool _VDBL_aclentry::drop`

These flags describe the permissions

Definition at line 121 of file `vdbl_database.h`.

7.3.4.4 `bool _VDBL_aclentry::read`

These flags describe the permissions

Definition at line 118 of file `vdbl_database.h`.

7.3.4.5 `bool _VDBL_aclentry::write`

These flags describe the permissions

Definition at line 119 of file `vdbl_database.h`.

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

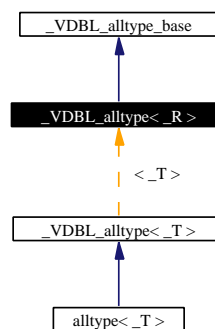
- [vdbl_database.h](#)

7.4 `_VDBL_alltype` Class Template Reference

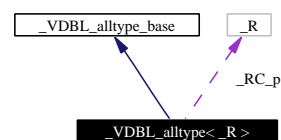
The templated class for the `all_type` class.

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

Inheritance diagram for `_VDBL_alltype`:



Collaboration diagram for `_VDBL_alltype`:

**Public Methods**

- [_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 _Self &p)
- bool `operator!=` (const _Self &p)

7.4.1 Detailed Description

template<class `_R`> class `_VDBL_alltype`< `_R` >

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 90 of file `vdbl_alltype.h`.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `template<class _R> _VDBL_alltype< _R >::_VDBL_alltype () [inline]`

This is the empty constructor which produces an empty `all_type`

Definition at line 107 of file `vdbl_alltype.h`.

7.4.2.2 `template<class _R> _VDBL_alltype< _R >::_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 113 of file `vdbl_alltype.h`.

7.4.2.3 `template<class _R> _VDBL_alltype< _R >::_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 120 of file `vdbl_alltype.h`.

7.4.2.4 `template<class _R> virtual _VDBL_alltype< _R >::~~_VDBL_alltype () [inline, virtual]`

The destructor removes the allocated data to prevent memory leaks.

Definition at line 125 of file `vdbl_alltype.h`.

7.4.3 Member Function Documentation

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

This [method](#) returns a const reference to the stored data

Definition at line 136 of file `vdbl_alltype.h`.

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

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

Definition at line 131 of file `vdbl_alltype.h`.

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

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

Definition at line 160 of file `vdbl_alltype.h`.

7.4.3.4 `template<class _R> void _VDBL_alltype<_R>::operator= (const cont_type * p)` [inline]

The assignment operators can take either pointers to the @cont_type 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 148 of file `vdbl_alltype.h`.

7.4.3.5 `template<class _R> void _VDBL_alltype<_R>::operator= (const void * p)` [inline]

The assignment operators can take either pointers to the @cont_type 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 145 of file `vdbl_alltype.h`.

7.4.3.6 `template<class _R> bool _VDBL_alltype<_R>::operator==(const _Self & p)` [inline]

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

Definition at line 157 of file `vdbl_alltype.h`.

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

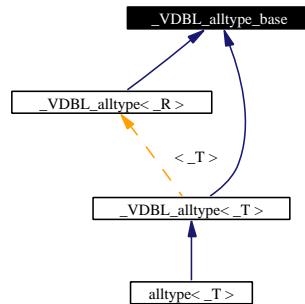
- [vdbl_alltype.h](#)

7.5 `_VDBL_alltype_base` Class Reference

The base class for the `all_type` class.

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

Inheritance diagram for `_VDBL_alltype_base`:



Public Methods

- `_VDBL_alltype_base ()`
- `virtual ~_VDBL_alltype_base ()`

7.5.1 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 54 of file `vdbl_alltype.h`.

7.5.2 Constructor & Destructor Documentation

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

standard constructor

Definition at line 60 of file `vdbl_alltype.h`.

7.5.2.2 `virtual _VDBL_alltype_base::~~_VDBL_alltype_base ()` [inline, virtual]

standard destructor

Definition at line 64 of file `vdbl_alltype.h`.

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

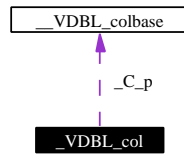
- [vdbl_alltype.h](#)

7.6 `_VDBL_col` Class Reference

The generic column class (the external structure).

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

Collaboration diagram for `_VDBL_col`:



Public Methods

- `_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 _R> void set (_VDBL_colbase< _R > *_p)`
- `template<class _R> void get (_R &c) const`
- `template<class _R> void def (_R &d) const`
- `template<class _R> void get_ptr (_R const *&c) const`
- `template<class _R> void get_copy (_R *&p) const`
- `template<class _R> void def_copy (_R *&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)`

7.6.1 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<_T>` classes.

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

Definition at line 241 of file `vdbl_col.h`.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 `_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 260 of file `vdbl_col.h`.

7.6.2.2 `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 365 of file `vdbl_col.h`.

7.6.3 Member Function Documentation

7.6.3.1 `template<class _R> void _VDBL_col::def(_R & d) const` [inline]

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

Definition at line 379 of file `vdbl_col.h`.

7.6.3.2 `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 396 of file `vdbl_col.h`.

7.6.3.3 `template<class _R> void _VDBL_col::def_copy(_R *& 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 390 of file `vdbl_col.h`.

7.6.3.4 `template<class _R> void _VDBL_col::get(_R & c) const` [inline]

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

Definition at line 376 of file `vdbl_col.h`.

7.6.3.5 `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 393 of file `vdbl_col.h`.

7.6.3.6 `template<class R> void _VDBL_col::get_copy (R *& 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 386 of file `vdbl_col.h`.

7.6.3.7 `template<class R> void _VDBL_col::get_ptr (R 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 382 of file `vdbl_col.h`.

7.6.3.8 `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 348 of file `vdbl_col.h`.

7.6.3.9 `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 341 of file `vdbl_col.h`.

7.6.3.10 `template<class R> void _VDBL_col::set (_VDBL_colbase< R > * 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 284 of file `vdbl_col.h`.

7.6.4 Friends And Related Function Documentation**7.6.4.1** `std::ostream& operator<< (std::ostream & o, const _VDBL_col & c)` [friend]

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

Definition at line 359 of file `vdbl_col.h`.

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

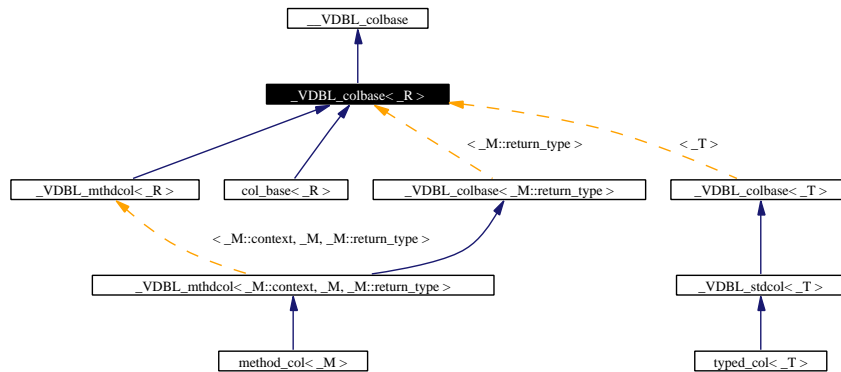
- [vdbl_col.h](#)

7.7 `_VDBL_colbase` Class Template Reference

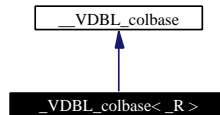
The type dependent base class of the internal column structure.

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

Inheritance diagram for `_VDBL_colbase`:



Collaboration diagram for `_VDBL_colbase`:



Public Types

- typedef `_R` `return_type`
return_type is the type of object stored.

Public Methods

- virtual `_Self * new_copy ()` const
- virtual void `setcontext (const context *_c, const _VDBL_row *_r)` VDBL_PURE_VIRTUAL virtual void `get(return_type &c)` const VDBL_PURE_VIRTUAL virtual void `def(return_type &d)` const VDBL_PURE_VIRTUAL virtual void `get_ptr(return_type const *&c)` const VDBL_PURE_VIRTUAL virtual 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
- `_VDBL_colbase ()`
- `_VDBL_colbase (const _Self &_c)`
- virtual `~_VDBL_colbase ()`

7.7.1 Detailed Description

`template<class _R> class _VDBL_colbase< _R >`

`_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 131 of file `vdbl_col.h`.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 `template<class _R> _VDBL_colbase< _R >::_VDBL_colbase () [inline]`

standard constructor, copy constructor, and destructor

Definition at line 143 of file `vdbl_col.h`.

7.7.2.2 `template<class _R> _VDBL_colbase< _R >::_VDBL_colbase (const _Self & _c) [inline]`

standard constructor, copy constructor, and destructor

Definition at line 144 of file `vdbl_col.h`.

7.7.2.3 `template<class _R> virtual _VDBL_colbase< _R >::~~_VDBL_colbase () [inline, virtual]`

standard constructor, copy constructor, and destructor

Definition at line 145 of file `vdbl_col.h`.

7.7.3 Member Function Documentation

7.7.3.1 `template<class _R> virtual void _VDBL_colbase< _R >::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.

Definition at line 209 of file `vdbl_col.h`.

7.7.3.2 `template<class _R> virtual void _VDBL_colbase< _R >::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 deleted by the user to avoid memory leaks.

Definition at line 187 of file `vdbl_col.h`.

7.7.3.3 `template<class _R> virtual void _VDBL_colbase< _R >::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.

Definition at line 196 of file `vdbl_col.h`.

7.7.3.4 `template<class _R> virtual _Self* _VDBL_colbase< _R >::new_copy () const` [`inline`, `virtual`]

`new_copy` is the clone operation for copy-constructor overloading.

Reimplemented from `__VDBL_colbase`.

Reimplemented in `_VDBL_stdcol`.

Definition at line 151 of file `vdbl_col.h`.

7.7.3.5 `template<class _R> virtual const std::type_info& _VDBL_colbase< _R >::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 220 of file `vdbl_col.h`.

7.7.3.6 `template<class _R> virtual void _VDBL_colbase< _R >::setcontext (const context * c, const _VDBL_row * r) const` [`inline`, `virtual`]

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 156 of file `vdbl_col.h`.

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

- [vdbl_col.h](#)

7.8 `_VDBL_colflags` Class Reference

additional [table](#) information for a column.

```
#include <vdbl_types.h>
```

Public Methods

- [_VDBL_colflags](#) (`bool` `_d`=false, `bool` `_mi`=false)
- [~_VDBL_colflags](#) ()

Public Attributes

- `bool` [master_index](#)
- `bool` [has_default](#)

7.8.1 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 65 of file `vdbl_types.h`.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 `_VDBL_colflags::_VDBL_colflags (bool _d = false, bool _mi = false)` [inline]

standard constructor, optionally setting the entries

Definition at line 75 of file `vdbl_types.h`.

7.8.2.2 `_VDBL_colflags::~_VDBL_colflags ()` [inline]

standard destructor

Definition at line 78 of file `vdbl_types.h`.

7.8.3 Member Data Documentation

7.8.3.1 `bool _VDBL_colflags::has_default`

see class description

Definition at line 71 of file `vdbl_types.h`.

7.8.3.2 `bool _VDBL_colflags::master_index`

see class description

Definition at line 70 of file `vdbl_types.h`.

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

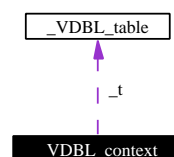
- [vdbl_types.h](#)

7.9 `_VDBL_context` Class Reference

base class for context objects.

```
#include <vdbl_context.h>
```

Collaboration diagram for `_VDBL_context`:



Public Methods

- [_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 \(\)`](#)

7.9.1 Detailed Description

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

Definition at line 50 of file `vdbl_context.h`.

7.9.2 Constructor & Destructor Documentation

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

standard constructor, copy constructor, and destructor.

Definition at line 61 of file `vdbl_context.h`.

7.9.2.2 `_VDBL_context::_VDBL_context (const _VDBL_context & c)` [inline]

standard constructor, copy constructor, and destructor.

Definition at line 62 of file `vdbl_context.h`.

7.9.2.3 `virtual _VDBL_context::~~_VDBL_context ()` [inline, virtual]

standard constructor, copy constructor, and destructor.

Definition at line 63 of file `vdbl_context.h`.

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

constructor which explicitly sets the `table` pointer

Definition at line 69 of file `vdbl_context.h`.

7.9.3 Member Function Documentation

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

set the `table` pointer

Definition at line 79 of file `vdbl_context.h`.

7.9.3.2 `const _VDBL_table* _VDBL_context::table () const` [inline]

retrieve a pointer to the `table` this object belongs to

Definition at line 74 of file `vdbl_context.h`.

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

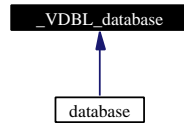
- [vdbl_context.h](#)

7.10 `_VDBL_database` Class Reference

the `database` class.

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

Inheritance diagram for `_VDBL_database`:



Public Methods

- `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 Methods

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

Friends

- class `_VDBL_viewbase`

7.10.1 Detailed Description

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

Definition at line 336 of file `vdbl_database.h`.

7.10.2 Constructor & Destructor Documentation

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

standard constructor

Definition at line 759 of file `vdbl_database.h`.

7.10.2.2 `_VDBL_database::_VDBL_database(const _VDBL_database & _t)` [inline]

copy constructor

Definition at line 765 of file `vdbl_database.h`.

7.10.2.3 `virtual _VDBL_database::~~_VDBL_database()` [inline, virtual]

standard destructor

Definition at line 779 of file `vdbl_database.h`.

7.10.3 Member Function Documentation

7.10.3.1 `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 405 of file `vdbl_database.h`.

7.10.3.2 `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.

Definition at line 640 of file `vdbl_database.h`.

7.10.3.3 `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.

Definition at line 508 of file `vdbl_database.h`.

7.10.3.4 `bool _VDBL_database::drop_table(const _VDBL_tableid & _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 495 of file `vdbl_database.h`.

7.10.3.5 `bool _VDBL_database::drop_table (const _D_tables::iterator & _t, const _D_table_names::iterator & _tn, const _VDBL_userid & _Cu) [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 470 of file `vdbl_database.h`.

7.10.3.6 `bool _VDBL_database::drop_view (const std::string & _Ci, const _VDBL_userid & _Cu) [inline]`

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

Definition at line 707 of file `vdbl_database.h`.

7.10.3.7 `bool _VDBL_database::drop_view (const _VDBL_viewid & _Ci, const _VDBL_userid & _Cu) [inline]`

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

Definition at line 694 of file `vdbl_database.h`.

7.10.3.8 `bool _VDBL_database::drop_view (const _D_views::iterator & _v, const _D_view_names::iterator & _vn, const _VDBL_userid & _Cu) [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 669 of file `vdbl_database.h`.

7.10.3.9 `_VDBL_table* _VDBL_database::get_table (const std::string & _Ci, const _VDBL_userid & _Cu) const [inline]`

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

Reimplemented in `database`.

Definition at line 549 of file `vdbl_database.h`.

7.10.3.10 `_VDBL_table* _VDBL_database::get_table (const _VDBL_tableid & _Ci, const _VDBL_userid & _Cu) const [inline]`

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

Definition at line 536 of file `vdbl_database.h`.

7.10.3.11 `_VDBL_tableid _VDBL_database::get_tableid (const std::string & _Ci, const _VDBL_userid & _Cu) const [inline]`

return the `table` id for a given name

Definition at line 447 of file `vdbl_database.h`.

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

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

Definition at line 391 of file `vdbl_database.h`.

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

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

Definition at line 392 of file `vdbl_database.h`.

7.10.3.14 `_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`.

Definition at line 746 of file `vdbl_database.h`.

7.10.3.15 `_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 734 of file `vdbl_database.h`.

7.10.3.16 `_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 626 of file `vdbl_database.h`.

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

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

Definition at line 393 of file `vdbl_database.h`.

7.10.3.18 `bool` `_VDBL_database::has_table (const std::string & C.i, const _VDBL_userid & C.u) const` [`inline`]

check whether the `table` `.C.i` exists

Definition at line 529 of file `vdbl_database.h`.

7.10.3.19 `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 517 of file `vdbl_database.h`.

7.10.3.20 `bool` `_VDBL_database::has_view (const std::string & C.i, const _VDBL_userid & C.u) const` [`inline`]

check whether the `view` `.C.i` exists

Definition at line 728 of file `vdbl_database.h`.

7.10.3.21 `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 716 of file `vdbl_database.h`.

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

- [vdbl_database.h](#)

7.11 `_VDBL_date` Class Reference

The VDBL date class.

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

Public Methods

- [_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](#)

7.11.1 Detailed Description

The date base class for the [database](#)

Definition at line 172 of file `vdbl_alltype.h`.

7.11.2 Member Function Documentation

7.11.2.1 `_VDBL_date& _VDBL_date::operator+` (const `_VDBL_dateinterval` & `_v`) const

These operators add or subtract time differences to a date

7.11.2.2 `_VDBL_date& _VDBL_date::operator+=` (const `_VDBL_dateinterval` & `_v`)

These operators add or subtract time differences to a date

7.11.2.3 `_VDBL_date& _VDBL_date::operator-` (const `_VDBL_dateinterval` & `_v`) const

These operators add or subtract time differences to a date

7.11.2.4 `_VDBL_dateinterval` `_VDBL_date::operator-` (const `_VDBL_date` & `_v`) const

This [method](#) computes the time difference between two dates.

7.11.2.5 `_VDBL_date&` `_VDBL_date::operator+=` (const `_VDBL_dateinterval` & `_v`)

These operators add or subtract time differences to a date

7.11.3 Member Data Documentation

7.11.3.1 signed char `_VDBL_date::day`

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

Definition at line 185 of file `vdbl_alltype.h`.

7.11.3.2 unsigned int `_VDBL_date::microseconds`

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

Definition at line 187 of file `vdbl_alltype.h`.

7.11.3.3 signed char `_VDBL_date::month`

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

Definition at line 185 of file `vdbl_alltype.h`.

7.11.3.4 unsigned int `_VDBL_date::seconds`

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

Definition at line 186 of file `vdbl_alltype.h`.

7.11.3.5 int `_VDBL_date::timezone`

This defines the timezone in minutes deviation from GMT.

Definition at line 178 of file `vdbl_alltype.h`.

7.11.3.6 int `_VDBL_date::year`

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

Definition at line 184 of file `vdbl_alltype.h`.

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

- [vdbl_alltype.h](#)

7.12 `_VDBL_dateinterval` Class Reference

The VDBL date interval class.

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

Public Methods

- `_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`

7.12.1 Detailed Description

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

Definition at line 240 of file `vdbl_alltype.h`.

7.12.2 Member Function Documentation**7.12.2.1 `_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 $(d1-d2)/2$.

7.12.2.2 `_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 $(d1-d2)/2$.

7.12.2.3 `_VDBL_date _VDBL_dateinterval::operator+ (const _VDBL_date & _d) const`

Add a date difference to a date

7.12.2.4 `_VDBL_dateinterval& _VDBL_dateinterval::operator+ (const _VDBL_dateinterval & _v) const`

Date differences can be added/subtracted using this operator

7.12.2.5 `_VDBL_dateinterval& _VDBL_dateinterval::operator+= (const _VDBL_dateinterval & _v)`

Date differences can be added/subtracted using this operator

7.12.2.6 `_VDBL_dateinterval& _VDBL_dateinterval::operator-` (const `_VDBL_dateinterval & _y`) const

Date differences can be added/subtracted using this operator

7.12.2.7 `_VDBL_dateinterval& _VDBL_dateinterval::operator+=` (const `_VDBL_dateinterval & _y`)

Date differences can be added/subtracted using this operator

7.12.2.8 `_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 $(d1-d2)/2$.

7.12.2.9 `_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 $(d1-d2)/2$.

7.12.3 Member Data Documentation

7.12.3.1 `short int _VDBL_dateinterval::days`

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

Definition at line 249 of file `vdbl_alltype.h`.

7.12.3.2 `int _VDBL_dateinterval::microseconds`

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

Definition at line 251 of file `vdbl_alltype.h`.

7.12.3.3 `int _VDBL_dateinterval::seconds`

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

Definition at line 250 of file `vdbl_alltype.h`.

7.12.3.4 `int _VDBL_dateinterval::years`

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

Definition at line 248 of file `vdbl_alltype.h`.

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

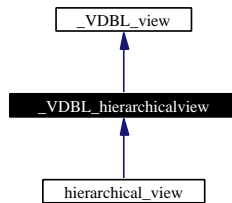
- [vdbl_alltype.h](#)

7.13 `_VDBL_hierarchicalview` Class Reference

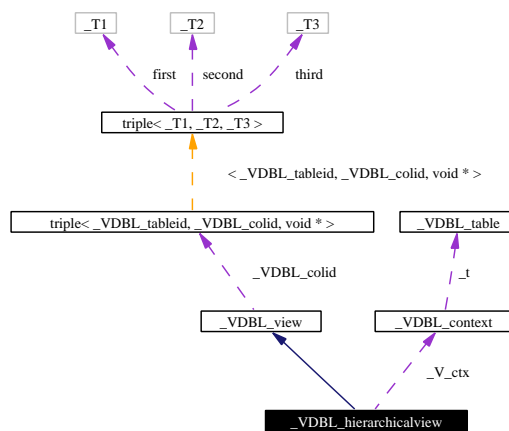
hierarchical [view](#) class.

```
#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 Methods

- `_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` ()
- const `std::type_info` & `get_colinfo` (`const std::string &_C_n, triple< bool, _VDBL.colid, _VDBL.colflags > &_r`) const
- `bool` `remove` (`std::pair< _VDBL.tableid, _VDBL.rowid >` _r)
- template<class `_R`> `bool` `get` (`const std::pair< _VDBL.tableid, _VDBL.rowid > &_ri, const _VDBL.colid &_ci, _R &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 Methods

- `triple< _VDBL_tableid, _VDBL_colid, void * >` `_next_def_col` (const `_VDBL_tableid` &`_t`, const `_VDBL_colid` &`_c`, void *`_d`) const
- `triple< _VDBL_tableid, _VDBL_colid, void * >` `_prev_def_col` (const `_VDBL_tableid` &`_t`, const `_VDBL_colid` &`_c`, void *`_d`) const
- void * `_copy_def_data` (void *`_d`) const
- `triple< _VDBL_tableid, _VDBL_colid, void * >` `_next_col` (const `_VDBL_tableid` &`_t`, const `_VDBL_rowid` &`_r`, const `_VDBL_colid` &`_c`, void *`_d`) const
- `triple< _VDBL_tableid, _VDBL_colid, void * >` `_prev_col` (const `_VDBL_tableid` &`_t`, const `_VDBL_rowid` &`_r`, const `_VDBL_colid` &`_c`, void *`_d`) const
- void * `_copy_col_data` (void *`_d`) const
- `triple< _VDBL_tableid, _VDBL_rowid, void * >` `_next_row` (const `_VDBL_tableid` &`_t`, const `_VDBL_rowid` &`_r`, void *`_d`) const
- `triple< _VDBL_tableid, _VDBL_rowid, void * >` `_prev_row` (const `_VDBL_tableid` &`_t`, const `_VDBL_rowid` &`_r`, 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`

7.13.1 Detailed Description

This class implements a hierarchical [view](#) onto various tables.

Definition at line 48 of file `vdbl_hrview.h`.

7.13.2 Member Typedef Documentation

7.13.2.1 `typedef std::pair<std::string, _VDBL_col> _VDBL_view::T_colspec` [inherited]

This is the description of one column

Definition at line 84 of file `vdbl_view.h`.

7.13.2.2 `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 460 of file `vdbl_view.h`.

7.13.2.3 `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 324 of file `vdbl_view.h`.

7.13.2.4 `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 589 of file `vdbl_view.h`.

7.13.3 Constructor & Destructor Documentation

7.13.3.1 `_VDBL_hierarchicalview::_VDBL_hierarchicalview (const _VDBL_tableid & _ti, _VDBL_table * _t, const _VDBL_context & _c, _V_enum _en)` [inline]

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

Definition at line 341 of file `vdbl_hrview.h`.

7.13.3.2 `_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 initializes 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 368 of file `vdbl_hrview.h`.

7.13.3.3 `_VDBL_hierarchicalview::_VDBL_hierarchicalview (const _VDBL_hierarchicalview & _y)` [inline]

copy constructor

Definition at line 394 of file `vdbl_hrview.h`.

7.13.3.4 `virtual _VDBL_hierarchicalview::~~_VDBL_hierarchicalview ()` [inline, virtual]

standard destructor

Definition at line 403 of file `vdbl_hrview.h`.

7.13.4 Member Function Documentation**7.13.4.1** `void* _VDBL_hierarchicalview::_copy_col_data (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 `row` for a `_row_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 289 of file `vdbl_hrview.h`.

7.13.4.2 `void* _VDBL_hierarchicalview::_copy_def_data (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 column for a `_col_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 232 of file `vdbl_hrview.h`.

7.13.4.3 `void* _VDBL_hierarchicalview::_copy_row_data (void * d) const` [inline, protected, virtual]

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 326 of file `vdbl_hrview.h`.

7.13.4.4 `triple<_VDBL_tableid, VDBL_colid, void*> _VDBL_hierarchicalview::_next_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [inline, protected]

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`.

Reimplemented from `_VDBL_view`.

Definition at line 237 of file `vdbl_hrview.h`.

7.13.4.5 `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 previous default of a column a `_default_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 180 of file `vdbl_hrview.h`.

7.13.4.6 `triple<_VDBL_tableid, VDBL_rowid, void*> _VDBL_hierarchicalview::next_row (const _VDBL_tableid & t, const _VDBL_rowid & r, void * d) const` [`inline`, `protected`]

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`.

Reimplemented from `_VDBL_view`.

Definition at line 294 of file `vdbl_hrview.h`.

7.13.4.7 `triple<_VDBL_tableid, VDBL_colid, void*> _VDBL_hierarchicalview::prev_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [`inline`, `protected`]

This function destroys the additional data needed by a `_col_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 259 of file `vdbl_hrview.h`.

7.13.4.8 `triple<_VDBL_tableid, VDBL_colid, void*> _VDBL_hierarchicalview::prev_def_col (const _VDBL_tableid & t, const _VDBL_colid & c, void * d) const` [`inline`, `protected`]

This function destroys the additional data needed by a `_default_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 202 of file `vdbl_hrview.h`.

7.13.4.9 `triple<_VDBL_tableid, VDBL_rowid, void*> _VDBL_hierarchicalview::prev_row (const _VDBL_tableid & t, const _VDBL_rowid & r, void * d) const` [`inline`, `protected`]

This function destroys the additional data needed by a `_row_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 308 of file `vdbl_hrview.h`.

7.13.4.10 `template<class _R> bool _VDBL_hierarchicalview::get (const std::pair< _VDBL_tableid, _VDBL_rowid > & ri, const _VDBL_colid & ci, _R & 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 `_R`.

Definition at line 601 of file `vdbl_hrview.h`.

7.13.4.11 `const std::type_info& _VDBL_hierarchicalview::get_colinfo (const std::string & Cn, triple< bool, _VDBL_colid, _VDBL_colflags > & r) const` [`inline`, `virtual`]

return the type of this `view`

Reimplemented from `_VDBL_view`.

Definition at line 487 of file `vdbl_hrview.h`.

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

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

Definition at line 458 of file `vdbl_hrview.h`.

7.13.4.13 `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 433 of file `vdbl_hrview.h`.

7.13.4.14 `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 408 of file `vdbl_hrview.h`.

7.13.4.15 `bool _VDBL_hierarchicalview::remove (std::pair< _VDBL_tableid, _VDBL_rowid > r)` [inline]

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

Definition at line 519 of file `vdbl_hrview.h`.

7.13.5 Member Data Documentation**7.13.5.1** `_V_cols _VDBL_hierarchicalview::_V_c` [protected]

This contains all columns of the `view`

Definition at line 89 of file `vdbl_hrview.h`.

7.13.5.2 `_V_colxref _VDBL_hierarchicalview::_V_cx` [protected]

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

Definition at line 93 of file `vdbl_hrview.h`.

7.13.5.3 `_V_rows _VDBL_hierarchicalview::_V_r` [protected]

This contains all rows of the `view`

Definition at line 85 of file `vdbl_hrview.h`.

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

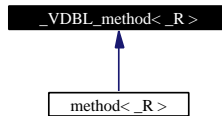
- [vdbl_hrview.h](#)

7.14 `_VDBL_method` Class Template Reference

base class for methods usable in `_VDBL_mthdcol` columns.

```
#include <vdbl_method.h>
```

Inheritance diagram for `_VDBL_method`:



Public Methods

- `_VDBL_method ()`
- `_VDBL_method (const _VDBL_method &_m)`
- virtual `~_VDBL_method ()`
- virtual const `return_type & operator() ()` `VDBL_PURE_VIRTUAL` virtual const `return_type &def()` `VDBL_PURE_VIRTUAL` virtual void `setcontext(const context *c`

7.14.1 Detailed Description

template<class `_R`> class `_VDBL_method<_R>`

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 55 of file `vdbl_method.h`.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 `template<class _R> _VDBL_method<_R>::_VDBL_method () [inline]`

standard constructor

Definition at line 65 of file `vdbl_method.h`.

7.14.2.2 `template<class _R> _VDBL_method<_R>::_VDBL_method (const _VDBL_method<_R> &_m) [inline]`

copy constructor

Definition at line 69 of file `vdbl_method.h`.

7.14.2.3 `template<class _R> virtual _VDBL_method<_R>::~~_VDBL_method () [inline, virtual]`

standard destructor

Definition at line 74 of file `vdbl_method.h`.

7.14.3 Member Function Documentation

7.14.3.1 `template<class _R> virtual const return_type& _VDBL_method<_R>::operator() () const [virtual]`

set the evaluation context and the evaluation [row](#).

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

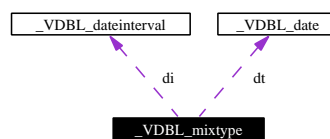
- [vdbl_method.h](#)

7.15 `_VDBL_mixtype` Class Reference

mixed type.

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

Collaboration diagram for `_VDBL_mixtype`:



Public Methods

- 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

7.15.1 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 313 of file `vdbl_alltype.h`.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 `_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 415 of file `vdbl_alltype.h`.

7.15.2.2 `_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 416 of file `vdbl_alltype.h`.

7.15.2.3 `_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 418 of file `vdbl_alltype.h`.

7.15.2.4 `_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 420 of file `vdbl_alltype.h`.

7.15.2.5 `_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 422 of file `vdbl_alltype.h`.

7.15.2.6 `_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 424 of file `vdbl_alltype.h`.

7.15.2.7 `_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 426 of file `vdbl_alltype.h`.

7.15.2.8 `_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 428 of file `vdbl_alltype.h`.

7.15.2.9 `_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 430 of file `vdbl_alltype.h`.

7.15.2.10 `_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 432 of file `vdbl_alltype.h`.

7.15.2.11 `_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 434 of file `vdbl_alltype.h`.

7.15.2.12 `_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 436 of file `vdbl_alltype.h`.

7.15.2.13 `_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 439 of file `vdbl_alltype.h`.

7.15.3 Member Function Documentation**7.15.3.1** `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 555 of file `vdbl_alltype.h`.

7.15.3.2 `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 557 of file `vdbl_alltype.h`.

7.15.3.3 `_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 553 of file `vdbl_alltype.h`.

7.15.3.4 `_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 552 of file `vdbl_alltype.h`.

7.15.3.5 `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 556 of file `vdbl_alltype.h`.

7.15.3.6 `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 548 of file `vdbl_alltype.h`.

7.15.3.7 `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 550 of file `vdbl_alltype.h`.

7.15.3.8 `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 549 of file `vdbl_alltype.h`.

7.15.3.9 `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 551 of file `vdbl_alltype.h`.

7.15.3.10 `_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 517 of file `vdbl_alltype.h`.

7.15.3.11 `_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 511 of file `vdbl_alltype.h`.

7.15.3.12 `_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 505 of file `vdbl_alltype.h`.

7.15.3.13 `_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 499 of file `vdbl_alltype.h`.

7.15.3.14 `_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 493 of file `vdbl_alltype.h`.

7.15.3.15 `_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 487 of file `vdbl_alltype.h`.

7.15.3.16 `_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 482 of file `vdbl_alltype.h`.

7.15.3.17 `_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 477 of file `vdbl_alltype.h`.

7.15.3.18 `_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 472 of file `vdbl_alltype.h`.

7.15.3.19 `_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 467 of file `vdbl_alltype.h`.

7.15.3.20 `_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 462 of file `vdbl_alltype.h`.

7.15.3.21 `_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 457 of file `vdbl_alltype.h`.

7.15.3.22 `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 554 of file `vdbl_alltype.h`.

7.15.3.23 `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 558 of file `vdbl_alltype.h`.

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

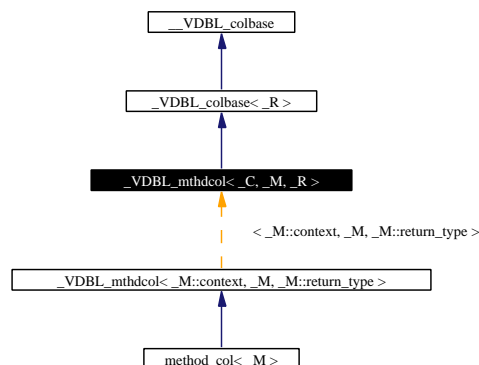
- [vdbl_alltype.h](#)

7.16 `_VDBL_mthdcol` Class Template Reference

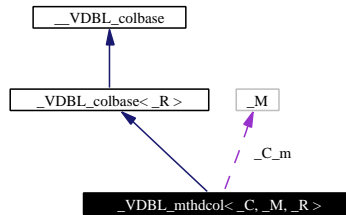
generic column class for methods.

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

Inheritance diagram for `_VDBL_mthdcol`:



Collaboration diagram for `_VDBL_mthdcol`:



Public Methods

- `_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.
- void `def_copy` (`return_type` *&d) const
- virtual void `setcontext` (const `context` *_c, const `_VDBL_row` *_r) `VDBL_PURE_VIRTUAL` virtual void `get`(`return_type` &c) const `VDBL_PURE_VIRTUAL` virtual void `def`(`return_type` &d) const `VDBL_PURE_VIRTUAL` virtual void `get_ptr`(`return_type` const *&c) const `VDBL_PURE_VIRTUAL` virtual void `get_copy`(`return_type` *&c) 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` ()
- `_VDBL_mthdcol` (const `_Self` &_c)
- virtual `~_VDBL_mthdcol` ()

7.16.1 Detailed Description

```
template<class _C, class _M, class _R> class _VDBL_mthdcol< _C, _M, _R >
```

`_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 506 of file `vdbl_col.h`.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 `template<class _C, class _M, class _R> _VDBL_mthdcol< _C, _M, _R >::_VDBL_mthdcol()` [`inline`]

standard constructor, copy constructor, and destructor

Definition at line 525 of file `vdbl_col.h`.

7.16.2.2 `template<class _C, class _M, class _R> _VDBL_mthdcol< _C, _M, _R >::_VDBL_mthdcol(const _Self & _c)` [`inline`]

standard constructor, copy constructor, and destructor

Definition at line 526 of file `vdbl_col.h`.

7.16.2.3 `template<class _C, class _M, class _R> virtual _VDBL_mthdcol< _C, _M, _R >::~~_VDBL_mthdcol()` [`inline`, `virtual`]

standard constructor, copy constructor, and destructor

Definition at line 527 of file `vdbl_col.h`.

7.16.2.4 `template<class _C, class _M, class _R> _VDBL_mthdcol< _C, _M, _R >::_VDBL_mthdcol(const method & _m)` [`inline`]

constructor for explicitly setting the [method](#)

Definition at line 533 of file `vdbl_col.h`.

7.16.3 Member Function Documentation

7.16.3.1 `template<class _R> virtual void _VDBL_colbase< _R >::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 209 of file `vdbl_col.h`.

7.16.3.2 `template<class _C, class _M, class _R> void _VDBL_mthdcol< _C, _M, _R >::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 deleted by the user to avoid memory leaks.

Reimplemented from [_VDBL_colbase](#).

Definition at line 559 of file `vdbl_col.h`.

7.16.3.3 `template<class _R> virtual void _VDBL_colbase< _R >::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 196 of file `vdbl_col.h`.

7.16.3.4 `template<class _C, class _M, class _R> _Self* _VDBL_mthdcol< _C, _M, _R >::new_copy () const` [`inline`, `virtual`]

`new_copy` is the clone operation for copy-constructor overloading.

Reimplemented from `_VDBL_colbase`.

Definition at line 535 of file `vdbl_col.h`.

7.16.3.5 `template<class _R> virtual const std::type_info& _VDBL_colbase< _R >::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 220 of file `vdbl_col.h`.

7.16.3.6 `template<class _R> virtual void _VDBL_colbase< _R >::setcontext (const context * _c, const _VDBL_row * _r) const` [`inline`, `virtual`, `inherited`]

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 156 of file `vdbl_col.h`.

7.16.3.7 `template<class _C, class _M, class _R> virtual void _VDBL_mthdcol< _C, _M, _R >::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 541 of file `vdbl_col.h`.

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

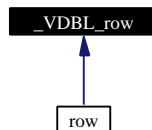
- [vdbl_col.h](#)

7.17 `_VDBL_row` Class Reference

`row` class.

```
#include <vdbl_row.h>
```

Inheritance diagram for `_VDBL_row`:



Public Methods

- [_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)

7.17.1 Detailed Description

This class implements rows of a [table](#) as a map column id -> column

Definition at line 53 of file `vdbl_row.h`.

7.17.2 Constructor & Destructor Documentation

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

standard constructor which optionally initializes the global ACL entry

Definition at line 66 of file `vdbl_row.h`.

7.17.2.2 `_VDBL_row::_VDBL_row` (const `_VDBL_row` &_r) [inline]

copy constructor

Definition at line 70 of file `vdbl_row.h`.

7.17.2.3 virtual `_VDBL_row::~~_VDBL_row` () [inline, virtual]

standard destructor

Definition at line 74 of file `vdbl_row.h`.

7.17.3 Member Function Documentation

7.17.3.1 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 148 of file `vdbl_row.h`.

7.17.3.2 `_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 102 of file `vdbl_row.h`.

7.17.3.3 `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 81 of file `vdbl_row.h`.

7.17.3.4 `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 121 of file `vdbl_row.h`.

7.17.3.5 `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 131 of file `vdbl_row.h`.

7.17.3.6 `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 165 of file `vdbl_row.h`.

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

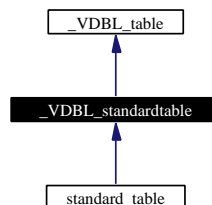
- [vdbl_row.h](#)

7.18 `_VDBL_standardtable` Class Reference

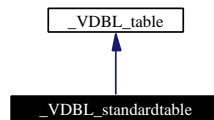
standard [table](#) in databases, constructed from rows and columns.

```
#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 Methods

- `const std::type_info & get_colinfo (const std::string &_C_n, triple< bool, _VDBL_colid, _VDBL_colflags > &r) const`
- `_VDBL_colid get_col_id (const std::string &_C_n) const`
- `_VDBL_standardtable ()`
- `_VDBL_standardtable (const _VDBL_standardtable &t)`
- `template<template< class _Tp, class _AllocTp > class _SequenceCtr, class Allocator1> _VDBL_standardtable (const _SequenceCtr< triple< std::string, _VDBL_col, _VDBL_colflags >, Allocator1 >` &`cc`)
- `virtual ~_VDBL_standardtable ()`
- `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 bool add_col (const std::string &_C_n, const _VDBL_col &c, const _VDBL_colflags &f)`
`VDBL_PURE_VIRTUAL virtual bool modify_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 Methods

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

Friends

- `class _VDBL_view`

7.18.1 Detailed Description

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

Definition at line 543 of file `vdbl_table.h`.

7.18.2 Member Typedef Documentation

7.18.2.1 `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 561 of file `vdbl_table.h`.

7.18.2.2 `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 566 of file `vdbl_table.h`.

7.18.2.3 `typedef Base::col_const_iterator _VDBL_standardtable::col_const_iterator`

const iterator over all columns

Reimplemented from `_VDBL_table`.

Definition at line 571 of file `vdbl_table.h`.

7.18.2.4 `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 320 of file `vdbl_table.h`.

7.18.3 Constructor & Destructor Documentation

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

standard constructor which optionally initializes the global ACL entry

Definition at line 621 of file `vdbl_table.h`.

7.18.3.2 `_VDBL_standardtable::_VDBL_standardtable(const _VDBL_standardtable & _t)` [inline]

copy constructor

Definition at line 625 of file `vdbl_table.h`.

7.18.3.3 `template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1> _VDBL_standardtable::_VDBL_standardtable (const __SequenceCtr< 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 635 of file `vdbl_table.h`.

7.18.3.4 `virtual _VDBL_standardtable::~~_VDBL_standardtable () [inline, virtual]`

standard destructor

Definition at line 665 of file `vdbl_table.h`.

7.18.4 Member Function Documentation

7.18.4.1 `std::pair<std::string, _VDBL_colid> _VDBL_standardtable::next_col (const std::pair< std::string, _VDBL_colid > & __ci) const [inline, virtual]`

This 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 978 of file `vdbl_table.h`.

7.18.4.2 `_VDBL_rowid _VDBL_standardtable::next_row (const _VDBL_rowid & __ci) const [inline, virtual]`

standard constructor

Reimplemented from [_VDBL_table](#).

Definition at line 1013 of file `vdbl_table.h`.

7.18.4.3 `virtual bool _VDBL_table::add_col (const std::string & __C_n, const _VDBL_col & __c, const _VDBL_colflags & __f) const [inline, virtual, inherited]`

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 376 of file `vdbl_table.h`.

7.18.4.4 `_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 607 of file `vdbl_table.h`.

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

generate new unique id's for rows and columns

Definition at line 98 of file `vdbl_table.h`.

7.18.4.6 `const std::type_info& _VDBL_standardtable::get_colinfo (const std::string & C_n, triple< bool, _VDBL_colid, _VDBL_colflags > & r) const` [inline, virtual]

what was the id of the last change to the [table](#)

Reimplemented from [_VDBL_table](#).

Definition at line 584 of file `vdbl_table.h`.

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

generate new unique id's for rows and columns

Definition at line 99 of file `vdbl_table.h`.

7.18.4.8 `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 457 of file `vdbl_table.h`.

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

increment the `last_change` counter.

Definition at line 105 of file `vdbl_table.h`.

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

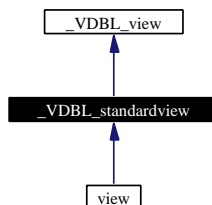
- [vdbl_table.h](#)

7.19 `_VDBL_standardview` Class Reference

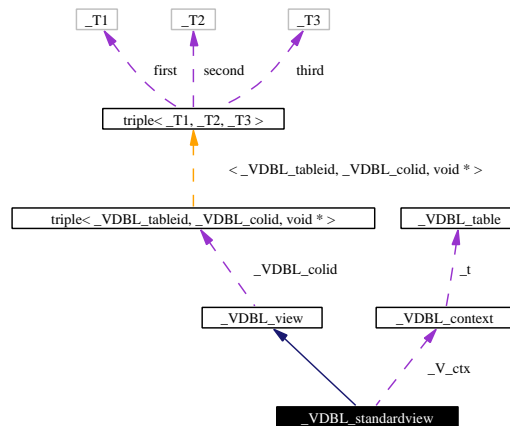
standard [view](#) onto **one** [table](#).

```
#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 Methods

- `_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` ()
- `const std::type_info & get_colinfo` (`const std::string &_C.n, triple< bool, _VDBL_colid, _VDBL_colflags > &_r`) `const`
- `bool remove` (`std::pair< _VDBL_tableid, _VDBL_rowid > _r`)
- `template<class _R> bool get` (`const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _R &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 Methods

- `triple< _VDBL_tableid, _VDBL_colid, void * >` `_next_def_col` (`const _VDBL_tableid &_t, const _VDBL_colid &_c, void *_d`) `const`
- `triple< _VDBL_tableid, _VDBL_colid, void * >` `_prev_def_col` (`const _VDBL_tableid &_t, const _VDBL_colid &_c, void *_d`) `const`

- `void * _copy_def_data (void *_d) const`
- `triple< _VDBL_tableid, _VDBL_colid, void * > _next_col (const _VDBL_tableid &_t, const _VDBL_rowid &_r, const _VDBL_colid &_c, void *_d) const`
- `triple< _VDBL_tableid, _VDBL_colid, void * > _prev_col (const _VDBL_tableid &_t, const _VDBL_rowid &_r, const _VDBL_colid &_c, void *_d) const`
- `void * _copy_col_data (void *_d) const`
- `triple< _VDBL_tableid, _VDBL_rowid, void * > _next_row (const _VDBL_tableid &_t, const _VDBL_rowid &_r, void *_d) const`
- `triple< _VDBL_tableid, _VDBL_rowid, void * > _prev_row (const _VDBL_tableid &_t, const _VDBL_rowid &_r, 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`

7.19.1 Detailed Description

This class implements views onto a single [table](#).

Definition at line 48 of file `vdbl_stview.h`.

7.19.2 Member Typedef Documentation

7.19.2.1 `typedef std::pair<std::string, _VDBL_col> _VDBL_view::T_colspec` [inherited]

This is the description of one column

Definition at line 84 of file `vdbl_view.h`.

7.19.2.2 `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 460 of file `vdbl_view.h`.

7.19.2.3 `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 324 of file `vdbl_view.h`.

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

const iterator over all rows

Definition at line 589 of file `vdbl_view.h`.

7.19.3 Constructor & Destructor Documentation

7.19.3.1 `_VDBL_standardview::_VDBL_standardview (const _VDBL_tableid & _ti, _VDBL_table * _t, const _VDBL_context & _c, _V_enum _e)` [inline]

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

Definition at line 246 of file `vdbl_stview.h`.

7.19.3.2 `_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 initializes 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 269 of file `vdbl_stview.h`.

7.19.3.3 `_VDBL_standardview::_VDBL_standardview (const _VDBL_standardview & _v)` [inline]

copy constructor

Definition at line 288 of file `vdbl_stview.h`.

7.19.3.4 `virtual _VDBL_standardview::~~_VDBL_standardview ()` [inline, virtual]

standard destructor

Definition at line 298 of file `vdbl_stview.h`.

7.19.4 Member Function Documentation

7.19.4.1 `void* _VDBL_standardview::_copy_col_data (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 `row` for a `_row_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 194 of file `vdbl_stview.h`.

7.19.4.2 `void* _VDBL_standardview::_copy_def_data (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 column for a `_col_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 144 of file `vdbl_stview.h`.

7.19.4.3 `void* _VDBL_standardview::_copy_row_data (void * d) const` [`inline`, `protected`, `virtual`]

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 230 of file `vdbl_stview.h`.

7.19.4.4 `triple<_VDBL_tableid, VDBL_colid, void*> _VDBL_standardview::_next_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [`inline`, `protected`]

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`.

Reimplemented from `_VDBL_view`.

Definition at line 149 of file `vdbl_stview.h`.

7.19.4.5 `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 previous default of a column a `_default_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 100 of file `vdbl_stview.h`.

7.19.4.6 `triple<_VDBL_tableid, VDBL_rowid, void*> _VDBL_standardview::_next_row (const _VDBL_tableid & t, const _VDBL_rowid & r, void * d) const` [`inline`, `protected`]

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`.

Reimplemented from `_VDBL_view`.

Definition at line 199 of file `vdbl_stview.h`.

7.19.4.7 `triple<_VDBL_tableid, VDBL_colid, void*> _VDBL_standardview::_prev_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [`inline`, `protected`]

This function destroys the additional data needed by a `_col_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 169 of file `vdbl_stview.h`.

7.19.4.8 `triple<_VDBL_tableid, VDBL_colid, void*> _VDBL_standardview::_prev_def_col (const _VDBL_tableid & t, const _VDBL_colid & c, void * d) const` [`inline`, `protected`]

This function destroys the additional data needed by a `_default_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 120 of file `vdbl_stview.h`.

7.19.4.9 `triple<_VDBL_tableid, _VDBL_rowid, void*> _VDBL_standardview::_prev_row (const _VDBL_tableid & t, const _VDBL_rowid & r, void * d) const` [`inline`, `protected`]

This function destroys the additional data needed by a `_row_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 213 of file `vdbl_stview.h`.

7.19.4.10 `template<class _R> bool _VDBL_standardview::get (const std::pair< _VDBL_tableid, _VDBL_rowid > & ri, const _VDBL_colid & ci, _R & 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 `_R`.

Definition at line 393 of file `vdbl_stview.h`.

7.19.4.11 `const std::type_info& _VDBL_standardview::get_colinfo (const std::string & Cn, triple< bool, _VDBL_colid, _VDBL_colflags > & r) const` [`inline`, `virtual`]

return the type of this `view`

Reimplemented from `_VDBL_view`.

Definition at line 301 of file `vdbl_stview.h`.

7.19.4.12 `bool _VDBL_standardview::remove (std::pair< _VDBL_tableid, _VDBL_rowid > r)` [`inline`]

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

Definition at line 330 of file `vdbl_stview.h`.

7.19.5 Member Data Documentation

7.19.5.1 `_V_cols` `_VDBL_standardview::_V_c` [`protected`]

This contains all columns of the `view`

Definition at line 86 of file `vdbl_stview.h`.

7.19.5.2 `_V_rows` `_VDBL_standardview::_V_r` [`protected`]

This contains all rows of the `view`

Definition at line 82 of file `vdbl_stview.h`.

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

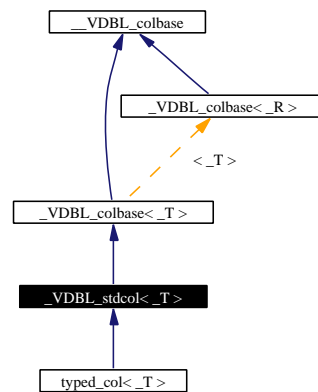
- `vdbl_stview.h`

7.20 `_VDBL_stdcol` Class Template Reference

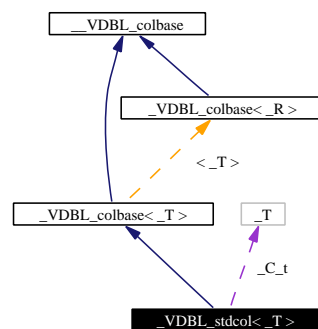
generic column class for constant values.

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

Inheritance diagram for `_VDBL_stdcol`:



Collaboration diagram for `_VDBL_stdcol`:



Public Methods

- `_VDBL_stdcol` (const type &_t)
- `_Self * new_copy` () const
- `void set` (const `_Self` &p)
- `void setcontext` (const `context` *c, const `_VDBL_row` *r)
- `void def` (type &d) const
- `void set` (const type &_t)
- `void set_default` (const type &_t)
- `const type & get_val` () const

- `_VDBL_stdcol` ()
- `_VDBL_stdcol` (const `_Self` &_c)
- `virtual ~_VDBL_stdcol` ()

7.20.1 Detailed Description

template<class `_T`> class `_VDBL_stdcol<_T>`

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

Definition at line 405 of file `vdbl_col.h`.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 `template<class _T> _VDBL_stdcol<_T>::_VDBL_stdcol () [inline]`

standard constructor, copy constructor, destructor

Definition at line 422 of file `vdbl_col.h`.

7.20.2.2 `template<class _T> _VDBL_stdcol<_T>::_VDBL_stdcol (const _Self & _c) [inline]`

standard constructor, copy constructor, destructor

Definition at line 423 of file `vdbl_col.h`.

7.20.2.3 `template<class _T> virtual _VDBL_stdcol<_T>::~~_VDBL_stdcol () [inline, virtual]`

standard constructor, copy constructor, destructor

Definition at line 424 of file `vdbl_col.h`.

7.20.2.4 `template<class _T> _VDBL_stdcol<_T>::_VDBL_stdcol (const type & _t) [inline]`

explicit constructor setting the column's value

Definition at line 430 of file `vdbl_col.h`.

7.20.3 Member Function Documentation

7.20.3.1 `template<class _T> void _VDBL_stdcol<_T>::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 453 of file `vdbl_col.h`.

7.20.3.2 `template<class _T> const type& _VDBL_stdcol<_T>::get_val () const [inline]`

get a const reference to the column value

Definition at line 492 of file `vdbl_col.h`.

7.20.3.3 `template<class _T> _Self* _VDBL_stdcol<_T>::new_copy () const [inline, virtual]`

This function is used to overload the copy constructor.

Reimplemented from `_VDBL_colbase<_T>`.

Definition at line 432 of file `vdbl_col.h`.

7.20.3.4 `template<class _T> void _VDBL_stdcol<_T>::set (const type & _t)` [inline]

set the column value

Definition at line 480 of file `vdbl_col.h`.

7.20.3.5 `template<class _T> void _VDBL_stdcol<_T>::set (const _Self & _p)` [inline]

explicit copy operation

Definition at line 437 of file `vdbl_col.h`.

7.20.3.6 `template<class _T> void _VDBL_stdcol<_T>::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 487 of file `vdbl_col.h`.

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

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

Definition at line 443 of file `vdbl_col.h`.

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

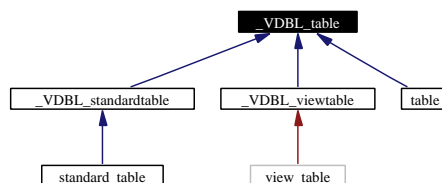
- [vdbl_col.h](#)

7.21 `_VDBL_table` Class Reference

the base class describing `database` tables.

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

Inheritance diagram for `_VDBL_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 Methods

- virtual const `std::type_info &` `get_colinfo` (const `std::string &_C_n`, `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_VIRTUAL` virtual `std`
- virtual `_VDBL_rowid _next_row` (const `_VDBL_rowid &_ci`) const `VDBL_PURE_VIRTUAL` virtual `_VDBL_rowid _prev_row`(const `_VDBL_rowid &_ci`) const `VDBL_PURE_VIRTUAL` virtual `row_const_iterator row_begin`() const `VDBL_PURE_VIRTUAL` virtual `row_const_iterator row_end`() 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< triple< std::string, _VDBL_col, _VDBL_colflags >`, `Allocator1 >` &`_cc`)
- virtual `~_VDBL_table` ()
- virtual bool `add_col` (const `std::string &_C_n`, const `_VDBL_col &_c`, const `_VDBL_colflags &_f`) `VDBL_PURE_VIRTUAL` virtual bool `modify_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 Methods

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

Friends

- class `_VDBL_view`

7.21.1 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 67 of file `vdbl_table.h`.

7.21.2 Member Typedef Documentation

7.21.2.1 `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`.

Definition at line 77 of file `vdbl_table.h`.

7.21.2.2 `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 82 of file `vdbl_table.h`.

7.21.2.3 `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 213 of file `vdbl_table.h`.

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

const iterator over all rows

Definition at line 320 of file `vdbl_table.h`.

7.21.3 Constructor & Destructor Documentation

7.21.3.1 `_VDBL_table::_VDBL_table (const _VDBL_table & t) [inline]`

copy constructor

Definition at line 353 of file `vdbl_table.h`.

7.21.3.2 `template<template< class Tp, class _AllocTp > class _SequenceCtr, class Allocator1> _VDBL_table::_VDBL_table (const _SequenceCtr< triple< std::string, _VDBL_col, _VDBL_colflags>, Allocator1> & ce) [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 362 of file `vdbl_table.h`.

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

standard destructor

Definition at line 369 of file `vdbl_table.h`.

7.21.4 Member Function Documentation

7.21.4.1 `virtual std::pair<std::string, _VDBL_colid> _VDBL_table::next_col (const std::pair<std::string, _VDBL_colid> & _ci) const` [`inline`, `virtual`]

This 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 220 of file `vdbl_table.h`.

7.21.4.2 `virtual _VDBL_rowid _VDBL_table::next_row (const _VDBL_rowid & _ci) const` [`inline`, `virtual`]

standard constructor

Reimplemented in [_VDBL_standardtable](#).

Definition at line 327 of file `vdbl_table.h`.

7.21.4.3 `virtual bool _VDBL_table::add_col (const std::string & _C_n, const _VDBL_col & _c, const _VDBL_colflags & _f) const` [`inline`, `virtual`]

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 376 of file `vdbl_table.h`.

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

generate new unique id's for rows and columns

Definition at line 98 of file `vdbl_table.h`.

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

what was the id of the last change to the `table`

Reimplemented in [_VDBL_standardtable](#).

Definition at line 115 of file `vdbl_table.h`.

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

generate new unique id's for rows and columns

Definition at line 99 of file `vdbl_table.h`.

7.21.4.7 `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 457 of file `vdbl_table.h`.

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

increment the `last_change` counter.

Definition at line 105 of file `vdbl_table.h`.

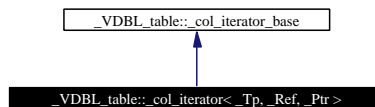
The documentation for this class was generated from the following file:

- [vdbl_table.h](#)

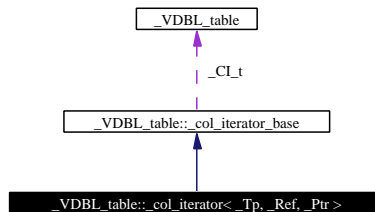
7.22 `_VDBL_table::col_iterator` Struct Template Reference

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

Inheritance diagram for `_VDBL_table::col_iterator`:



Collaboration diagram for `_VDBL_table::col_iterator`:



7.22.1 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 168 of file `vdbl_table.h`.

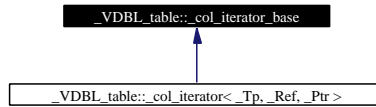
The documentation for this struct was generated from the following file:

- [vdbl_table.h](#)

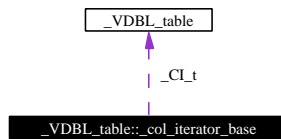
7.23 `_VDBL_table::col_iterator_base` Class Reference

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

Inheritance diagram for `_VDBL_table::_col_iterator_base`:



Collaboration diagram for `_VDBL_table::_col_iterator_base`:



7.23.1 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 131 of file `vdbl_table.h`.

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

- [vdbl_table.h](#)

7.24 `_VDBL_table::_row_iterator` Struct Template Reference

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

7.24.1 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 282 of file `vdbl_table.h`.

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

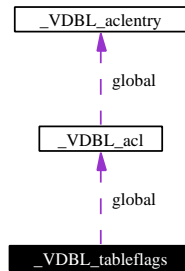
- [vdbl_table.h](#)

7.25 `_VDBL_tableflags` Class Reference

flags for one [table](#).

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

Collaboration diagram for `_VDBL_tableflags`:



Public Methods

- `_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`

7.25.1 Detailed Description

This class describes the additional information for a [table](#) within a [database](#), including access control

Definition at line 217 of file `vdbl_database.h`.

7.25.2 Constructor & Destructor Documentation

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

standard constructor which optionally initializes some members

Definition at line 242 of file `vdbl_database.h`.

7.25.2.2 `_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 249 of file `vdbl_database.h`.

7.25.2.3 `_VDBL_tableflags::_VDBL_tableflags` (const `_VDBL_tableflags` & `_t`) [inline]

copy constructor

Definition at line 257 of file `vdbl_database.h`.

7.25.2.4 `virtual _VDBL_tableflags::~~_VDBL_tableflags ()` [`inline`, `virtual`]

standard destructor

Definition at line 265 of file `vdbl_database.h`.

7.25.3 Member Function Documentation**7.25.3.1** `_VDBL_tableflags& _VDBL_tableflags::operator= (const _VDBL_tableflags & _f)` [`inline`]

assignment operator

Definition at line 270 of file `vdbl_database.h`.

7.25.4 Member Data Documentation**7.25.4.1** `std::map<_VDBL_userid,_VDBL_acl> _VDBL_tableflags::ACLs`

this is an access control list for every single user

Definition at line 236 of file `vdbl_database.h`.

7.25.4.2 `_VDBL_acl _VDBL_tableflags::global`

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

Definition at line 232 of file `vdbl_database.h`.

7.25.4.3 `bool _VDBL_tableflags::temporary`

decides whether this is a temporary [table](#)

Definition at line 224 of file `vdbl_database.h`.

7.25.4.4 `bool _VDBL_tableflags::unrestricted`

decides whether this is [table](#) is completely unrestricted

Definition at line 228 of file `vdbl_database.h`.

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

- [vdbl_database.h](#)

7.26 `_VDBL_userflags` Class Reference

The permission flags for a user.

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

Public Methods

- [_VDBL_userflags \(\)](#)
- [_VDBL_userflags \(const _VDBL_userflags &_a\)](#)
- `virtual ~_VDBL_userflags ()`

Public Attributes

- struct {
} [table_privileges](#)
- struct {
} [view_privileges](#)

7.26.1 Detailed Description

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

Definition at line 59 of file `vdbl_database.h`.

7.26.2 Constructor & Destructor Documentation

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

standard constructor

Definition at line 92 of file `vdbl_database.h`.

7.26.2.2 `_VDBL_userflags::_VDBL_userflags (const _VDBL_userflags & a)` [inline]

copy constructor

Definition at line 96 of file `vdbl_database.h`.

7.26.2.3 `virtual _VDBL_userflags::~_VDBL_userflags ()` [inline, virtual]

standard destructor

Definition at line 102 of file `vdbl_database.h`.

7.26.3 Member Data Documentation

7.26.3.1 `struct { ... } _VDBL_userflags::table_privileges`

The global [table](#) privileges

- read
- write
- modify
- append
- delete

7.26.3.2 `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:

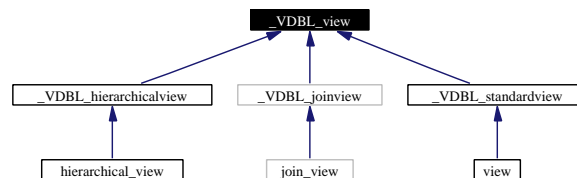
- [vdbl_database.h](#)

7.27 `_VDBL_view` Class Reference

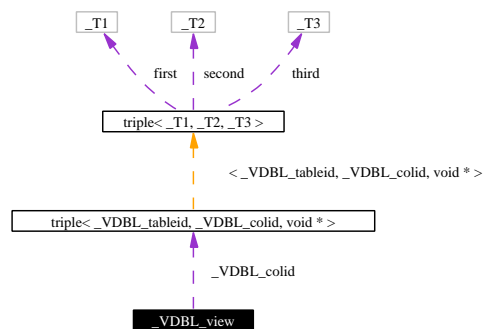
base class of all views.

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

Inheritance diagram for `_VDBL_view`:



Collaboration diagram for `_VDBL_view`:



Public Types

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

Public Methods

- `_VDBL_view` (`_V_enum` `..e=V_independent`)
- `_VDBL_view` (`const` `_VDBL_view` `&..v`)
- virtual `~_VDBL_view` ()
- virtual const `std::type_info` `&` `get_colinfo` (`const` `std::string` `&_C_n`, `triple< bool, _VDBL_colid, _VDBL_colflags >` `&_r`) const `VDBL_PURE_VIRTUAL` virtual `bool` `insert`(`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 Methods

- void `made_change` ()
increment the change counter.
- unsigned int `get_change_ctr` () const
read the change counter.
- virtual `triple<_VDBL_tableid, _VDBL_colid, void * > _next_def_col` (const `_VDBL_tableid` &t, const `_VDBL_colid` &c, void *d) const `VDBL_PURE_VIRTUAL` virtual `triple<_VDBL_tableid`
- virtual `triple<_VDBL_tableid, _VDBL_colid, void * > void * _prev_def_col` (const `_VDBL_tableid` &t, const `_VDBL_colid` &c, void *d) const `VDBL_PURE_VIRTUAL` virtual void `_destroy_def_data`(void *&d) const
- virtual void * `_copy_def_data` (void *d) const `VDBL_PURE_VIRTUAL` virtual `triple<_VDBL_tableid`
- virtual void void * `_next_col` (const `_VDBL_tableid` &t, const `_VDBL_rowid` &r, const `_VDBL_colid` &c, void *d) const `VDBL_PURE_VIRTUAL` virtual `triple<_VDBL_tableid`
- virtual void void void * `_prev_col` (const `_VDBL_tableid` &t, const `_VDBL_rowid` &r, const `_VDBL_colid` &c, void *d) const `VDBL_PURE_VIRTUAL` virtual void `_destroy_col_data`(void *&d) const
- virtual void * `_copy_col_data` (void *d) const `VDBL_PURE_VIRTUAL` virtual `triple<_VDBL_tableid`
- virtual void void * `_next_row` (const `_VDBL_tableid` &t, const `_VDBL_rowid` &c, void *d) const `VDBL_PURE_VIRTUAL` virtual `triple<_VDBL_tableid`
- virtual void void void * `_prev_row` (const `_VDBL_tableid` &t, const `_VDBL_rowid` &c, void *d) const `VDBL_PURE_VIRTUAL` virtual void `_destroy_row_data`(void *&d) const
- virtual void * `_copy_row_data` (void *d) const `VDBL_PURE_VIRTUAL` public

7.27.1 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 78 of file `vdbl_view.h`.

7.27.2 Member Typedef Documentation**7.27.2.1 `typedef std::pair<std::string, _VDBL_col> _VDBL_view::_T_colspec`**

This is the description of one column

Definition at line 84 of file `vdbl_view.h`.

7.27.2.2 `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 460 of file `vdbl_view.h`.

7.27.2.3 `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 324 of file `vdbl_view.h`.

7.27.2.4 `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 589 of file `vdbl_view.h`.

7.27.3 Constructor & Destructor Documentation

7.27.3.1 `_VDBL_view::_VDBL_view (_V_enum _e = V_independent) [inline]`

standard constructor which optionally initializes the [view](#) type

Definition at line 595 of file `vdbl_view.h`.

7.27.3.2 `_VDBL_view::_VDBL_view (const _VDBL_view & _v) [inline]`

copy constructor

Definition at line 599 of file `vdbl_view.h`.

7.27.3.3 `virtual _VDBL_view::~~_VDBL_view () [inline, virtual]`

standard destructor

Definition at line 603 of file `vdbl_view.h`.

7.27.4 Member Function Documentation

7.27.4.1 `virtual void* _VDBL_view::copy_col_data (void * _d) const [protected, virtual]`

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](#).

Reimplemented in [_VDBL_hierarchicalview](#).

7.27.4.2 `virtual void* _VDBL_view::copy_def_data (void * _d) const [protected, virtual]`

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](#).

Reimplemented in [_VDBL_hierarchicalview](#).

7.27.4.3 `virtual void* _VDBL_view::copy_row_data (void * _d) const [inline, protected, virtual]`

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](#).

Definition at line 193 of file `vdbl_view.h`.

7.27.4.4 `virtual void void* _VDBL_view::next_col (const _VDBL_tableid & _t, const _VDBL_rowid & _r, const _VDBL_colid & _c, void * _d) const [protected]`

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`.

Reimplemented in `_VDBL_hierarchicalview`.

7.27.4.5 `virtual triple<_VDBL_tableid,_VDBL_colid,void*> _VDBL_view::next_def_col (const _VDBL_tableid & t, const _VDBL_colid & c, void * d) const` [protected, virtual]

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`.

Reimplemented in `_VDBL_hierarchicalview`.

7.27.4.6 `virtual void void* _VDBL_view::next_row (const _VDBL_tableid & t, const _VDBL_rowid & c, void * d) const` [protected]

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`.

Reimplemented in `_VDBL_hierarchicalview`.

7.27.4.7 `virtual void void void* _VDBL_view::prev_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [inline, protected]

This function destroys the additional data needed by a `_col_iterator`

Reimplemented in `_VDBL_hierarchicalview`.

Definition at line 147 of file `vdbl_view.h`.

7.27.4.8 `virtual triple<_VDBL_tableid,_VDBL_colid,void*> void* _VDBL_view::prev_def_col (const _VDBL_tableid & t, const _VDBL_colid & c, void * d) const` [inline, protected]

This function destroys the additional data needed by a `_default_iterator`

Reimplemented in `_VDBL_hierarchicalview`.

Definition at line 115 of file `vdbl_view.h`.

7.27.4.9 `virtual void void void* _VDBL_view::prev_row (const _VDBL_tableid & t, const _VDBL_rowid & c, void * d) const` [inline, protected]

This function destroys the additional data needed by a `_row_iterator`

Reimplemented in `_VDBL_hierarchicalview`.

Definition at line 179 of file `vdbl_view.h`.

7.27.4.10 `virtual const std::type_info& _VDBL_view::get_colinfo (const std::string & C_n, triple<bool, _VDBL_colid, _VDBL_colflags > & r) const` [inline, virtual]

return the type of this `view`

Reimplemented in `_VDBL_hierarchicalview`.

Definition at line 612 of file `vdbl_view.h`.

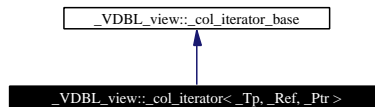
The documentation for this class was generated from the following file:

- [vdbl_view.h](#)

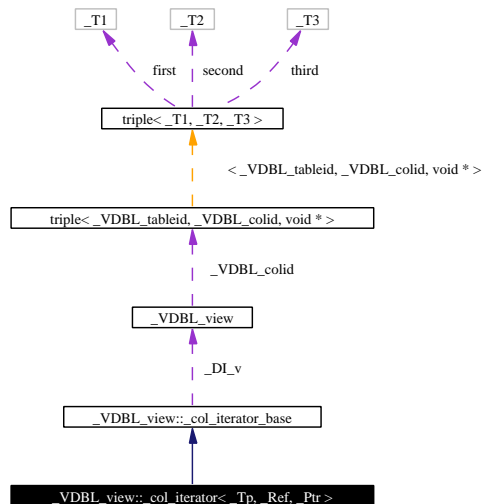
7.28 `_VDBL_view::col_iterator` Struct Template Reference

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

Inheritance diagram for `_VDBL_view::col_iterator`:



Collaboration diagram for `_VDBL_view::col_iterator`:



7.28.1 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 410 of file `vdbl_view.h`.

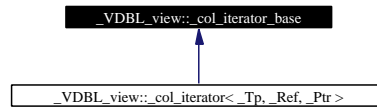
The documentation for this struct was generated from the following file:

- [vdbl_view.h](#)

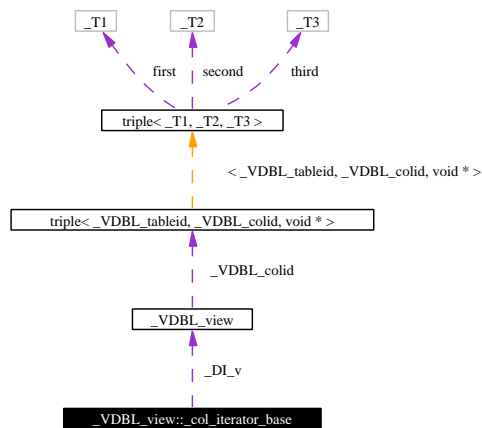
7.29 `_VDBL_view::col_iterator_base` Class Reference

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

Inheritance diagram for `_VDBL_view::col_iterator_base`:



Collaboration diagram for `_VDBL_view::col_iterator_base`:



7.29.1 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 331 of file `vdbl_view.h`.

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

- [vdbl_view.h](#)

7.30 `_VDBL_view::default_iterator` Struct Template Reference

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

7.30.1 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 274 of file `vdbl_view.h`.

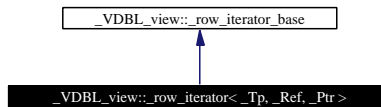
The documentation for this struct was generated from the following file:

- [vdbl_view.h](#)

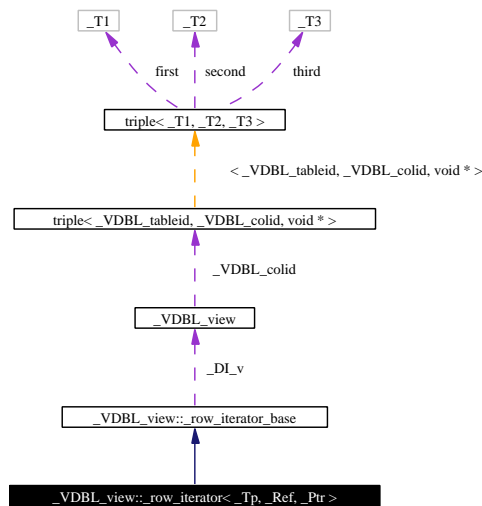
7.31 `_VDBL_view::row_iterator` Struct Template Reference

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

Inheritance diagram for `_VDBL_view::row_iterator`:



Collaboration diagram for `_VDBL_view::row_iterator`:



7.31.1 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 541 of file `vdbl_view.h`.

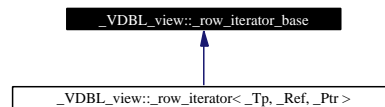
The documentation for this struct was generated from the following file:

- [vdbl_view.h](#)

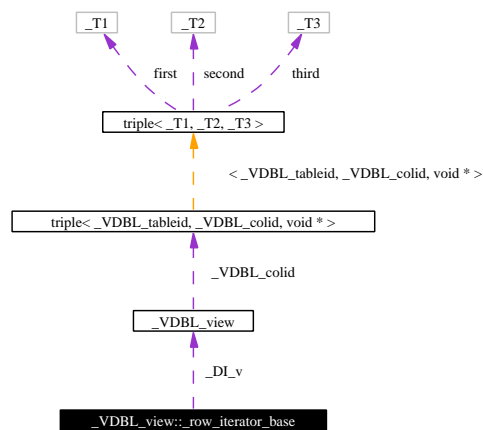
7.32 `_VDBL_view::row_iterator_base` Class Reference

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

Inheritance diagram for `_VDBL_view::_row_iterator_base`:



Collaboration diagram for `_VDBL_view::_row_iterator_base`:



7.32.1 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 467 of file `vdbl_view.h`.

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

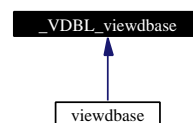
- [vdbl_view.h](#)

7.33 _VDBL_viewbase Class Reference

a [view](#) to a complete [database](#).

```
#include <vdbl_viewbase.h>
```

Inheritance diagram for `_VDBL_viewbase`:



Public Methods

- `_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` (const `_VDBL_database` &_db, const `_VDBL_userid` &uid, const `_VDBL_context` &_c)
- `template<template< class _C, class _A > class _SqCtr, class _AI> _VDBL_viewdbase` (const `_VDBL_database` &_db, const `_VDBL_userid` &uid, const `_VDBL_context` &_c, const `_SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _AI > &_an`)
- virtual `~_VDBL_viewdbase` ()

7.33.1 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 50 of file `vdbl_viewdbase.h`.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 `_VDBL_viewdbase::_VDBL_viewdbase` (const `_VDBL_database` & `_db`, const `_VDBL_userid` & `uid`, const `_VDBL_context` & `_c`) [inline]

constructor which builds a `view` to the `database` from

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

Definition at line 117 of file `vdbl_viewdbase.h`.

7.33.2.2 `template<template< class _C, class _A > class _SqCtr, class _AI> _VDBL_viewdbase::_VDBL_viewdbase` (const `_VDBL_database` & `_db`, const `_VDBL_userid` & `uid`, const `_VDBL_context` & `_c`, const `_SqCtr< std::pair< _VDBL_tableid, _VDBL_rowid >, _AI > &_an`) [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 138 of file `vdbl_viewdbase.h`.

7.33.2.3 `virtual _VDBL_viewdbase::~~_VDBL_viewdbase` () [inline, virtual]

standard destructor

Definition at line 155 of file `vdbl_viewdbase.h`.

7.33.3 Member Function Documentation

7.33.3.1 `_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 65 of file `vdbl_viewdbase.h`.

7.33.3.2 `_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 109 of file `vdbl_viewdbase.h`.

7.33.3.3 `_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 97 of file `vdbl_viewdbase.h`.

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

check whether the `view` `_C_i` exists

Definition at line 91 of file `vdbl_viewdbase.h`.

7.33.3.5 `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 79 of file `vdbl_viewdbase.h`.

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

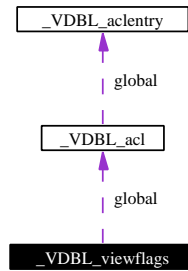
- [vdbl_viewdbase.h](#)

7.34 `_VDBL_viewflags` Class Reference

flags for one `view`.

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

Collaboration diagram for `_VDBL_viewflags`:



Public Methods

- `_VDBL_viewflags ()`
- `_VDBL_viewflags (const _VDBL_acl & _gac)`
- `_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`

7.34.1 Detailed Description

This class describes the additional information for a [view](#) within a [database](#), including access control
 Definition at line 286 of file `vdbl_database.h`.

7.34.2 Constructor & Destructor Documentation

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

standard constructor

Definition at line 302 of file `vdbl_database.h`.

7.34.2.2 `_VDBL_viewflags::_VDBL_viewflags (const _VDBL_acl & _gac) [inline]`

constructor which initializes the global ACL entry

Definition at line 307 of file `vdbl_database.h`.

7.34.2.3 `_VDBL_viewflags::_VDBL_viewflags (const _VDBL_viewflags & __v) [inline]`

copy constructor

Definition at line 312 of file `vdbl_database.h`.

7.34.2.4 `virtual _VDBL_viewflags::~~_VDBL_viewflags ()` [inline, virtual]

standard destructor

Definition at line 317 of file `vdbl_database.h`.

7.34.3 Member Function Documentation**7.34.3.1** `_VDBL_viewflags& _VDBL_viewflags::operator= (const _VDBL_viewflags & _v)` [inline]

assignment operator

Definition at line 322 of file `vdbl_database.h`.

7.34.4 Member Data Documentation**7.34.4.1** `std::map<_VDBL_userid,_VDBL_acl> _VDBL_viewflags::ACLs`

this is an access control list for every single user

Definition at line 296 of file `vdbl_database.h`.

7.34.4.2 `_VDBL_acl _VDBL_viewflags::global`

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

Definition at line 292 of file `vdbl_database.h`.

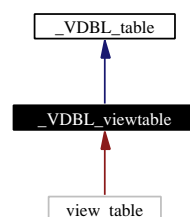
The documentation for this class was generated from the following file:

- [vdbl_database.h](#)

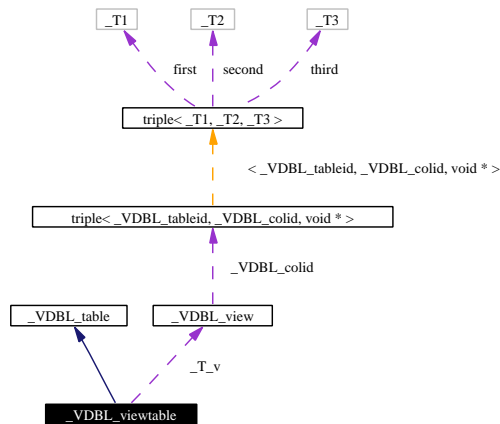
7.35 `_VDBL_viewtable` Class Reference

```
#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 Methods

- virtual `std::pair< std::string, _VDBL_colid > _next_col (const std::pair< std::string, _VDBL_colid > &_ci) const` `VDBL_PURE_VIRTUAL` virtual `std`
- virtual `_VDBL_rowid _next_row (const _VDBL_rowid &_ci) const` `VDBL_PURE_VIRTUAL` virtual `_VDBL_rowid _prev_row(const _VDBL_rowid &_ci) const` `VDBL_PURE_VIRTUAL` virtual `row_` `const _iterator row_begin() const` `VDBL_PURE_VIRTUAL` virtual `row_` `const _iterator row_end() const` `VDBL_PURE_VIRTUAL` public
- virtual `bool add_col (const std::string &_C_n, const _VDBL_col &_c, const _VDBL_colflags &_f)` `VDBL_PURE_VIRTUAL` virtual `bool modify_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 Methods

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

Friends

- class `_VDBL_view`

7.35.1 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 46 of file `vdbl_vtable.h`.

7.35.2 Member Typedef Documentation

7.35.2.1 `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 62 of file `vdbl_vtable.h`.

7.35.2.2 `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 82 of file `vdbl_table.h`.

7.35.2.3 `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 213 of file `vdbl_table.h`.

7.35.2.4 `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 320 of file `vdbl_table.h`.

7.35.3 Member Function Documentation

7.35.3.1 `virtual std::pair<std::string, _VDBL_colid> _VDBL_table::next_col (const std::pair<std::string, _VDBL_colid> & ci) const` [inline, virtual, inherited]

This 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 220 of file `vdbl_table.h`.

7.35.3.2 `virtual _VDBL_rowid _VDBL_table::next_row (const _VDBL_rowid & ci) const` [inline, virtual, inherited]

standard constructor

Reimplemented in [_VDBL_standardtable](#).

Definition at line 327 of file [vdbl_table.h](#).

7.35.3.3 `virtual bool _VDBL_table::add_col (const std::string & _C_n, const _VDBL_col & _c, const _VDBL_colflags & _f) const` [inline, virtual, inherited]

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 376 of file [vdbl_table.h](#).

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

generate new unique id's for rows and columns

Definition at line 98 of file [vdbl_table.h](#).

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

generate new unique id's for rows and columns

Definition at line 99 of file [vdbl_table.h](#).

7.35.3.6 `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 457 of file [vdbl_table.h](#).

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

increment the `last_change` counter.

Definition at line 105 of file [vdbl_table.h](#).

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

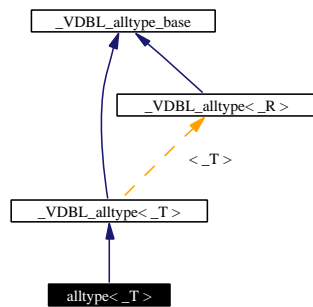
- [vdbl_vtable.h](#)

7.36 alltype Class Template Reference

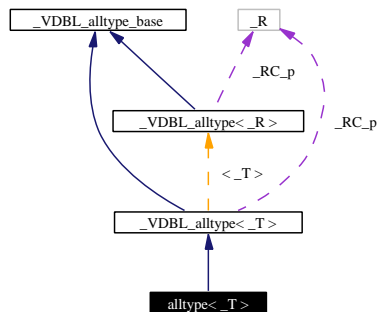
The templated alltype class.

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

Inheritance diagram for alltype:



Collaboration diagram for alltype:



Public Types

- typedef `_Base::cont_type` `cont_type`
the type of the internally stored data.

7.36.1 Detailed Description

```
template<class _T> class alltype< _T >
```

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

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

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

Definition at line 608 of file `vdbl_alltype.h`.

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

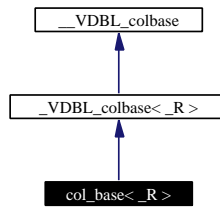
- [vdbl_alltype.h](#)

7.37 col_base Class Template Reference

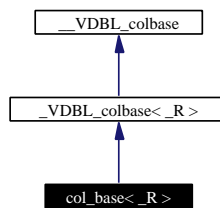
column base class.

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

Inheritance diagram for col_base:



Collaboration diagram for col_base:



Public Types

- typedef `_R` `return_type`
return_type is the type of object stored.

Public Methods

- virtual `_Self * new_copy ()` const
- virtual void `setcontext (const context *_c, const _VDBL_row *_r)` `VDBL_PURE_VIRTUAL` virtual void `get(return_type &c)` const `VDBL_PURE_VIRTUAL` virtual void `def(return_type &d)` const `VDBL_PURE_VIRTUAL` virtual void `get_ptr(return_type const *&c)` const `VDBL_PURE_VIRTUAL` virtual 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

7.37.1 Detailed Description

`template<class _R> class col_base< _R >`

this is the external name of the column base class

Definition at line 585 of file `vdbl_col.h`.

7.37.2 Member Function Documentation

7.37.2.1 `template<class R> virtual void _VDBL_colbase< R >::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 209 of file `vdbl_col.h`.

7.37.2.2 `template<class R> virtual void _VDBL_colbase< R >::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 deleted by the user to avoid memory leaks.

Definition at line 187 of file `vdbl_col.h`.

7.37.2.3 `template<class R> virtual void _VDBL_colbase< R >::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 196 of file `vdbl_col.h`.

7.37.2.4 `template<class R> virtual Self* _VDBL_colbase< R >::new_copy () const` [inline, virtual, inherited]

`new_copy` is the clone operation for copy-constructor overloading.

Reimplemented from `__VDBL_colbase`.

Reimplemented in `_VDBL_stdcol`.

Definition at line 151 of file `vdbl_col.h`.

7.37.2.5 `template<class R> virtual const std::type_info& _VDBL_colbase< R >::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 220 of file `vdbl_col.h`.

7.37.2.6 `template<class R> virtual void _VDBL_colbase< R >::setcontext (const context * c, const _VDBL_row * r) const` [inline, virtual, inherited]

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 156 of file `vdbl_col.h`.

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

- [vdbl_col.h](#)

7.38 col_spec Class Reference

column specification.

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

Public Methods

- [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)

7.38.1 Detailed Description

This class is used to specify columns of a [table](#).

Definition at line 1069 of file vdbl_table.h.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 [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 1080 of file vdbl_table.h.

7.38.2.2 [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 1083 of file vdbl_table.h.

7.38.2.3 [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 1093 of file vdbl_table.h.

7.38.2.4 [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 1097 of file vdbl_table.h.

7.38.2.5 `col_spec::col_spec (const col_spec & _c) [inline]`

copy constructor

Definition at line 1104 of file `vdbl_table.h`.**7.38.2.6** `virtual col_spec::~~col_spec () [inline, virtual]`

standard destructor

Definition at line 1109 of file `vdbl_table.h`.

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

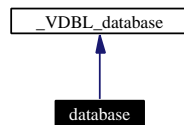
- [vdbl_table.h](#)

7.39 database Class Reference

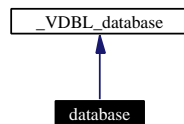
the database class.

`#include <vdbl_database.h>`

Inheritance diagram for database:



Collaboration diagram for database:

**Public Methods**

- `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 char *_C_i, const _VDBL_userid &_C_u)`
- `bool has_table (const char *_C_i, const _VDBL_userid &_C_u) const`
- `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 char *_C_i, const `_VDBL_userid` &_C_u)
- bool `has_view` (const char *_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 `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
- 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

Protected Methods

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

Friends

- class `_VDBL_viewbase`

7.39.1 Detailed Description

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

Definition at line 796 of file `vdbl_database.h`.

7.39.2 Member Function Documentation

7.39.2.1 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 824 of file `vdbl_database.h`.

7.39.2.2 `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 814 of file `vdbl_database.h`.

7.39.2.3 `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, inherited]`

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 640 of file `vdbl_database.h`.

7.39.2.4 `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 877 of file `vdbl_database.h`.

7.39.2.5 `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 868 of file `vdbl_database.h`.

7.39.2.6 `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 859 of file `vdbl_database.h`.

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

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

Definition at line 508 of file `vdbl_database.h`.

7.39.2.8 `bool _VDBL_database::drop_table (const _VDBL_tableid & _Ci, const _VDBL_userid & _Cu)` [inline, inherited]

delete a [table](#), whose name is provided. return `true`, if deleting the [table](#) has worked.

Definition at line 495 of file `vdbl_database.h`.

7.39.2.9 `bool _VDBL_database::drop_table (const _D_tables::iterator & _t, const _D_table_names::iterator & _tn, const _VDBL_userid & _Cu)` [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 470 of file `vdbl_database.h`.

7.39.2.10 `bool database::drop_table (const char * _Ci, const _VDBL_userid & _Cu)` [inline]

delete a [table](#), whose name is provided. return `true`, if deleting the [table](#) has worked.

Definition at line 847 of file `vdbl_database.h`.

7.39.2.11 `bool _VDBL_database::drop_view (const std::string & _Ci, const _VDBL_userid & _Cu)` [inline, inherited]

delete a [view](#), whose name is provided. return `true`, if deleting the [table](#) has worked.

Definition at line 707 of file `vdbl_database.h`.

7.39.2.12 `bool _VDBL_database::drop_view (const _VDBL_viewid & _Ci, const _VDBL_userid & _Cu)` [inline, inherited]

delete a [view](#), whose id is provided. return `true`, if deleting the [table](#) has worked.

Definition at line 694 of file `vdbl_database.h`.

7.39.2.13 `bool _VDBL_database::drop_view (const _D_views::iterator & _v, const _D_view_names::iterator & _vn, const _VDBL_userid & _Cu)` [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 669 of file `vdbl_database.h`.

7.39.2.14 `bool database::drop_view (const char * _Ci, const _VDBL_userid & _Cu)` [inline]

delete a [view](#), whose name is provided. return `true`, if deleting the [table](#) has worked.

Definition at line 888 of file `vdbl_database.h`.

7.39.2.15 `_VDBL_table* _VDBL_database::get_table (const _VDBL_tableid & _Ci, const _VDBL_userid & _Cu) const` [inline, inherited]

return a pointer to the [table](#) with id `_Ci`.

Definition at line 536 of file `vdbl_database.h`.

7.39.2.16 `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 923 of file `vdbl_database.h`.

7.39.2.17 `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 918 of file `vdbl_database.h`.

7.39.2.18 `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 913 of file `vdbl_database.h`.

7.39.2.19 `_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 447 of file `vdbl_database.h`.

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

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

Definition at line 391 of file `vdbl_database.h`.

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

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

Definition at line 392 of file `vdbl_database.h`.

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

return a pointer to the `view` with name `_C.i`.

Definition at line 746 of file `vdbl_database.h`.

7.39.2.23 `_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 734 of file `vdbl_database.h`.

7.39.2.24 `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 898 of file `vdbl_database.h`.

7.39.2.25 `_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 626 of file `vdbl_database.h`.

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

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

Definition at line 393 of file `vdbl_database.h`.

7.39.2.27 `bool _VDBL_database::has_table (const std::string & _C.i, const _VDBL_userid & _C.u) const` [inline, inherited]

check whether the [table](#) *_C.i* exists

Definition at line 529 of file `vdbl_database.h`.

7.39.2.28 `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 517 of file `vdbl_database.h`.

7.39.2.29 `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 852 of file `vdbl_database.h`.

7.39.2.30 `bool _VDBL_database::has_view (const std::string & _C.i, const _VDBL_userid & _C.u) const` [inline, inherited]

check whether the [view](#) *_C.i* exists

Definition at line 728 of file `vdbl_database.h`.

7.39.2.31 `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 716 of file `vdbl_database.h`.

7.39.2.32 `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 893 of file `vdbl_database.h`.

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

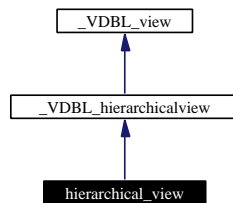
- [vdbl_database.h](#)

7.40 hierarchical_view Class Reference

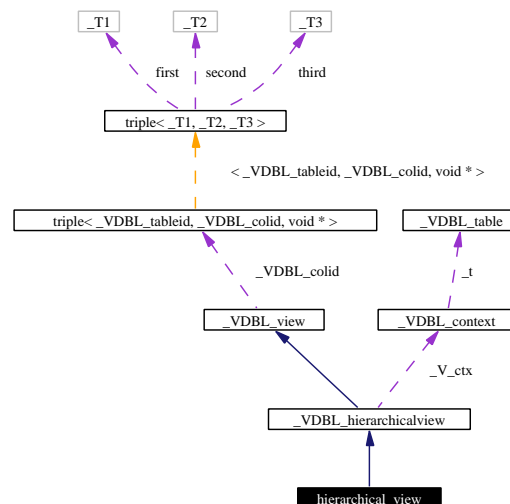
hierarchical [view](#) class onto a stack of tables.

```
#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 Methods

- `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 `_R`> bool `get` (const `tableid` &_ti, const `rowid` &_ri, const `colid` &_ci, `_R` &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` ()
- const std::type_info & `get_colinfo` (const std::string &_C_n, `triple`< bool, `_VDBL_colid`, `_VDBL_colflags` > &r) const
- bool `remove` (std::pair< `_VDBL_tableid`, `_VDBL_rowid` > _r)
- template<class `_R`> bool `get` (const std::pair< `_VDBL_tableid`, `_VDBL_rowid` > &_ri, const `_VDBL_colid` &_ci, `_R` &r) const
- template<class `_R`> bool `get` (const `rowid` &_ri, const std::string &_c, `_R` &r) const
- template<class `_R`> bool `get` (const `rowid` &_ri, const char *_c, `_R` &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 Methods

- `triple`< `_VDBL_tableid`, `_VDBL_colid`, void * > `_next_def_col` (const `_VDBL_tableid` &t, const `_VDBL_colid` &c, void *_d) const
- `triple`< `_VDBL_tableid`, `_VDBL_colid`, void * > `_prev_def_col` (const `_VDBL_tableid` &t, const `_VDBL_colid` &c, void *_d) const
- void * `_copy_def_data` (void *_d) const
- `triple`< `_VDBL_tableid`, `_VDBL_colid`, void * > `_next_col` (const `_VDBL_tableid` &t, const `_VDBL_rowid` &r, const `_VDBL_colid` &c, void *_d) const
- `triple`< `_VDBL_tableid`, `_VDBL_colid`, void * > `_prev_col` (const `_VDBL_tableid` &t, const `_VDBL_rowid` &r, const `_VDBL_colid` &c, void *_d) const
- void * `_copy_col_data` (void *_d) const
- `triple`< `_VDBL_tableid`, `_VDBL_rowid`, void * > `_next_row` (const `_VDBL_tableid` &t, const `_VDBL_rowid` &r, void *_d) const
- `triple`< `_VDBL_tableid`, `_VDBL_rowid`, void * > `_prev_row` (const `_VDBL_tableid` &t, const `_VDBL_rowid` &r, 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`

7.40.1 Detailed Description

This is the hierarchical [view](#). It is an in-memory [view](#) onto a stack of VDBL tables.

In a hierarchical [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 688 of file `vdbl_hrview.h`.

7.40.2 Member Typedef Documentation

7.40.2.1 `typedef std::pair<std::string, VDBL_col> VDBL_view::T_colspec` [inherited]

This is the description of one column

Definition at line 84 of file `vdbl_view.h`.

7.40.2.2 `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 460 of file `vdbl_view.h`.

7.40.2.3 `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 324 of file `vdbl_view.h`.

7.40.2.4 `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 589 of file `vdbl_view.h`.

7.40.3 Constructor & Destructor Documentation

7.40.3.1 `hierarchical_view::hierarchical_view (const VDBL_tableid & _ti, VDBL_table * _t, const VDBL_context & _c, V_enum _e = V_window)` [inline]

standard constructor which initializes the [table](#) and the [tableid](#) of the master [table](#), the evaluation context, and the [view](#) type.

Definition at line 700 of file `vdbl_hrview.h`.

7.40.3.2 `hierarchical_view::hierarchical_view (const hierarchical_view & v)` [inline]

copy constructor

Definition at line 707 of file `vdbl_hrview.h`.

7.40.4 Member Function Documentation**7.40.4.1** `void* _VDBL_hierarchicalview::_copy_col_data (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 `row` for a `_row_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 289 of file `vdbl_hrview.h`.

7.40.4.2 `void* _VDBL_hierarchicalview::_copy_def_data (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 column for a `_col_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 232 of file `vdbl_hrview.h`.

7.40.4.3 `void* _VDBL_hierarchicalview::_copy_row_data (void * d) const` [inline, protected, virtual, inherited]

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 326 of file `vdbl_hrview.h`.

7.40.4.4 `triple<_VDBL_tableid, _VDBL_colid, void*> _VDBL_hierarchicalview::_next_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [inline, protected, inherited]

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`.

Reimplemented from `_VDBL_view`.

Definition at line 237 of file `vdbl_hrview.h`.

7.40.4.5 `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 previous default of a column a `_default_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 180 of file `vdbl_hrview.h`.

7.40.4.6 `triple<_VDBL_tableid, _VDBL_rowid, void*> _VDBL_hierarchicalview::next_row (const _VDBL_tableid & t, const _VDBL_rowid & r, void * d) const` [inline, protected, inherited]

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`.

Reimplemented from `_VDBL_view`.

Definition at line 294 of file `vdbl_hrview.h`.

7.40.4.7 `triple<_VDBL_tableid, _VDBL_colid, void*> _VDBL_hierarchicalview::prev_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [inline, protected, inherited]

This function destroys the additional data needed by a `_col_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 259 of file `vdbl_hrview.h`.

7.40.4.8 `triple<_VDBL_tableid, _VDBL_colid, void*> _VDBL_hierarchicalview::prev_def_col (const _VDBL_tableid & t, const _VDBL_colid & c, void * d) const` [inline, protected, inherited]

This function destroys the additional data needed by a `_default_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 202 of file `vdbl_hrview.h`.

7.40.4.9 `triple<_VDBL_tableid, _VDBL_rowid, void*> _VDBL_hierarchicalview::prev_row (const _VDBL_tableid & t, const _VDBL_rowid & r, void * d) const` [inline, protected, inherited]

This function destroys the additional data needed by a `_row_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 308 of file `vdbl_hrview.h`.

7.40.4.10 `template<class _R> bool _VDBL_hierarchicalview::get (const std::pair< _VDBL_tableid, _VDBL_rowid > & ri, const _VDBL_colid & ci, _R & 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 `_R`.

Definition at line 601 of file `vdbl_hrview.h`.

7.40.4.11 `template<class _R> bool hierarchical_view::get (const rowid & ri, const char * c, _R & r) const` [inline]

get the data from column `c` in `row ri`. The data stored in the column must be of type `_R`.

Definition at line 728 of file `vdbl_hrview.h`.

7.40.4.12 `template<class _R> bool hierarchical_view::get (const rowid & ri, const std::string & c, _R & r) const` [inline]

get the data from column `_c` in row `_ri`. The data stored in the column must be of type `_R`.

Definition at line 723 of file `vdbl_hrview.h`.

7.40.4.13 `template<class _R> bool hierarchical_view::get (const tableid & _ti, const rowid & _ri, const colid & _ci, _R & 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 `_R`.

Definition at line 714 of file `vdbl_hrview.h`.

7.40.4.14 `const std::type_info& _VDBL_hierarchicalview::get_colinfo (const std::string & _Cn, triple< bool, _VDBL_colid, _VDBL_colflags > & r) const` [inline, virtual, inherited]

return the type of this view

Reimplemented from `_VDBL_view`.

Definition at line 487 of file `vdbl_hrview.h`.

7.40.4.15 `_VDBL_tableid _VDBL_hierarchicalview::pop_table ()` [inline, inherited]

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

Definition at line 458 of file `vdbl_hrview.h`.

7.40.4.16 `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 433 of file `vdbl_hrview.h`.

7.40.4.17 `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 408 of file `vdbl_hrview.h`.

7.40.4.18 `bool _VDBL_hierarchicalview::remove (std::pair< _VDBL_tableid, _VDBL_rowid > r)` [inline, inherited]

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

Definition at line 519 of file `vdbl_hrview.h`.

7.40.5 Member Data Documentation

7.40.5.1 `_V_cols _VDBL_hierarchicalview::_V_c` [protected, inherited]

This contains all columns of the view

Definition at line 89 of file `vdbl_hrview.h`.

7.40.5.2 `_V_colxref` `_VDBL_hierarchicalview::_V_cx` [protected, inherited]

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

Definition at line 93 of file `vdbl_hrview.h`.

7.40.5.3 `_V_rows` `_VDBL_hierarchicalview::_V_r` [protected, inherited]

This contains all rows of the `view`

Definition at line 85 of file `vdbl_hrview.h`.

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

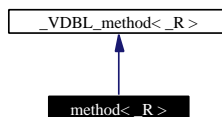
- `vdbl_hrview.h`

7.41 method Class Template Reference

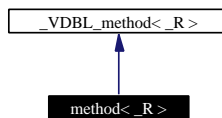
base class for methods usable in method columns.

```
#include <vdbl_method.h>
```

Inheritance diagram for method:



Collaboration diagram for method:

**Public Types**

- typedef `_R` `return_type`

Public Methods

- virtual const `return_type` & `operator()` () `VDBL_PURE_VIRTUAL` virtual const `return_type` & `def()` `VDBL_PURE_VIRTUAL` virtual void `setcontext(const context *_c`

7.41.1 Detailed Description

```
template<class _R> class method<_R>
```

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 100 of file vdbl_method.h.

7.41.2 Member Typedef Documentation

7.41.2.1 template<class *R*> typedef *R* method::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](#).

Definition at line 107 of file vdbl_method.h.

7.41.3 Member Function Documentation

7.41.3.1 template<class *R*> virtual const [return_type](#)& [_VDBL_method](#)< *R* >::operator() () const [virtual, inherited]

set the evaluation context and the evaluation [row](#).

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

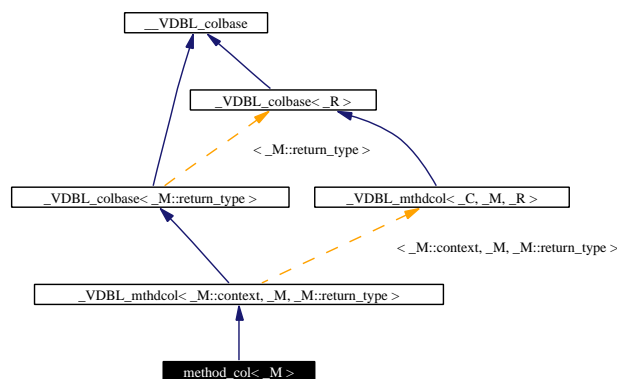
- [vdbl_method.h](#)

7.42 method_col Class Template Reference

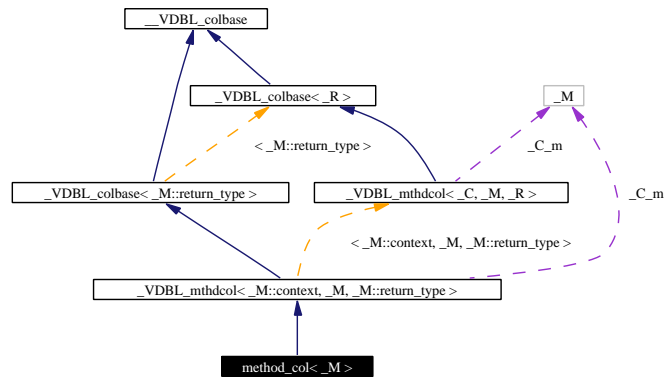
external name for computed columns.

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

Inheritance diagram for method_col:



Collaboration diagram for method_col:



Public Methods

- `_Self * new_copy () const`

7.42.1 Detailed Description

`template<class _M> class method_col< _M >`

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 631 of file `vdbl_col.h`.

7.42.2 Member Function Documentation

7.42.2.1 `_Self* _VDBL_mthdcol< _M::context, _M, _M::return_type >::new_copy () const`
`[inline, virtual, inherited]`

`new_copy` is the clone operation for copy-constructor overloading.

Reimplemented from `_VDBL_colbase< _M::return_type >`.

Definition at line 535 of file `vdbl_col.h`.

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

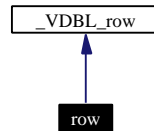
- [vdbl_col.h](#)

7.43 row Class Reference

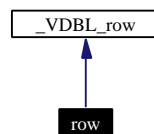
class implementing `table` rows.

```
#include <vdbl_row.h>
```

Inheritance diagram for row:



Collaboration diagram for row:



Public Methods

- [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)

7.43.1 Detailed Description

This class implements the rows of a [table](#)

Definition at line 179 of file [vdbl_row.h](#).

7.43.2 Constructor & Destructor Documentation

7.43.2.1 row::row () [inline]

standard constructor

Definition at line 190 of file [vdbl_row.h](#).

7.43.2.2 row::row (const row & _r) [inline]

copy constructor

Definition at line 194 of file [vdbl_row.h](#).

7.43.2.3 row::row (const Base & r) [inline]

copy constructor from internal class

Definition at line 198 of file vdbl_row.h.

7.43.3 Member Function Documentation**7.43.3.1 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 148 of file vdbl_row.h.

7.43.3.2 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 102 of file vdbl_row.h.

7.43.3.3 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 81 of file vdbl_row.h.

7.43.3.4 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 121 of file vdbl_row.h.

7.43.3.5 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 131 of file vdbl_row.h.

7.43.3.6 row& row::operator= (const Base & x) [inline]

assignment operator from internal class

Definition at line 211 of file vdbl_row.h.

7.43.3.7 row& row::operator= (const row & x) [inline]

assignment operator

Definition at line 203 of file vdbl_row.h.

7.43.3.8 `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 165 of file `vdbl_row.h`.

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

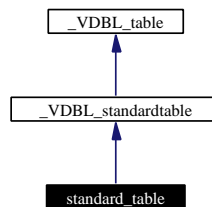
- [vdbl_row.h](#)

7.44 `standard_table` Class Reference

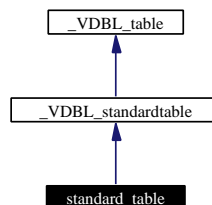
standard [table](#) of a [database](#).

```
#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 Methods

- `standard_table ()`
- `standard_table (const standard_table &...t)`

- `template<template< class __Tp, class __AllocTp > class __SequenceCtr, class Allocator1> standard_table (const __SequenceCtr< triple< std::string, _VDBL_col, _VDBL_colflags >, Allocator1 > &__cc)`
- `virtual ~standard_table ()`
- `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)`
- `const row & get_row (const rowid &_ri, bool &error) const`
- `row & get_row (const rowid &_ri, bool &error)`
- `const std::type_info & get_colinfo (const std::string &_C_n, triple< bool, _VDBL_colid, _VDBL_colflags > &_r) const`
- `_VDBL_colid get_col_id (const std::string &_C_n) const`
- `virtual bool add_col (const std::string &_C_n, const _VDBL_col &_c, const _VDBL_colflags &_f) VDBL_PURE_VIRTUAL virtual bool modify_col(const std`
- `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 Methods

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

7.44.1 Detailed Description

This class describes the standard [table](#) of a [database](#), consisting of rows and columns.

Definition at line 1118 of file `vdbl_table.h`.

7.44.2 Member Typedef Documentation

7.44.2.1 `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 561 of file `vdbl_table.h`.

7.44.2.2 `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 566 of file `vdbl_table.h`.

7.44.2.3 `typedef _Base::col_const_iterator _VDBL_standardtable::col_const_iterator` [inherited]

const iterator over all columns

Reimplemented from [_VDBL_table](#).

Definition at line 571 of file `vdbl_table.h`.

7.44.2.4 `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 320 of file `vdbl_table.h`.

7.44.3 Constructor & Destructor Documentation

7.44.3.1 `standard_table::standard_table ()` [inline]

standard constructor

Definition at line 1133 of file `vdbl_table.h`.

7.44.3.2 `standard_table::standard_table (const standard_table & t)` [inline]

copy constructor

Definition at line 1137 of file `vdbl_table.h`.

7.44.3.3 `template<template< class _Tp, class _AllocTp > class _SequenceCtr, class Allocator1> standard_table::standard_table (const _SequenceCtr< 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 1145 of file `vdbl_table.h`.

7.44.3.4 `virtual standard_table::~~standard_table()` [`inline`, `virtual`]

standard destructor

Definition at line 1151 of file `vdbl_table.h`.

7.44.4 Member Function Documentation**7.44.4.1** `std::pair<std::string, VDBL_colid> _VDBL_standardtable::next_col (const std::pair<std::string, VDBL_colid> & _ci) const` [`inline`, `virtual`, `inherited`]

This 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 978 of file `vdbl_table.h`.

7.44.4.2 `_VDBL_rowid _VDBL_standardtable::next_row (const _VDBL_rowid & _ci) const` [`inline`, `virtual`, `inherited`]

standard constructor

Reimplemented from `_VDBL_table`.

Definition at line 1013 of file `vdbl_table.h`.

7.44.4.3 `virtual bool _VDBL_table::add_col (const std::string & _C_n, const _VDBL_col & _c, const _VDBL_colflags & _f) const` [`inline`, `virtual`, `inherited`]

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 376 of file `vdbl_table.h`.

7.44.4.4 `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 1164 of file `vdbl_table.h`.

7.44.4.5 `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 1160 of file `vdbl_table.h`.

7.44.4.6 `_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 607 of file `vdbl_table.h`.

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

generate new unique id's for rows and columns

Definition at line 98 of file `vdbl_table.h`.

7.44.4.8 `colid` `standard_table::get_colid (const char * C_n) const` [inline]

return the column id of column `_C_n`

Definition at line 1262 of file `vdbl_table.h`.

7.44.4.9 `colid` `standard_table::get_colid (const std::string & C_n) const` [inline]

return the column id of column `_C_n`

Definition at line 1259 of file `vdbl_table.h`.

7.44.4.10 `const std::type_info&` `_VDBL_standardtable::get_colinfo (const std::string & C_n, triple< bool, _VDBL_colid, _VDBL_colflags > & r) const` [inline, virtual, inherited]

what was the id of the last change to the `table`

Reimplemented from `_VDBL_table`.

Definition at line 584 of file `vdbl_table.h`.

7.44.4.11 `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 1252 of file `vdbl_table.h`.

7.44.4.12 `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 1245 of file `vdbl_table.h`.

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

generate new unique id's for rows and columns

Definition at line 99 of file `vdbl_table.h`.

7.44.4.14 `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 1181 of file `vdbl_table.h`.

7.44.4.15 `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 1174 of file `vdbl_table.h`.

7.44.4.16 `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 457 of file `vdbl_table.h`.

7.44.4.17 `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 1227 of file `vdbl_table.h`.

7.44.4.18 `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 1211 of file `vdbl_table.h`.

7.44.4.19 `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 1193 of file `vdbl_table.h`.

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

increment the `last_change` counter.

Definition at line 105 of file `vdbl_table.h`.

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

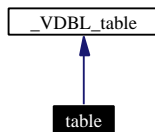
- [vdbl_table.h](#)

7.45 table Class Reference

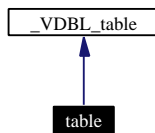
base class for tables in a [database](#).

```
#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 Methods

- virtual const std::type_info & [get_colinfo](#) (const std::string &[_C_n](#), [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_VIRTUAL](#) virtual std
- virtual [_VDBL_rowid](#) [_next_row](#) (const [_VDBL_rowid](#) &[_ci](#)) const [VDBL_PURE_VIRTUAL](#) virtual [_VDBL_rowid](#) [_prev_row](#)(const [_VDBL_rowid](#) &[_ci](#)) const [VDBL_PURE_VIRTUAL](#) virtual [row_](#)const [_iterator](#) [row_begin](#)() const [VDBL_PURE_VIRTUAL](#) virtual [row_](#)const [_iterator](#) [row_end](#)() const [VDBL_PURE_VIRTUAL](#) public
- virtual bool [add_col](#) (const std::string &[_C_n](#), const [_VDBL_col](#) &[_c](#), const [_VDBL_colflags](#) &[_f](#)) [VDBL_PURE_VIRTUAL](#) virtual bool [modify_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 Methods

- void [made_change](#) ()
- [_VDBL_colid](#) [get_colid](#) ()
- [_VDBL_rowid](#) [get_rowid](#) ()

7.45.1 Detailed Description

This is the base class for all tables in a [database](#)

Definition at line 1052 of file [vdbl_table.h](#).

7.45.2 Member Typedef Documentation

7.45.2.1 `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](#).

Definition at line 77 of file [vdbl_table.h](#).

7.45.2.2 `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 82 of file [vdbl_table.h](#).

7.45.2.3 `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 213 of file [vdbl_table.h](#).

7.45.2.4 `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 320 of file [vdbl_table.h](#).

7.45.3 Member Function Documentation

7.45.3.1 `virtual std::pair<std::string, _VDBL_colid> _VDBL_table::next_col (const std::pair<std::string, _VDBL_colid> & ci) const [inline, virtual, inherited]`

This 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 220 of file `vdbl_table.h`.

7.45.3.2 `virtual _VDBL_rowid _VDBL_table::next_row (const _VDBL_rowid & _ci) const` [inline, virtual, inherited]

standard constructor

Reimplemented in [_VDBL_standardtable](#).

Definition at line 327 of file `vdbl_table.h`.

7.45.3.3 `virtual bool _VDBL_table::add_col (const std::string & _Cn, const _VDBL_col & _c, const _VDBL_colflags & _f) const` [inline, virtual, inherited]

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 376 of file `vdbl_table.h`.

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

generate new unique id's for rows and columns

Definition at line 98 of file `vdbl_table.h`.

7.45.3.5 `virtual const std::type_info& _VDBL_table::get_colinfo (const std::string & _Cn, triple< bool, _VDBL_colid, _VDBL_colflags > & _r) const` [inline, virtual, inherited]

what was the id of the last change to the table

Reimplemented in [_VDBL_standardtable](#).

Definition at line 115 of file `vdbl_table.h`.

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

generate new unique id's for rows and columns

Definition at line 99 of file `vdbl_table.h`.

7.45.3.7 `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 457 of file `vdbl_table.h`.

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

increment the `last_change` counter.

Definition at line 105 of file vdbl_table.h.

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

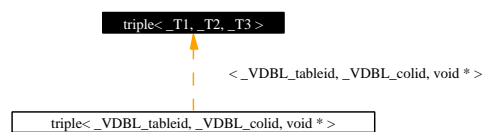
- [vdbl_table.h](#)

7.46 triple Struct Template Reference

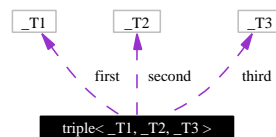
triple holds three objects of arbitrary type.

```
#include <vdbl_triple.h>
```

Inheritance diagram for triple:



Collaboration diagram for triple:



Public Types

- typedef `_T1` `first_type`
`first_type` is the first bound type.
- typedef `_T2` `second_type`
`second_type` is the second bound type.
- typedef `_T3` `third_type`
`third_type` is the second bound type.

Public Methods

- `triple` (const `_T1` &__a, const `_T2` &__b, const `_T3` &__c)
- `template<class _U1, class _U2, class _U3> triple` (const triple< `_U1`, `_U2`, `_U3` > &__t)

Public Attributes

- `_T1` `first`
`first` is a copy of the first object.

- `_T2 second`
`second` is a copy of the second object.
- `_T3 third`
`third` is a copy of the second object.

7.46.1 Detailed Description

```
template<class _T1, class _T2, class _T3> struct triple< _T1, _T2, _T3 >
```

This class is used to hold three objects of arbitrary types. It is a slight generalization of `std::pair`.

Definition at line 41 of file `vdbl_triple.h`.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 `template<class _T1, class _T2, class _T3> triple< _T1, _T2, _T3 >::triple (const _T1 & _a, const _T2 & _b, const _T3 & _c) [inline]`

Three objects may be passed to a `triple` constructor to be copied.

Definition at line 68 of file `vdbl_triple.h`.

7.46.2.2 `template<class _T1, class _T2, class _T3> template<class _U1, class _U2, class _U3> triple< _T1, _T2, _T3 >::triple (const triple< _U1, _U2, _U3 > & _t) [inline]`

There is also a templated copy constructor for the `triple` class itself.

Definition at line 75 of file `vdbl_triple.h`.

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

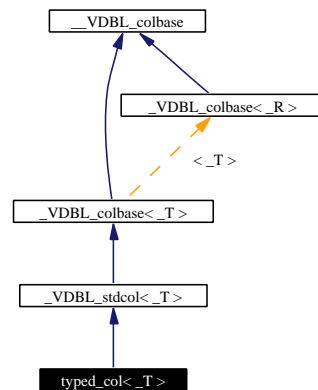
- [vdbl_triple.h](#)

7.47 `typed_col` Class Template Reference

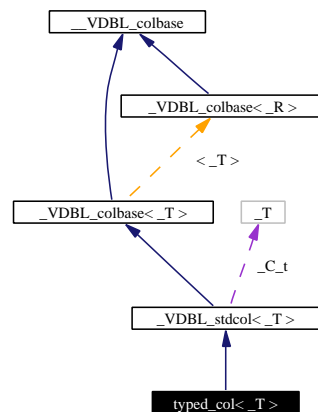
external name for constant data columns.

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

Inheritance diagram for `typed_col`:



Collaboration diagram for typed_col:



Public Methods

- `_Self * new_copy ()` const
- `void set (const _Self &p)`
- `void set (const type &_t)`
- `void setcontext (const context *_c, const _VDBL_row *_r)`
- `void def (type &d)` const
- `void set_default (const type &_t)`
- `const type & get_val ()` const

7.47.1 Detailed Description

template<class _T> class typed_col< _T >

this is the external name of the standard column for constant data all methods are implemented in class `_VDBL_stdcol<_T>`.

Definition at line 598 of file `vdbl_col.h`.

7.47.2 Member Function Documentation

7.47.2.1 `template<class _T> void _VDBL_stdcol< _T >::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 453 of file `vdbl_col.h`.

7.47.2.2 `template<class _T> const type& _VDBL_stdcol< _T >::get_val () const` [inline, inherited]

get a const reference to the column value

Definition at line 492 of file `vdbl_col.h`.

7.47.2.3 `template<class _T> _Self* _VDBL_stdcol< _T >::new_copy () const` [inline, virtual, inherited]

This function is used to overload the copy constructor.

Reimplemented from `_VDBL_colbase< _T >`.

Definition at line 432 of file `vdbl_col.h`.

7.47.2.4 `template<class _T> void _VDBL_stdcol< _T >::set (const type & _t)` [inline, inherited]

set the column value

Definition at line 480 of file `vdbl_col.h`.

7.47.2.5 `template<class _T> void _VDBL_stdcol< _T >::set (const _Self & _p)` [inline, inherited]

explicit copy operation

Definition at line 437 of file `vdbl_col.h`.

7.47.2.6 `template<class _T> void _VDBL_stdcol< _T >::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 487 of file `vdbl_col.h`.

7.47.2.7 `template<class _T> void _VDBL_stdcol< _T >::setcontext (const context * _c, const _VDBL_row * _r)` [inline, inherited]

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

Definition at line 443 of file `vdbl_col.h`.

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

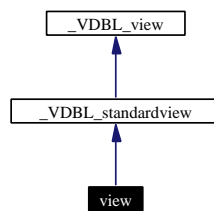
- [vdbl_col.h](#)

7.48 view Class Reference

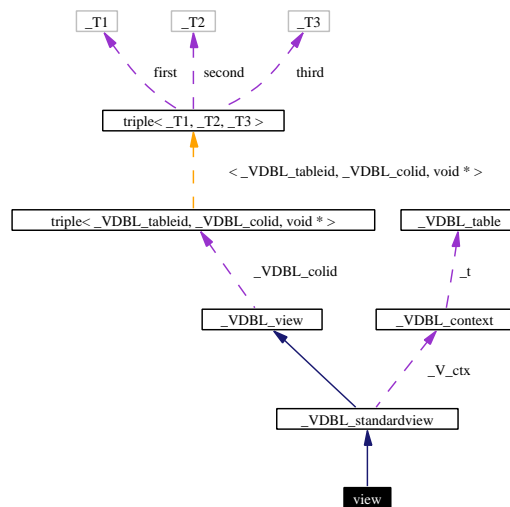
standard view class onto a single [table](#).

```
#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 Methods

- [view](#) (const [_VDBL_tableid](#) &_ti, [_VDBL_table](#) *_t, const [_VDBL_context](#) &_c, [_V_enum](#) _e=`V_-window`)
- [view](#) (const view &_v)

- `template<class _R> bool get (const tableid &_ti, const rowid &_ri, const colid &_ci, _R &r) const`
- `const col & get_col (const std::pair< tableid, rowid > &_ri, const colid &_ci, row const *&_rr, bool &error) const`
- `const std::type_info & get_colinfo (const std::string &_C_n, triple< bool, _VDBL_colid, _VDBL_colflags > &_r) const`
- `bool remove (std::pair< _VDBL_tableid, _VDBL_rowid > _r)`
- `template<class _R> bool get (const std::pair< _VDBL_tableid, _VDBL_rowid > &_ri, const _VDBL_colid &_ci, _R &r) const`
- `template<class _R> bool get (const rowid &_ri, const std::string &_c, _R &r) const`
- `template<class _R> bool get (const rowid &_ri, const char *_c, _R &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 Methods

- `triple< _VDBL_tableid, _VDBL_colid, void * > _next_def_col (const _VDBL_tableid &_t, const _VDBL_colid &_c, void *_d) const`
- `triple< _VDBL_tableid, _VDBL_colid, void * > _prev_def_col (const _VDBL_tableid &_t, const _VDBL_colid &_c, void *_d) const`
- `void * _copy_def_data (void *_d) const`
- `triple< _VDBL_tableid, _VDBL_colid, void * > _next_col (const _VDBL_tableid &_t, const _VDBL_rowid &_r, const _VDBL_colid &_c, void *_d) const`
- `triple< _VDBL_tableid, _VDBL_colid, void * > _prev_col (const _VDBL_tableid &_t, const _VDBL_rowid &_r, const _VDBL_colid &_c, void *_d) const`
- `void * _copy_col_data (void *_d) const`
- `triple< _VDBL_tableid, _VDBL_rowid, void * > _next_row (const _VDBL_tableid &_t, const _VDBL_rowid &_r, void *_d) const`
- `triple< _VDBL_tableid, _VDBL_rowid, void * > _prev_row (const _VDBL_tableid &_t, const _VDBL_rowid &_r, 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`

7.48.1 Detailed Description

This is the standard view. It is an in-memory view onto a single [VDBL table](#).

Definition at line 471 of file `vdbl_stview.h`.

7.48.2 Member Typedef Documentation

7.48.2.1 `typedef std::pair<std::string, _VDBL_col> _VDBL_view::T_colspec [inherited]`

This is the description of one column

Definition at line 84 of file `vdbl_view.h`.

7.48.2.2 `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 460 of file `vdbl_view.h`.

7.48.2.3 `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 324 of file `vdbl_view.h`.

7.48.2.4 `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 589 of file `vdbl_view.h`.

7.48.3 Constructor & Destructor Documentation

7.48.3.1 `view::view (const _VDBL_tableid & _ti, _VDBL_table * _t, const _VDBL_context & _c, _V_enum _e = V_window) [inline]`

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

Definition at line 483 of file `vdbl_stview.h`.

7.48.3.2 `view::view (const view & _v) [inline]`

copy constructor

Definition at line 490 of file `vdbl_stview.h`.

7.48.4 Member Function Documentation

7.48.4.1 `void* _VDBL_standardview::copy_col_data (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 `row` for a `_row_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 194 of file `vdbl_stview.h`.

7.48.4.2 `void* _VDBL_standardview::_copy_def_data (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 column for a `_col_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 144 of file `vdbl_stview.h`.

7.48.4.3 `void* _VDBL_standardview::_copy_row_data (void * d) const` [`inline`, `protected`, `virtual`, `inherited`]

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 230 of file `vdbl_stview.h`.

7.48.4.4 `triple<_VDBL_tableid, VDBL_colid, void*> _VDBL_standardview::_next_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [`inline`, `protected`, `inherited`]

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`.

Reimplemented from `_VDBL_view`.

Definition at line 149 of file `vdbl_stview.h`.

7.48.4.5 `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 previous default of a column a `_default_iterator`.

Reimplemented from `_VDBL_view`.

Definition at line 100 of file `vdbl_stview.h`.

7.48.4.6 `triple<_VDBL_tableid, VDBL_rowid, void*> _VDBL_standardview::_next_row (const _VDBL_tableid & t, const _VDBL_rowid & r, void * d) const` [`inline`, `protected`, `inherited`]

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`.

Reimplemented from `_VDBL_view`.

Definition at line 199 of file `vdbl_stview.h`.

7.48.4.7 `triple<_VDBL_tableid, _VDBL_colid, void*> _VDBL_standardview::prev_col (const _VDBL_tableid & t, const _VDBL_rowid & r, const _VDBL_colid & c, void * d) const` [inline, protected, inherited]

This function destroys the additional data needed by a `_col_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 169 of file `vdbl_stview.h`.

7.48.4.8 `triple<_VDBL_tableid, _VDBL_colid, void*> _VDBL_standardview::prev_def_col (const _VDBL_tableid & t, const _VDBL_colid & c, void * d) const` [inline, protected, inherited]

This function destroys the additional data needed by a `_default_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 120 of file `vdbl_stview.h`.

7.48.4.9 `triple<_VDBL_tableid, _VDBL_rowid, void*> _VDBL_standardview::prev_row (const _VDBL_tableid & t, const _VDBL_rowid & r, void * d) const` [inline, protected, inherited]

This function destroys the additional data needed by a `_row_iterator`

Reimplemented from `_VDBL_view`.

Definition at line 213 of file `vdbl_stview.h`.

7.48.4.10 `template<class _R> bool _VDBL_standardview::get (const std::pair< _VDBL_tableid, _VDBL_rowid > & ri, const _VDBL_colid & ci, _R & 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 `_R`.

Definition at line 393 of file `vdbl_stview.h`.

7.48.4.11 `template<class _R> bool view::get (const rowid & ri, const char * c, _R & r) const` [inline]

get the data from column `c` in row `ri`. The data stored in the column must be of type `_R`.

Definition at line 511 of file `vdbl_stview.h`.

7.48.4.12 `template<class _R> bool view::get (const rowid & ri, const std::string & c, _R & r) const` [inline]

get the data from column `c` in row `ri`. The data stored in the column must be of type `_R`.

Definition at line 506 of file `vdbl_stview.h`.

7.48.4.13 `template<class _R> bool view::get (const tableid & ti, const rowid & ri, const colid & ci, _R & 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 `_R`.

Definition at line 497 of file `vdbl_stview.h`.

7.48.4.14 `const col& view::get_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 `@false`, otherwise to `true`.

Definition at line 520 of file `vdbl_stview.h`.

7.48.4.15 `const std::type_info& _VDBL_standardview::get_colinfo (const std::string & _Cn, triple< bool, _VDBL_colid, _VDBL_colflags > & _r) const` [inline, virtual, inherited]

return the type of this view

Reimplemented from `_VDBL_view`.

Definition at line 301 of file `vdbl_stview.h`.

7.48.4.16 `bool _VDBL_standardview::remove (std::pair< _VDBL_tableid, _VDBL_rowid > _r)` [inline, inherited]

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

Definition at line 330 of file `vdbl_stview.h`.

7.48.5 Member Data Documentation

7.48.5.1 `_V_cols` `_VDBL_standardview::_V_c` [protected, inherited]

This contains all columns of the view

Definition at line 86 of file `vdbl_stview.h`.

7.48.5.2 `_V_rows` `_VDBL_standardview::_V_r` [protected, inherited]

This contains all rows of the view

Definition at line 82 of file `vdbl_stview.h`.

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

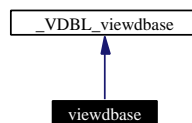
- [vdbl_stview.h](#)

7.49 viewbase Class Reference

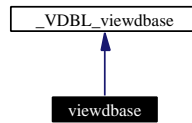
a `view` to a complete `database`.

```
#include <vdbl_viewbase.h>
```

Inheritance diagram for `viewbase`:



Collaboration diagram for viewbase:



Public Methods

- 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
- viewbase (const [database](#) &db, const [userid](#) &uid, const [context](#) &c)
- template<template< class _C, class _A > class _SqCtr, class _AI> [viewbase](#) (const [database](#) &db, const [userid](#) &uid, const [context](#) &c, const _SqCtr< std::pair< [tableid](#), [rowid](#) >, _AI > &_an)
- virtual ~viewbase ()
- [_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

7.49.1 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 170 of file `vdbl_viewbase.h`.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 `viewbase::viewbase (const database & db, const userid & uid, const context & c)` [inline]

constructor which builds a [view](#) to the [database](#) from

- db – the [database](#)
- c – the evaluation context for all views

Definition at line 200 of file `vdbl_viewbase.h`.

7.49.2.2 `template<template< class _C, class _A > class _SqCtr, class _AI> viewbase::viewbase (const database & db, const userid & uid, const context & c, const _SqCtr< std::pair< tableid, rowid >, _AI > & _an)` [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](#) The fourth argument is any sequential container of [table, row](#) pairs to which the [view](#) shall be restricted.

Definition at line 212 of file `vdbl_viewbase.h`.

7.49.2.3 virtual viewbase::~~viewbase () [inline, virtual]

standard destructor

Definition at line 219 of file `vdbl_viewbase.h`.

7.49.3 Member Function Documentation

7.49.3.1 `_VDBL_tableid` `_VDBL_viewbase::get_tableid (const std::string & _C.i) const` [inline, inherited]

return the [table](#) id (and [view](#) id) of [table](#) `_C.i`

Definition at line 65 of file `vdbl_viewbase.h`.

7.49.3.2 `_VDBL_view*` `_VDBL_viewbase::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 97 of file `vdbl_viewbase.h`.

7.49.3.3 `viewbase*` `viewbase::get_view (const std::string & _C.i) const` [inline]

this [method](#) returns a pointer to the [view](#) `_C.i`.

Reimplemented from `_VDBL_viewbase`.

Definition at line 192 of file `vdbl_viewbase.h`.

7.49.3.4 `viewbase*` `viewbase::get_view (const char * _C.i) const` [inline]

this [method](#) returns a pointer to the [view](#) `_C.i`.

Definition at line 187 of file `vdbl_viewbase.h`.

7.49.3.5 `bool` `_VDBL_viewbase::has_view (const std::string & _C.i) const` [inline, inherited]

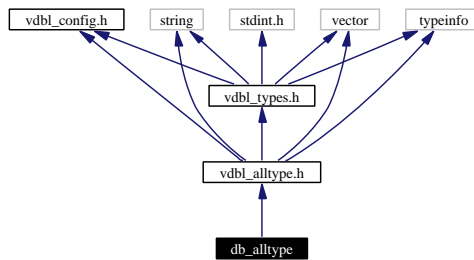
check whether the [view](#) `_C.i` exists

Definition at line 91 of file `vdbl_viewbase.h`.

7.49.3.6 `bool` `_VDBL_viewbase::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 79 of file `vdbl_viewbase.h`.



8.2.1 Detailed Description

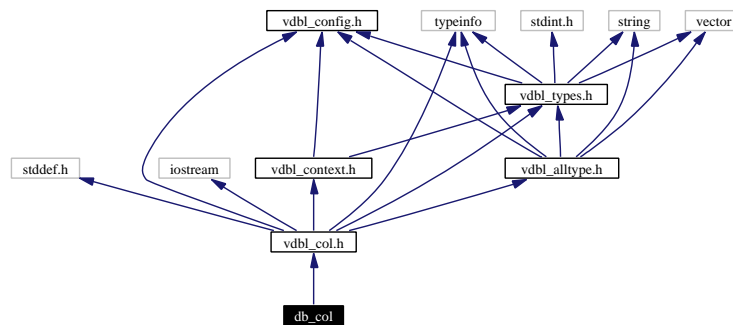
This is the external header file intended for direct use.

Definition in file [db_alltype](#).

8.3 db_col File Reference

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

Include dependency graph for db_col:



8.3.1 Detailed Description

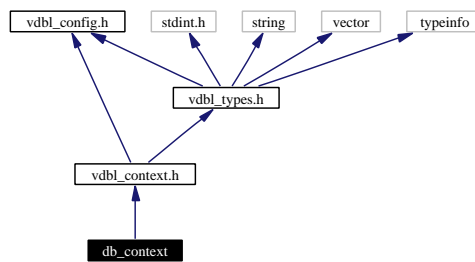
This is the external header file intended for direct use.

Definition in file [db_col](#).

8.4 db_context File Reference

```
#include <vdbl_context.h>
```

Include dependency graph for db_context:



8.4.1 Detailed Description

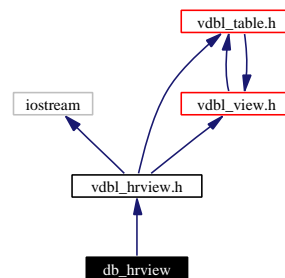
This is the external header file intended for direct use.

Definition in file [db_context](#).

8.5 db_hrview File Reference

```
#include <vdbl_hrview.h>
```

Include dependency graph for `db_hrview`:



8.5.1 Detailed Description

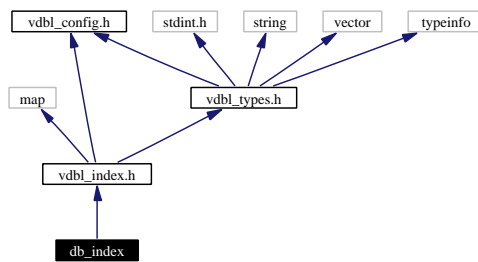
This is the external header file intended for direct use.

Definition in file [db_hrview](#).

8.6 db_index File Reference

```
#include <vdbl_index.h>
```

Include dependency graph for `db_index`:



8.6.1 Detailed Description

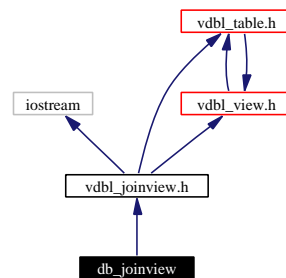
This is the external header file intended for direct use.

Definition in file [db_index](#).

8.7 db_joinview File Reference

```
#include <vdbl_joinview.h>
```

Include dependency graph for `db_joinview`:



8.7.1 Detailed Description

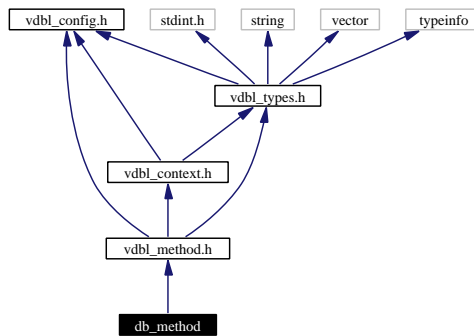
This is the external header file intended for direct use.

Definition in file [db_joinview](#).

8.8 db_method File Reference

```
#include <vdbl_method.h>
```

Include dependency graph for `db_method`:



8.8.1 Detailed Description

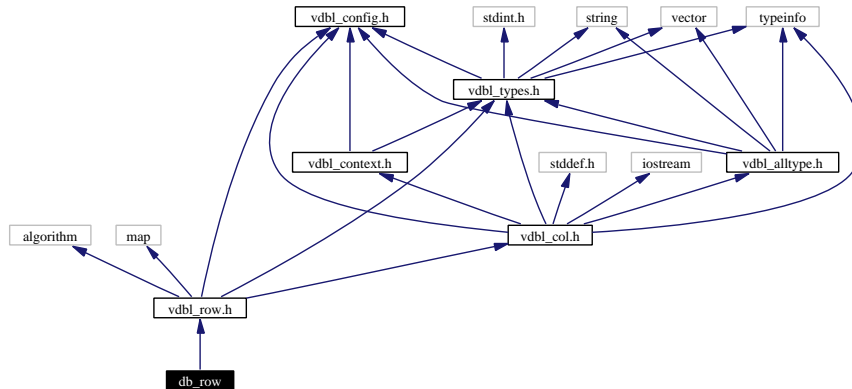
This is the external header file intended for direct use.

Definition in file [db_method](#).

8.9 db_row File Reference

```
#include <vdbl_row.h>
```

Include dependency graph for db_row:



8.9.1 Detailed Description

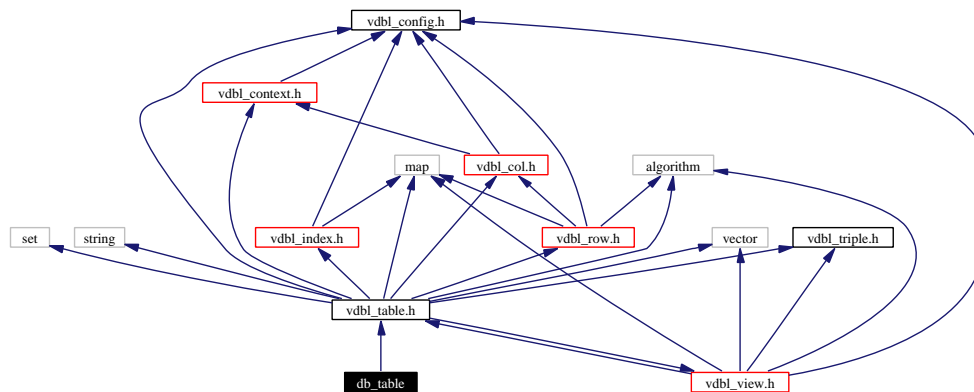
This is the external header file intended for direct use.

Definition in file [db_row](#).

8.10 db_table File Reference

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

Include dependency graph for db_table:



8.10.1 Detailed Description

This is the external header file intended for direct use.

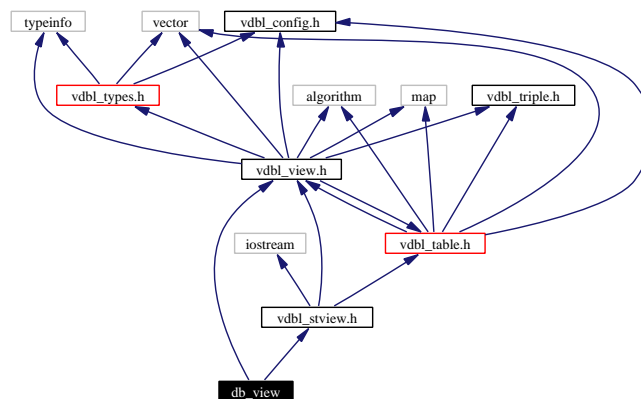
Definition in file [db_table](#).

8.11 db_view File Reference

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

```
#include <vdbl_stview.h>
```

Include dependency graph for db_view:



8.11.1 Detailed Description

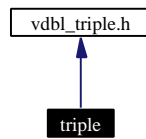
This is the external header file intended for direct use.

Definition in file [db_view](#).

8.12 triple File Reference

```
#include <vdbl_triple.h>
```

Include dependency graph for triple:



8.12.1 Detailed Description

This is the external header file intended for direct use.

Definition in file [triple](#).

8.13 vdbl_alltype.h File Reference

```
#include <vdbl_config.h>
```

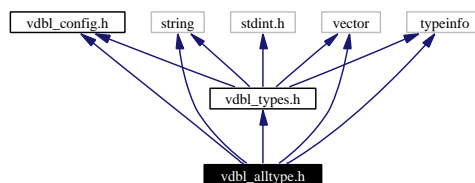
```
#include <vdbl_types.h>
```

```
#include <string>
```

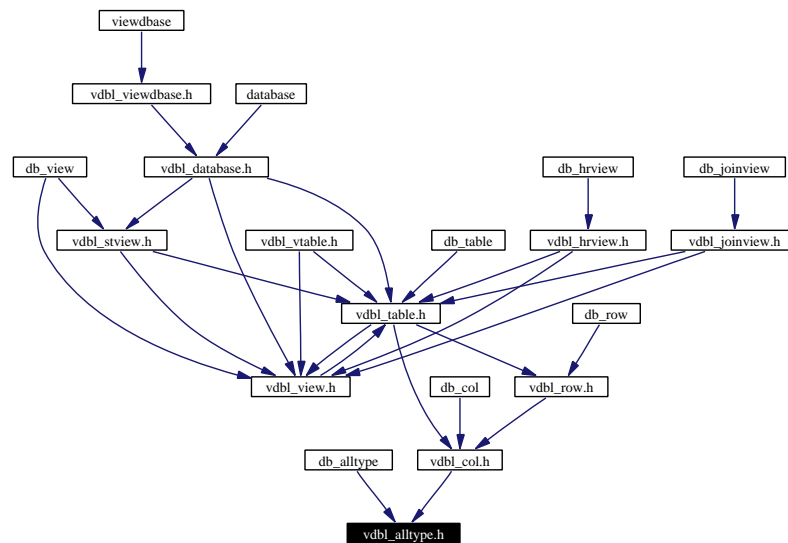
```
#include <vector>
```

```
#include <typeinfo>
```

Include dependency graph for vdbl_alltype.h:



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



Compounds

- class [_VDBL_alltype](#)
The templated class for the all_type class.
- class [_VDBL_alltype_base](#)
The base 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](#)
The templated alltype class.

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`.

8.13.1 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](#).

8.13.2 Define Documentation

8.13.2.1 `#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 323 of file `vdbl_alltype.h`.

8.13.2.2 `#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 325 of file `vdbl_alltype.h`.

8.13.2.3 `#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 324 of file `vdbl_alltype.h`.

8.13.2.4 `#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 322 of file `vdbl_alltype.h`.

8.13.2.5 `#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 327 of file `vdbl_alltype.h`.

8.13.2.6 `#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 326 of file `vdbl_alltype.h`.

8.13.2.7 #define VDBL_MIXTYPE_EMPTY -1

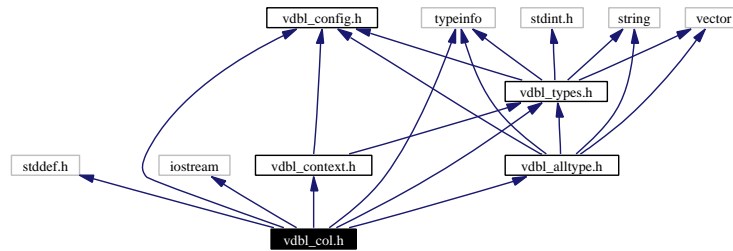
These defines internally describe which kind of vector has been allocated for a mixtype, or whether it is scalar or empty.

Definition at line 321 of file vdbl_alltype.h.

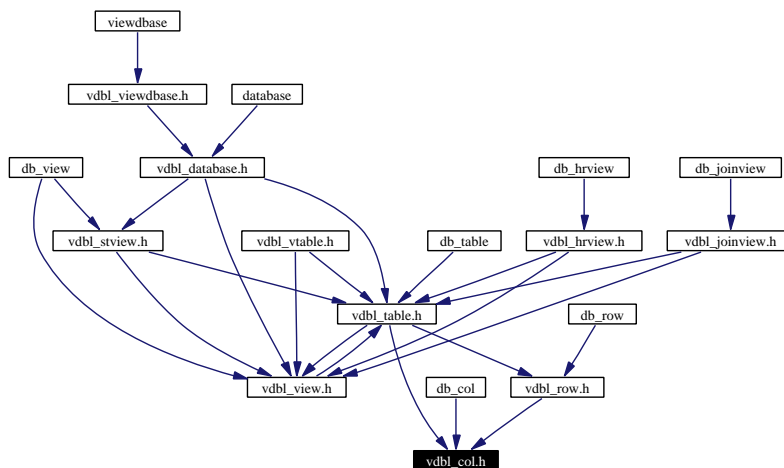
8.14 vdbl_col.h File Reference

```
#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:



Compounds

- class `__VDBL_colbase`
The base class of the internal column structure.
- class `_VDBL_col`
The generic column class (the external structure).
- class `_VDBL_colbase`
The type dependent base class of the internal column structure.
- class `_VDBL_mthdcol`
generic column class for methods.
- class `_VDBL_stdcol`
generic column class for constant values.
- class `col_base`
column base class.
- class `method_col`
external name for computed columns.
- class `typed_col`
external name for constant data 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 _C> std::ostream & print_it (std::ostream &o, const _C &t)`
- `std::ostream & operator<< (std::ostream &o, const _VDBL_col &c)`

8.14.1 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_col.h`.

8.14.2 Function Documentation

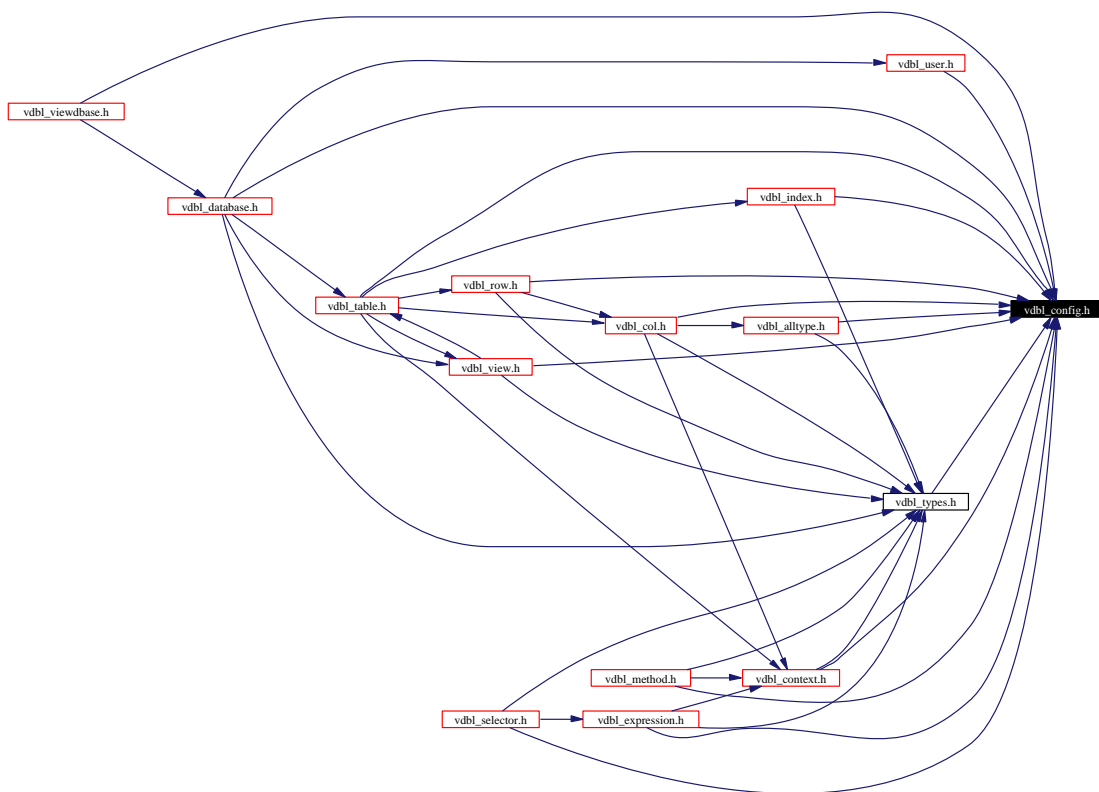
8.14.2.1 `std::ostream& operator<< (std::ostream & o, const _VDBL_col & c) [inline]`

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

Definition at line 359 of file `vdbl_col.h`.

8.15 vdbl_config.h File Reference

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



8.15.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

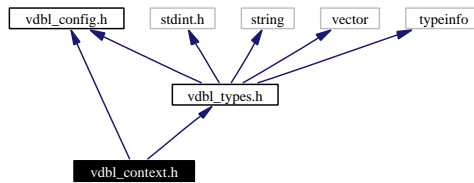
Definition in file `vdbl_config.h`.

8.16 vdbl_context.h File Reference

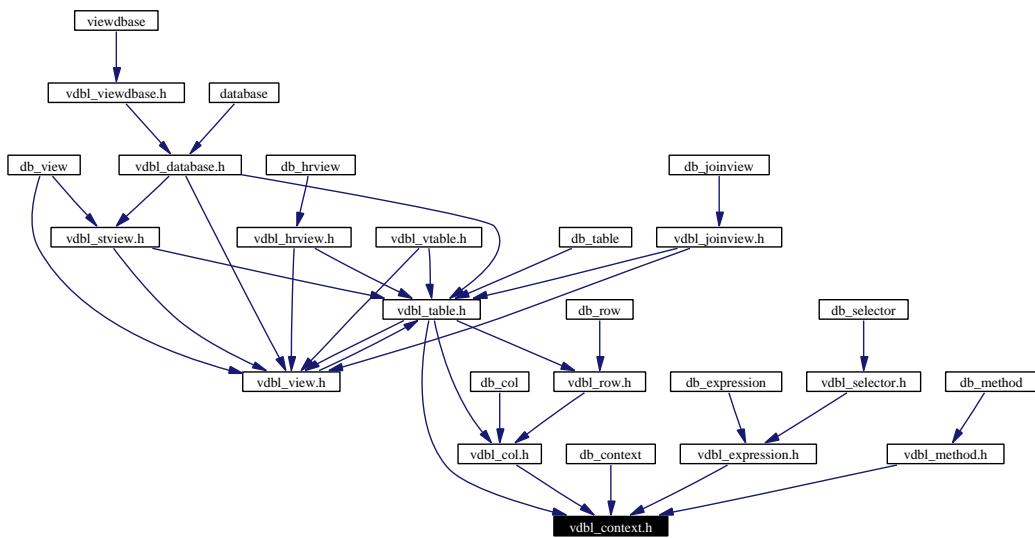
```
#include <vdbl_config.h>
```

```
#include <vdbl_types.h>
```

Include dependency graph for vdbl_context.h:



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



Compounds

- class `_VDBL_context`
base class for context objects.

Typedefs

- typedef `_VDBL_context context`
evaluation context base class.

8.16.1 Detailed Description

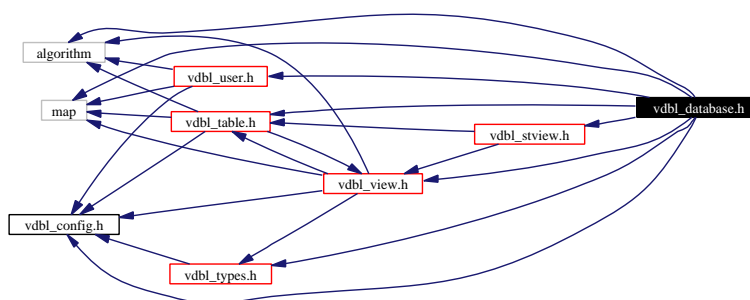
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vdbl_context.h](#).

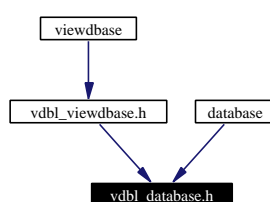
8.17 vdbl_database.h File Reference

```
#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:



Compounds

- class [_VDBL_acl](#)
Access control list.
- class [_VDBL_aclentry](#)
entry in the access control list.
- class [_VDBL_database](#)
the [database](#) class.

- class `_VDBL_tableflags`
flags for one table.
- class `_VDBL_userflags`
The permission flags for a user.
- class `_VDBL_viewflags`
flags for one view.
- 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.

8.17.1 Detailed Description

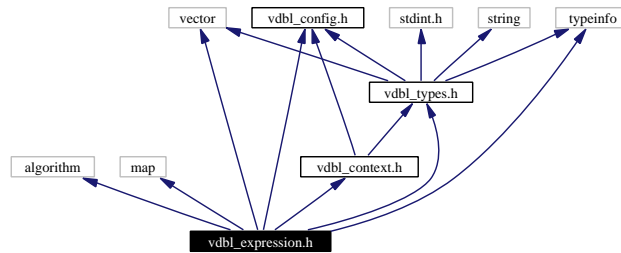
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file `vdbl_database.h`.

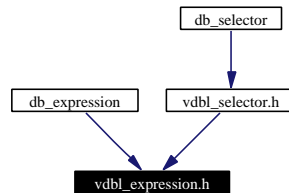
8.18 `vdbl_expression.h` File Reference

```
#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:



Compounds

- class `_VDBL_eval_expr`
- class `_VDBL_exprcol`
- class `_VDBL_exprequal`
- class `_VDBL_exprneql`
- class `_VDBL_exprrow`

8.18.1 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_expression.h](#).

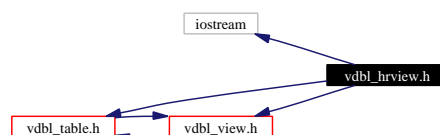
8.19 vdbl_hrview.h File Reference

```

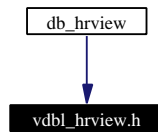
#include <iostream>
#include <vdbl_table.h>
#include <vdbl_view.h>

```

Include dependency graph for vdbl_hrview.h:



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



Compounds

- class `_VDBL_hierarchicalview`
hierarchical view class.
- class `hierarchical_view`
hierarchical view class onto a stack of tables.

8.19.1 Detailed Description

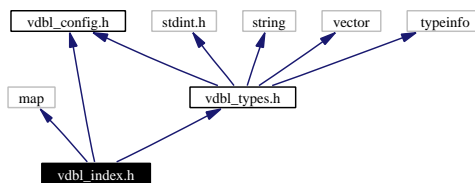
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vdbl_hrview.h](#).

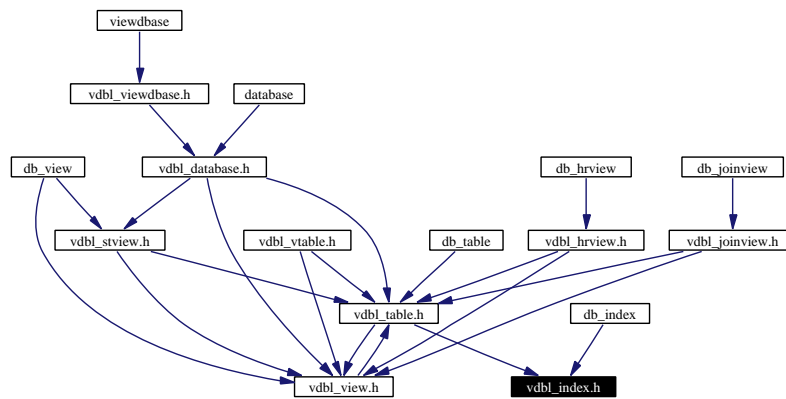
8.20 vdbl_index.h File Reference

```
#include <map>
#include <vdbl_config.h>
#include <vdbl_types.h>
```

Include dependency graph for vdbl_index.h:



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



Compounds

- class `__VDBL_index`
- class `_VDBL_index`
- class `index`

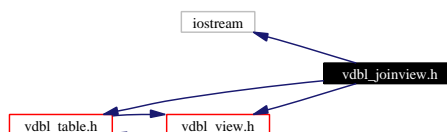
8.20.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Definition in file [vdbl_index.h](#).

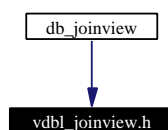
8.21 vdbl_joinview.h File Reference

```
#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:



Compounds

- class `_VDBL_joinview`
- class `join_view`

8.21.1 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_joinview.h](#).

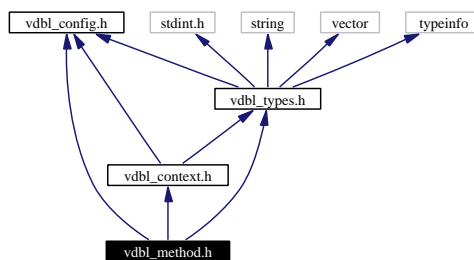
8.22 vdbl_method.h File Reference

```
#include <vdbl_config.h>
```

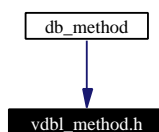
```
#include <vdbl_types.h>
```

```
#include <vdbl_context.h>
```

Include dependency graph for `vdbl_method.h`:



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



Compounds

- class `_VDBL_method`
base class for methods usable in `_VDBL.mthdcol` columns.
- class `method`
base class for methods usable in `method` columns.

8.22.1 Detailed Description

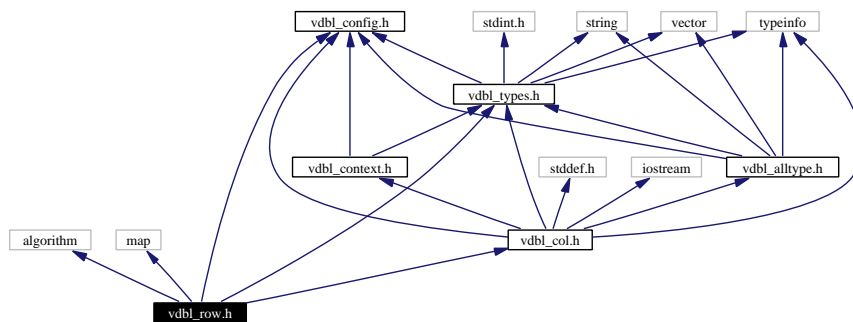
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vdbl_method.h](#).

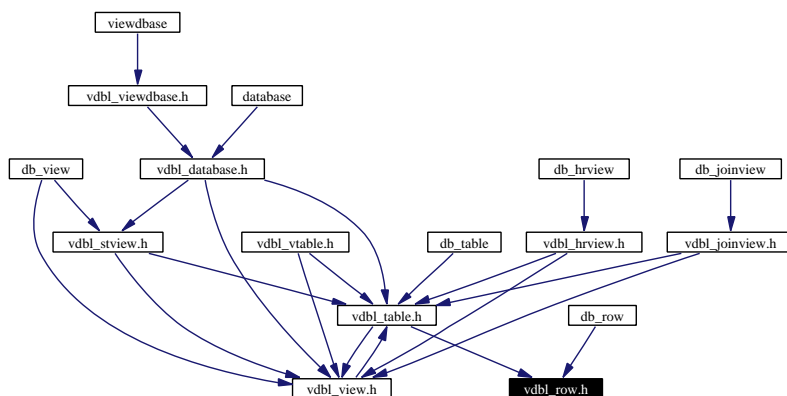
8.23 vdbl_row.h File Reference

```
#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:



Compounds

- class [_VDBL_row](#)
row class.

- class [row](#)
class implementing [table](#) rows.

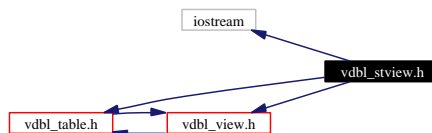
8.23.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Definition in file [vdbl_row.h](#).

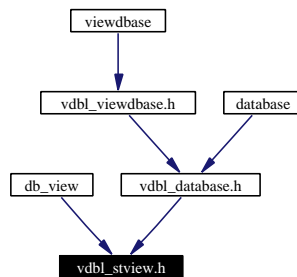
8.24 vdbl_stview.h File Reference

```
#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:



Compounds

- class [_VDBL_standardview](#)
*standard [view](#) onto **one** [table](#).*
- class [view](#)
standard [view](#) class onto a [single](#) [table](#).

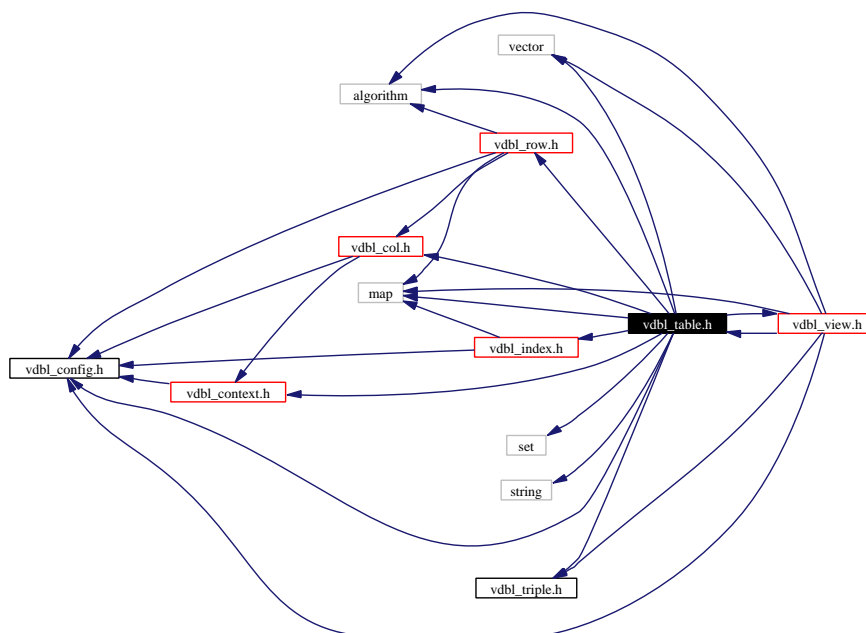
8.24.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Definition in file [vdbl_stview.h](#).

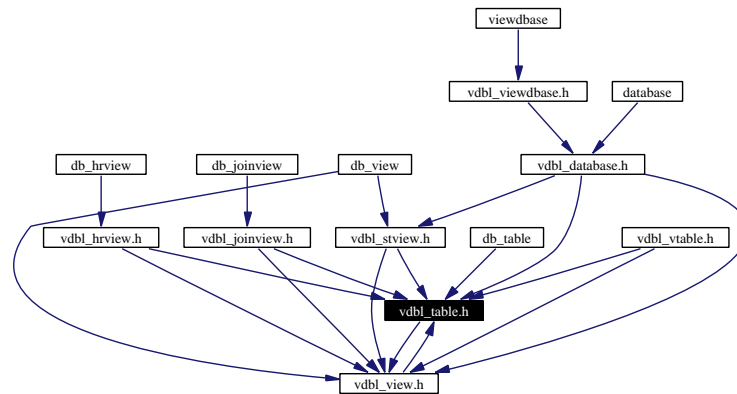
8.25 vdbl_table.h File Reference

```
#include <algorithm>
#include <map>
#include <set>
#include <vector>
#include <string>
#include <vdbl_config.h>
#include <vdbl_row.h>
#include <vdbl_col.h>
#include <vdbl_index.h>
#include <vdbl_context.h>
#include <vdbl_triple.h>
#include <vdbl_view.h>
```

Include dependency graph for vdbl_table.h:



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



Compounds

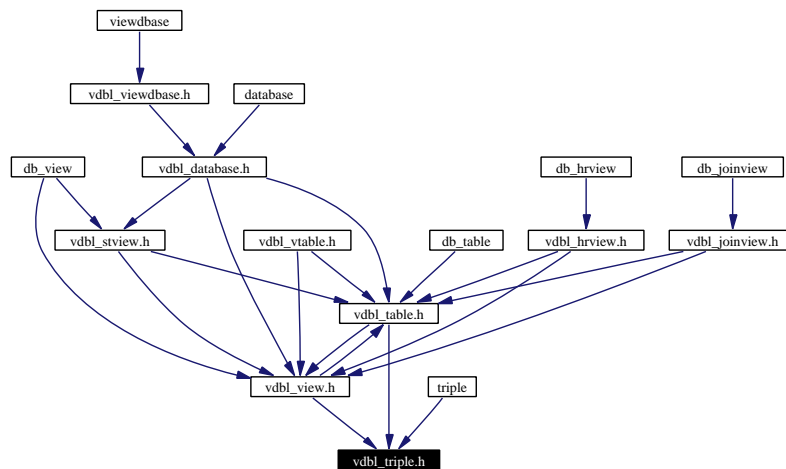
- class `_VDBL_standardtable`
standard table in databases, constructed from rows and columns.
- struct `_VDBL_table::row_iterator`
- struct `_VDBL_table::col_iterator`
- class `_VDBL_table::col_iterator_base`
- class `_VDBL_table`
the base class describing database tables.
- class `col_spec`
column specification.
- class `standard_table`
standard table of a database.
- class `table`
base class for tables in a database.

8.25.1 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_table.h](#).

8.26 vdbl_triple.h File Reference

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



Compounds

- struct [triple](#)
triple holds three objects of arbitrary type.

Functions

- `template<class _T1, class _T2, class _T3> bool operator==(const triple< _T1, _T2, _T3 > &_x, const triple< _T1, _T2, _T3 > &_y)`
Two triples of the same type are equal iff their members are equal.
- `template<class _T1, class _T2, class _T3> bool operator<(const triple< _T1, _T2, _T3 > &_x, const triple< _T1, _T2, _T3 > &_y)`
This is lexicographic ordering of triples.
- `template<class _T1, class _T2, class _T3> bool operator!=(const triple< _T1, _T2, _T3 > &_x, const triple< _T1, _T2, _T3 > &_y)`
Uses operator== to find the result.
- `template<class _T1, class _T2, class _T3> bool operator>(const triple< _T1, _T2, _T3 > &_x, const triple< _T1, _T2, _T3 > &_y)`
Uses operator< to find the result.
- `template<class _T1, class _T2, class _T3> bool operator<=(const triple< _T1, _T2, _T3 > &_x, const triple< _T1, _T2, _T3 > &_y)`
Uses operator< to find the result.
- `template<class _T1, class _T2, class _T3> bool operator>=(const triple< _T1, _T2, _T3 > &_x, const triple< _T1, _T2, _T3 > &_y)`
Uses operator< to find the result.
- `template<class _T1, class _T2, class _T3> triple< _T1, _T2, _T3 > make_triple(const _T1 &_x, const _T2 &_y, const _T3 &_z)`

A convenience wrapper for creating a [triple](#) from three objects.

8.26.1 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_triple.h](#).

8.26.2 Function Documentation

8.26.2.1 `template<class _T1, class _T2, class _T3> triple<_T1, _T2, _T3> make_triple (const _T1 & _x, const _T2 & _y, const _T3 & _z) [inline]`

Parameters:

- `x` The first object.
- `y` The second object.
- `z` The third object.

Returns:

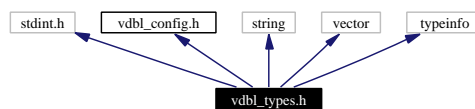
A newly-constructed [triple](#)<> object of the appropriate type.

Definition at line 140 of file [vdbl_triple.h](#).

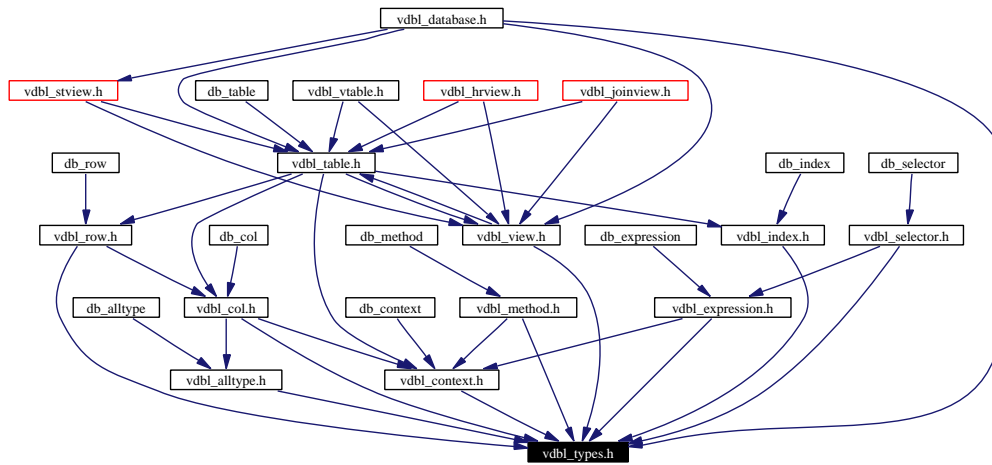
8.27 vdbl_types.h File Reference

```
#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:



Compounds

- class `_VDBL_colflags`
additional [table](#) information for a column.

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.

8.27.1 Detailed Description

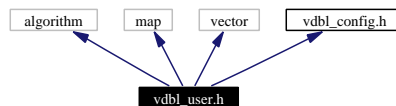
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vdbl_types.h](#).

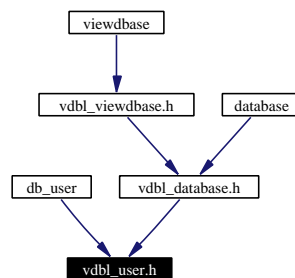
8.28 vdbl_user.h File Reference

```
#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:



Compounds

- class `_VDBL_user`

8.28.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

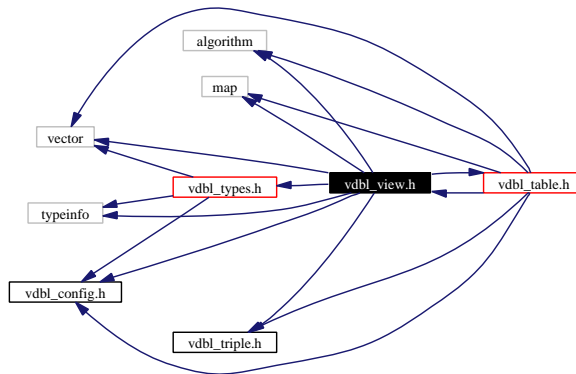
Definition in file [vdbl_user.h](#).

8.29 vdbl_view.h File Reference

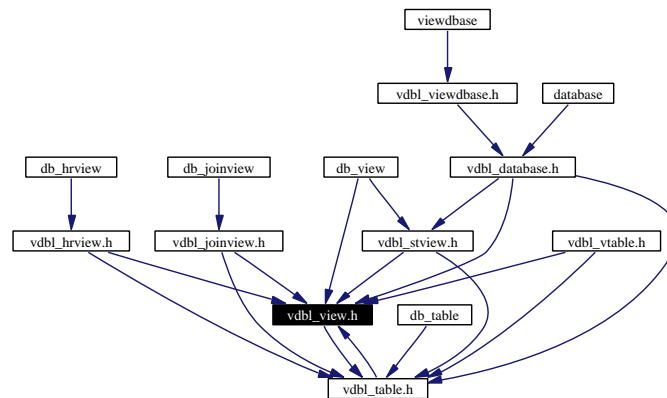
```
#include <algorithm>
#include <map>
#include <vector>
```

```
#include <typeinfo>
#include <vdbl_table.h>
#include <vdbl_triple.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:



Compounds

- struct [_VDBL_view::_row_iterator](#)
- class [_VDBL_view::_row_iterator_base](#)
- struct [_VDBL_view::_col_iterator](#)
- class [_VDBL_view::_col_iterator_base](#)
- struct [_VDBL_view::_default_iterator](#)
- class [_VDBL_view](#)

base class of all views.

Enumerations

- enum `_V_enum`
different view properties.

8.29.1 Detailed Description

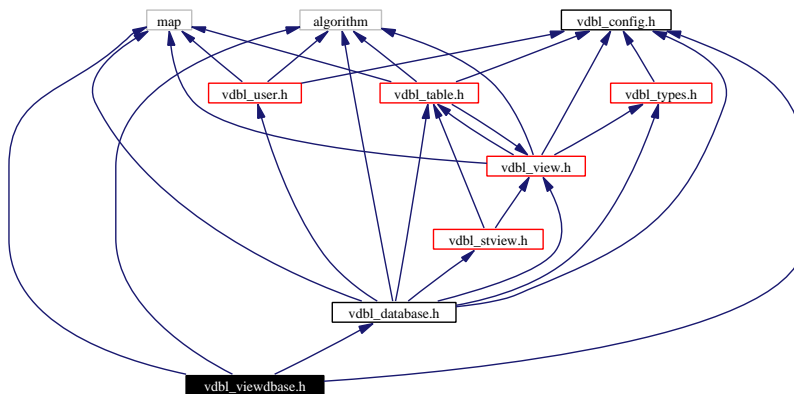
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vdbl_view.h](#).

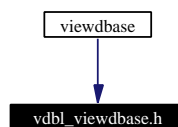
8.30 vdbl_viewbase.h File Reference

```
#include <algorithm>
#include <map>
#include <vdbl_config.h>
#include <vdbl_database.h>
```

Include dependency graph for vdbl_viewbase.h:



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



Compounds

- class `_VDBL_viewbase`
a view to a complete database.

- class [viewbase](#)
a *view* to a complete *database*.

8.30.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly.

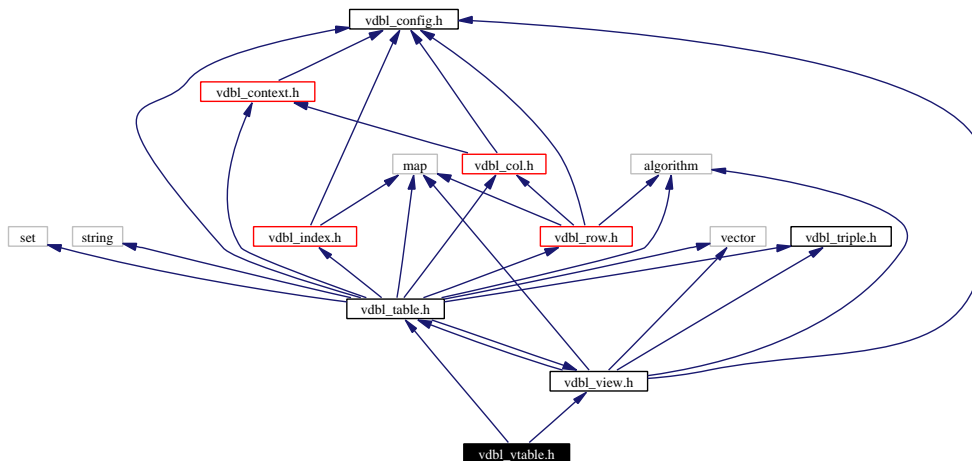
Definition in file [vdbl_viewbase.h](#).

8.31 vdbl_vtable.h File Reference

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

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

Include dependency graph for vdbl_vtable.h:



Compounds

- class [_VDBL_viewtable](#)
- class [view_table](#)

8.31.1 Detailed Description

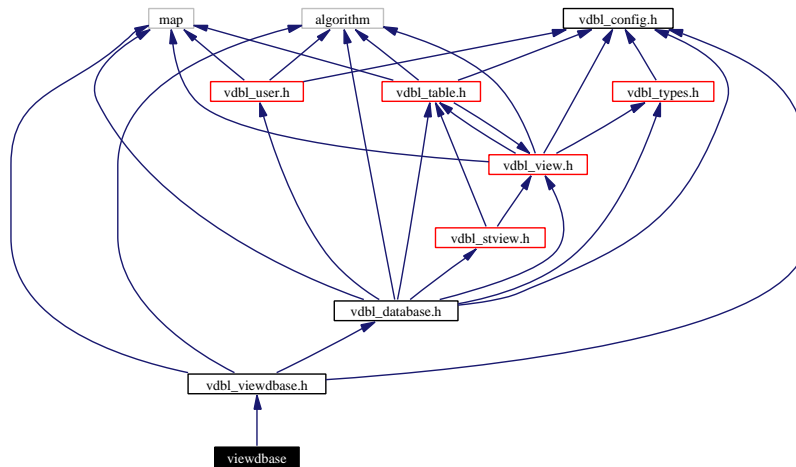
This is an internal header file, included by other library headers. You should not attempt to use it directly.

Definition in file [vdbl_vtable.h](#).

8.32 viewbase File Reference

```
#include <vdbl_viewbase.h>
```

Include dependency graph for viewbase:



8.32.1 Detailed Description

This is the external header file intended for direct use.

Definition in file [viewdbase](#).

Index

- ~_VDBL_acl
 - _VDBL_acl, 18
 - ~_VDBL_aclentry
 - _VDBL_aclentry, 19
 - ~_VDBL_alltype
 - _VDBL_alltype, 22
 - ~_VDBL_alltype_base
 - _VDBL_alltype_base, 24
 - ~_VDBL_col
 - _VDBL_col, 24
 - ~_VDBL_colbase
 - _VDBL_colbase, 28
 - ~_VDBL_colflags
 - _VDBL_colflags, 30
 - ~_VDBL_context
 - _VDBL_context, 31
 - ~_VDBL_database
 - _VDBL_database, 33
 - ~_VDBL_hierarchicalview
 - _VDBL_hierarchicalview, 43
 - ~_VDBL_method
 - _VDBL_method, 47
 - ~_VDBL_mixtype
 - _VDBL_mixtype, 48
 - ~_VDBL_mthdcol
 - _VDBL_mthdcol, 55
 - ~_VDBL_row
 - _VDBL_row, 58
 - ~_VDBL_standardtable
 - _VDBL_standardtable, 61
 - ~_VDBL_standardview
 - _VDBL_standardview, 66
 - ~_VDBL_stdcol
 - _VDBL_stdcol, 69
 - ~_VDBL_table
 - _VDBL_table, 73
 - ~_VDBL_tableflags
 - _VDBL_tableflags, 77
 - ~_VDBL_userflags
 - _VDBL_userflags, 79
 - ~_VDBL_view
 - _VDBL_view, 82
 - ~_VDBL_viewbase
 - _VDBL_viewbase, 88
 - ~_VDBL_viewflags
 - _VDBL_viewflags, 90
 - ~_VDBL_colbase
 - _VDBL_colbase, 17
 - ~col_spec
 - col_spec, 98
 - ~standard_table
 - standard_table, 118
 - ~viewbase
 - viewbase, 135
 - _T_colspec
 - _VDBL_hierarchicalview, 42
 - _VDBL_standardtable, 60
 - _VDBL_standardview, 65
 - _VDBL_table, 72
 - _VDBL_view, 81
 - _VDBL_viewtable, 92
 - hierarchical_view, 107
 - standard_table, 117
 - table, 122
 - view, 130
 - _T_ptrcolspec
 - _VDBL_standardtable, 60
 - _VDBL_table, 72
 - _VDBL_viewtable, 92
 - standard_table, 117
 - table, 123
 - _VDBL_acl, 17
 - ~_VDBL_acl, 18
 - _VDBL_acl, 18
 - colwise, 18
 - global, 18
 - operator=, 18
 - _VDBL_aclentry, 19
 - ~_VDBL_aclentry, 19
 - _VDBL_aclentry, 19
 - append, 20
 - commit_to_parent, 20
 - drop, 20
 - operator=, 20
 - read, 20
 - write, 20
 - _VDBL_alltype, 20
 - ~_VDBL_alltype, 22
 - _VDBL_alltype, 22
 - content, 22
 - get_type, 22
 - operator!=, 22
 - operator=, 22, 23
 - operator==, 23
 - _VDBL_alltype_base, 23
 - ~_VDBL_alltype_base, 24
 - _VDBL_alltype_base, 24
 - _VDBL_col, 24
 - ~_VDBL_col, 24
 - _VDBL_col, 24, 25
-

- def, 25
- def_copy, 25, 26
- get, 26
- get_copy, 26
- get_ptr, 26
- get_ptr_to_val, 26
- operator<<, 27
- return_type_id, 26
- set, 26
- setcontext, 24
- _VDBL_colbase, 27
 - ~_VDBL_colbase, 28
 - _VDBL_colbase, 28
 - def_copy, 29
 - get_copy, 29
 - new_copy, 29
 - return_type, 27
 - return_type_id, 29
 - setcontext, 29
- _VDBL_colflags, 30
 - ~_VDBL_colflags, 30
 - _VDBL_colflags, 30
 - has_default, 30
 - master_index, 30
- _VDBL_colid
 - internal, 15
- _VDBL_context, 31
 - ~_VDBL_context, 31
 - _VDBL_context, 31
 - table, 32
- _VDBL_database, 32
 - ~_VDBL_database, 33
 - _VDBL_database, 33
 - _VDBL_viewdbase, 33
 - create_table, 34
 - create_view, 34
 - drop_table, 34
 - drop_view, 34
 - get_table, 35
 - get_tableid, 35
 - get_userid, 35
 - get_view, 35
 - get_viewid, 35, 36
 - has_table, 36
 - has_view, 36
- _VDBL_date, 36
 - day, 37
 - microseconds, 37
 - month, 38
 - operator+, 37
 - operator+=", 37
 - operator-, 37
 - operator-=, 37
 - seconds, 38
 - timezone, 38
 - year, 38
- _VDBL_dateinterval, 38
 - days, 40
 - microseconds, 40
 - operator *, 39
 - operator *=", 39
 - operator+, 39
 - operator+=", 39
 - operator-, 39
 - operator-=, 39
 - operator/, 39
 - operator/=, 39
 - seconds, 40
 - years, 40
- _VDBL_hierarchicalview, 40
 - ~_VDBL_hierarchicalview, 43
 - _T_colspec, 42
 - _VDBL_hierarchicalview, 43
 - _VDBL_table, 42
 - _V_c, 46
 - _V_cx, 46
 - _V_r, 46
 - _copy_col_data, 43
 - _copy_def_data, 44
 - _copy_row_data, 44
 - _next_col, 44
 - _next_def_col, 44
 - _next_row, 44
 - _prev_col, 44
 - _prev_def_col, 45
 - _prev_row, 45
 - col_const_iterator, 42
 - default_const_iterator, 42
 - get, 45
 - get_change_ctr, 42
 - get_colinfo, 45
 - made_change, 42
 - pop_table, 45
 - push_table, 45
 - remove, 46
 - row_const_iterator, 43
- _VDBL_method, 46
 - ~_VDBL_method, 47
 - _VDBL_method, 47
 - operator(), 47
- _VDBL_mixtype, 47
 - ~_VDBL_mixtype, 48
 - _VDBL_mixtype, 48–51
 - b, 51
 - clear, 48
 - d, 51
 - di, 51
 - dt, 51

- empty, 48
- is_vector, 48
- n, 51
- nb, 51
- nd, 52
- nn, 52
- nu, 52
- operator=, 48, 52, 53
- s, 54
- u, 54
- _VDBL_mthdcol, 54
 - ~_VDBL_mthdcol, 55
 - _VDBL_mthdcol, 55, 56
 - def, 55
 - def_copy, 56
 - get, 55
 - get_copy, 56
 - get_ptr, 55
 - new_copy, 56
 - return_type_id, 56
 - setcontext, 57
- _VDBL_row, 57
 - ~_VDBL_row, 58
 - _VDBL_row, 58
 - drop, 58
 - get_col, 58
 - has_col, 58
 - insert, 58
 - update, 58
- _VDBL_rowid
 - internal, 15
- _VDBL_standardtable, 59
 - ~_VDBL_standardtable, 61
 - _T_colspec, 60
 - _T_ptrcolspec, 60
 - _VDBL_standardtable, 61
 - _VDBL_view, 60
 - _next_col, 61
 - _next_row, 61
 - add_col, 62
 - col_const_iterator, 60
 - get_col_id, 62
 - get_colid, 62
 - get_colinfo, 62
 - get_rowid, 62
 - insert_row, 62
 - made_change, 62
 - row_const_iterator, 61
- _VDBL_standardview, 63
 - ~_VDBL_standardview, 66
 - _T_colspec, 65
 - _VDBL_standardview, 65, 66
 - _VDBL_table, 65
 - _V_c, 68
- _V_r, 68
 - _copy_col_data, 66
 - _copy_def_data, 66
 - _copy_row_data, 66
 - _next_col, 66
 - _next_def_col, 66
 - _next_row, 67
 - _prev_col, 67
 - _prev_def_col, 67
 - _prev_row, 67
 - col_const_iterator, 65
 - default_const_iterator, 65
 - defaults_const_iterator, 63
 - get, 67
 - get_change_ctr, 64
 - get_colinfo, 67
 - made_change, 64
 - remove, 68
 - row_const_iterator, 65
- _VDBL_stdcol, 68
 - ~_VDBL_stdcol, 69
 - _VDBL_stdcol, 69, 70
 - def, 70
 - get_val, 70
 - new_copy, 70
 - set, 70
 - set_default, 70
 - setcontext, 70
- _VDBL_table, 71
 - ~_VDBL_table, 73
 - _T_colspec, 72
 - _T_ptrcolspec, 72
 - _VDBL_hierarchicalview, 42
 - _VDBL_standardview, 65
 - _VDBL_table, 73
 - _VDBL_view, 72
 - _next_col, 73
 - _next_row, 73
 - add_col, 73
 - col_const_iterator, 72
 - get_colid, 74
 - get_colinfo, 74
 - get_rowid, 74
 - insert_row, 74
 - made_change, 74
 - row_const_iterator, 72
- _VDBL_table::col_iterator, 74
- _VDBL_table::col_iterator_base, 75
- _VDBL_table::row_iterator, 76
- _VDBL_tableflags, 76
 - ~_VDBL_tableflags, 77
 - _VDBL_tableflags, 77
 - ACLs, 77
 - global, 78

- operator=, 77
- temporary, 78
- unrestricted, 78
- _VDBL_tableid
 - internal, 15
- _VDBL_userflags, 78
 - ~_VDBL_userflags, 79
 - _VDBL_userflags, 79
 - table_privileges, 79
 - view_privileges, 79
- _VDBL_userid
 - internal, 15
- _VDBL_view, 79
 - ~_VDBL_view, 82
 - _T_colspec, 81
 - _VDBL_standardtable, 60
 - _VDBL_table, 72
 - _VDBL_view, 82
 - _VDBL_viewtable, 92
 - _copy_col_data, 82
 - _copy_def_data, 82
 - _copy_row_data, 82
 - _next_col, 82
 - _next_def_col, 82
 - _next_row, 83
 - _prev_col, 83
 - _prev_def_col, 83
 - _prev_row, 83
 - col_const_iterator, 81
 - default_const_iterator, 81
 - get_change_ctr, 80
 - get_colinfo, 83
 - made_change, 80
 - row_const_iterator, 81
- _VDBL_view::_col_iterator, 84
- _VDBL_view::_col_iterator_base, 84
- _VDBL_view::_default_iterator, 85
- _VDBL_view::_row_iterator, 86
- _VDBL_view::_row_iterator_base, 86
- _VDBL_viewdbase, 87
 - ~_VDBL_viewdbase, 88
 - _VDBL_database, 33
 - _VDBL_viewdbase, 88
 - database, 100
 - get_tableid, 88
 - get_view, 89
 - has_view, 89
- _VDBL_viewflags, 89
 - ~_VDBL_viewflags, 90
 - _VDBL_viewflags, 90
 - ACLs, 91
 - global, 91
 - operator=, 90
- _VDBL_viewid
 - internal, 15
- _VDBL_viewtable, 91
 - _T_colspec, 92
 - _T_ptrcolspec, 92
 - _VDBL_view, 92
 - _next_col, 93
 - _next_row, 93
 - add_col, 93
 - col_const_iterator, 93
 - get_colid, 93
 - get_rowid, 94
 - insert_row, 94
 - made_change, 94
 - row_const_iterator, 93
- _V_c
 - _VDBL_hierarchicalview, 46
 - _VDBL_standardview, 68
 - hierarchical_view, 110
 - view, 133
- _V_cx
 - _VDBL_hierarchicalview, 46
 - hierarchical_view, 110
- _V_enum
 - external, 13
- _V_r
 - _VDBL_hierarchicalview, 46
 - _VDBL_standardview, 68
 - hierarchical_view, 110
 - view, 133
- VDBL_colbase, 16
 - ~--VDBL_colbase, 17
 - VDBL_colbase, 17
 - new_copy, 17
- _copy_col_data
 - _VDBL_hierarchicalview, 43
 - _VDBL_standardview, 66
 - _VDBL_view, 82
 - hierarchical_view, 107
 - view, 131
- _copy_def_data
 - _VDBL_hierarchicalview, 44
 - _VDBL_standardview, 66
 - _VDBL_view, 82
 - hierarchical_view, 107
 - view, 131
- _copy_row_data
 - _VDBL_hierarchicalview, 44
 - _VDBL_standardview, 66
 - _VDBL_view, 82
 - hierarchical_view, 108
 - view, 131
- _next_col
 - _VDBL_hierarchicalview, 44
 - _VDBL_standardtable, 61

- [_VDBL_standardview](#), 66
 - [_VDBL_table](#), 73
 - [_VDBL_view](#), 82
 - [_VDBL_viewtable](#), 93
 - [hierarchical_view](#), 108
 - [standard_table](#), 118
 - [table](#), 123
 - [view](#), 131
- [_next_def_col](#)
 - [_VDBL_hierarchicalview](#), 44
 - [_VDBL_standardview](#), 66
 - [_VDBL_view](#), 82
 - [hierarchical_view](#), 108
 - [view](#), 131
- [_next_row](#)
 - [_VDBL_hierarchicalview](#), 44
 - [_VDBL_standardtable](#), 61
 - [_VDBL_standardview](#), 67
 - [_VDBL_table](#), 73
 - [_VDBL_view](#), 83
 - [_VDBL_viewtable](#), 93
 - [hierarchical_view](#), 108
 - [standard_table](#), 118
 - [table](#), 123
 - [view](#), 131
- [_prev_col](#)
 - [_VDBL_hierarchicalview](#), 44
 - [_VDBL_standardview](#), 67
 - [_VDBL_view](#), 83
 - [hierarchical_view](#), 108
 - [view](#), 132
- [_prev_def_col](#)
 - [_VDBL_hierarchicalview](#), 45
 - [_VDBL_standardview](#), 67
 - [_VDBL_view](#), 83
 - [hierarchical_view](#), 109
 - [view](#), 132
- [_prev_row](#)
 - [_VDBL_hierarchicalview](#), 45
 - [_VDBL_standardview](#), 67
 - [_VDBL_view](#), 83
 - [hierarchical_view](#), 109
 - [view](#), 132
- [acl](#)
 - [external](#), 11
- [aclentry](#)
 - [external](#), 11
- [ACLs](#)
 - [_VDBL_tableflags](#), 77
 - [_VDBL_viewflags](#), 91
- [add_col](#)
 - [_VDBL_standardtable](#), 62
 - [_VDBL_table](#), 73
 - [_VDBL_viewtable](#), 93
 - [standard_table](#), 119
 - [table](#), 123
- [alltype](#), 94
 - [cont_type](#), 95
- [alltype_base](#)
 - [external](#), 11
- [append](#)
 - [_VDBL_aclentry](#), 20
- [b](#)
 - [_VDBL_mixtype](#), 51
- [Classes and types for external use](#), 9
- [Classes and types for internal use](#), 13
- [clear](#)
 - [_VDBL_mixtype](#), 48
- [col](#)
 - [external](#), 11
- [col_base](#), 95
 - [def_copy](#), 96
 - [get_copy](#), 96
 - [new_copy](#), 97
 - [return_type](#), 96
 - [return_type_id](#), 97
 - [setcontext](#), 97
- [col_const_iterator](#)
 - [_VDBL_hierarchicalview](#), 42
 - [_VDBL_standardtable](#), 60
 - [_VDBL_standardview](#), 65
 - [_VDBL_table](#), 72
 - [_VDBL_view](#), 81
 - [_VDBL_viewtable](#), 93
 - [hierarchical_view](#), 107
 - [standard_table](#), 117
 - [table](#), 123
 - [view](#), 130
- [col_spec](#), 97
 - [~col_spec](#), 98
 - [col_spec](#), 98
- [colflags](#)
 - [external](#), 11
- [colid](#)
 - [external](#), 11
- [colwise](#)
 - [_VDBL_acl](#), 18
- [commit_to_parent](#)
 - [_VDBL_aclentry](#), 20
- [cont_type](#)
 - [alltype](#), 95
- [content](#)
 - [_VDBL_alltype](#), 22
- [context](#)
 - [external](#), 11

- create_table
 - _VDBL_database, 34
 - database, 100
- create_view
 - _VDBL_database, 34
 - database, 100, 101
- d
 - _VDBL_mixtype, 51
- database, 98, 136
 - _VDBL_viewdbase, 100
 - create_table, 100
 - create_view, 100, 101
 - drop_table, 101
 - drop_view, 102
 - get_table, 102
 - get_tableid, 103
 - get_userid, 103
 - get_view, 103
 - get_viewid, 103
 - has_table, 104
 - has_view, 104
- date
 - external, 12
- dateinterval
 - external, 12
- day
 - _VDBL_date, 37
- days
 - _VDBL_dateinterval, 40
- db_alltype, 136
- db_col, 137
- db_context, 137
- db_hrview, 138
- db_index, 138
- db_joinview, 139
- db_method, 139
- db_row, 140
- db_table, 140
- db_view, 141
- def
 - _VDBL_col, 25
 - _VDBL_mthdcol, 55
 - _VDBL_stdcol, 70
 - typed_col, 127
- def_copy
 - _VDBL_col, 25, 26
 - _VDBL_colbase, 29
 - _VDBL_mthdcol, 56
 - col_base, 96
- default_const_iterator
 - _VDBL_hierarchicalview, 42
 - _VDBL_standardview, 65
 - _VDBL_view, 81
 - hierarchical_view, 107
 - view, 130
- defaults_const_iterator
 - _VDBL_standardview, 63
 - view, 129
- di
 - _VDBL_mixtype, 51
- drop
 - _VDBL_aclentry, 20
 - _VDBL_row, 58
 - row, 114
- drop_table
 - _VDBL_database, 34
 - database, 101
- drop_view
 - _VDBL_database, 34
 - database, 102
- dt
 - _VDBL_mixtype, 51
- empty
 - _VDBL_mixtype, 48
- external
 - _V_enum, 13
 - acl, 11
 - aclentry, 11
 - alltype_base, 11
 - col, 11
 - colflags, 11
 - colid, 11
 - context, 11
 - date, 12
 - dateinterval, 12
 - mixtype, 12
 - rowid, 12
 - standard_col, 12
 - tableflags, 12
 - tableid, 12
 - userflags, 12
 - userid, 12
 - viewflags, 13
 - viewid, 13
- first
 - triple, 125
- first_type
 - triple, 125
- get
 - _VDBL_col, 26
 - _VDBL_hierarchicalview, 45
 - _VDBL_mthdcol, 55
 - _VDBL_standardview, 67
 - hierarchical_view, 109

- view, [132](#), [133](#)
- get_change_ctr
 - [_VDBL_hierarchicalview](#), [42](#)
 - [_VDBL_standardview](#), [64](#)
 - [_VDBL_view](#), [80](#)
 - [hierarchical_view](#), [106](#)
 - [view](#), [130](#)
- get_col
 - [_VDBL_row](#), [58](#)
 - [row](#), [114](#)
 - [view](#), [133](#)
- get_col_id
 - [_VDBL_standardtable](#), [62](#)
 - [standard_table](#), [119](#)
- get_colid
 - [_VDBL_standardtable](#), [62](#)
 - [_VDBL_table](#), [74](#)
 - [_VDBL_viewtable](#), [93](#)
 - [standard_table](#), [119](#)
 - [table](#), [123](#)
- get_colinfo
 - [_VDBL_hierarchicalview](#), [45](#)
 - [_VDBL_standardtable](#), [62](#)
 - [_VDBL_standardview](#), [67](#)
 - [_VDBL_table](#), [74](#)
 - [_VDBL_view](#), [83](#)
 - [hierarchical_view](#), [109](#)
 - [standard_table](#), [119](#)
 - [table](#), [124](#)
 - [view](#), [133](#)
- get_copy
 - [_VDBL_col](#), [26](#)
 - [_VDBL_colbase](#), [29](#)
 - [_VDBL_mthdcol](#), [56](#)
 - [col_base](#), [96](#)
- get_ptr
 - [_VDBL_col](#), [26](#)
 - [_VDBL_mthdcol](#), [55](#)
- get_ptr_to_val
 - [_VDBL_col](#), [26](#)
- get_row
 - [standard_table](#), [120](#)
- get_rowid
 - [_VDBL_standardtable](#), [62](#)
 - [_VDBL_table](#), [74](#)
 - [_VDBL_viewtable](#), [94](#)
 - [standard_table](#), [120](#)
 - [table](#), [124](#)
- get_table
 - [_VDBL_database](#), [35](#)
 - [database](#), [102](#)
- get_tableid
 - [_VDBL_database](#), [35](#)
 - [_VDBL_viewdbase](#), [88](#)
- database, [103](#)
- viewdbase, [135](#)
- get_type
 - [_VDBL_alltype](#), [22](#)
- get_userid
 - [_VDBL_database](#), [35](#)
 - [database](#), [103](#)
- get_val
 - [_VDBL_stdcol](#), [70](#)
 - [typed_col](#), [127](#)
- get_view
 - [_VDBL_database](#), [35](#)
 - [_VDBL_viewdbase](#), [89](#)
 - [database](#), [103](#)
 - [viewdbase](#), [135](#)
- get_viewid
 - [_VDBL_database](#), [35](#), [36](#)
 - [database](#), [103](#)
- global
 - [_VDBL_acl](#), [18](#)
 - [_VDBL_tableflags](#), [78](#)
 - [_VDBL_viewflags](#), [91](#)
- has_col
 - [_VDBL_row](#), [58](#)
 - [row](#), [115](#)
- has_default
 - [_VDBL_colflags](#), [30](#)
- has_table
 - [_VDBL_database](#), [36](#)
 - [database](#), [104](#)
- has_view
 - [_VDBL_database](#), [36](#)
 - [_VDBL_viewdbase](#), [89](#)
 - [database](#), [104](#)
 - [viewdbase](#), [135](#), [136](#)
- hierarchical_view, [104](#)
 - [_T_colspec](#), [107](#)
 - [_V_c](#), [110](#)
 - [_V_cx](#), [110](#)
 - [_V_r](#), [110](#)
 - [.copy_col_data](#), [107](#)
 - [.copy_def_data](#), [107](#)
 - [.copy_row_data](#), [108](#)
 - [.next_col](#), [108](#)
 - [.next_def_col](#), [108](#)
 - [.next_row](#), [108](#)
 - [.prev_col](#), [108](#)
 - [.prev_def_col](#), [109](#)
 - [.prev_row](#), [109](#)
- [col_const_iterator](#), [107](#)
- [default_const_iterator](#), [107](#)
- [get](#), [109](#)
- [get_change_ctr](#), [106](#)

- get_colinfo, 109
- hierarchical_view, 107
- made_change, 106
- pop_table, 110
- push_table, 110
- remove, 110
- row_const_iterator, 107
- insert
 - _VDBL_row, 58
 - row, 115
 - standard_table, 120
- insert_row
 - _VDBL_standardtable, 62
 - _VDBL_table, 74
 - _VDBL_viewtable, 94
 - standard_table, 120, 121
 - table, 124
- internal
 - _VDBL_colid, 15
 - _VDBL_rowid, 15
 - _VDBL_tableid, 15
 - _VDBL_userid, 15
 - _VDBL_viewid, 15
 - print_it, 16
- is_vector
 - _VDBL_mixtype, 48
- made_change
 - _VDBL_hierarchicalview, 42
 - _VDBL_standardtable, 62
 - _VDBL_standardview, 64
 - _VDBL_table, 74
 - _VDBL_view, 80
 - _VDBL_viewtable, 94
 - hierarchical_view, 106
 - standard_table, 121
 - table, 124
 - view, 129
- make_triple
 - vdbl_triple.h, 160
- master_index
 - _VDBL_colflags, 30
- method, 111
 - operator(), 112
 - return_type, 111
- method_col, 112
 - new_copy, 113
- microseconds
 - _VDBL_date, 37
 - _VDBL_dateinterval, 40
- mixtype
 - external, 12
- month
 - _VDBL_date, 38
- n
 - _VDBL_mixtype, 51
- nb
 - _VDBL_mixtype, 51
- nd
 - _VDBL_mixtype, 52
- new_copy
 - _VDBL_colbase, 29
 - _VDBL_mthdcol, 56
 - _VDBL_stdcol, 70
 - __VDBL_colbase, 17
 - col_base, 97
 - method_col, 113
 - typed_col, 127
- nn
 - _VDBL_mixtype, 52
- nu
 - _VDBL_mixtype, 52
- operator *
 - _VDBL_dateinterval, 39
- operator *=
 - _VDBL_dateinterval, 39
- operator !=
 - _VDBL_alltype, 22
 - vdbl_triple.h, 159
- operator()
 - _VDBL_method, 47
 - method, 112
- operator+
 - _VDBL_date, 37
 - _VDBL_dateinterval, 39
- operator+=
 - _VDBL_date, 37
 - _VDBL_dateinterval, 39
- operator-
 - _VDBL_date, 37
 - _VDBL_dateinterval, 39
- operator-=
 - _VDBL_date, 37
 - _VDBL_dateinterval, 39
- operator/
 - _VDBL_dateinterval, 39
- operator/=
 - _VDBL_dateinterval, 39
- operator<
 - vdbl_triple.h, 159
- operator<<
 - _VDBL_col, 27
 - vdbl_col.h, 147
- operator<=
 - vdbl_triple.h, 159

- operator=
 - _VDBL_acl, 18
 - _VDBL_aclentry, 20
 - _VDBL_alltype, 22, 23
 - _VDBL_mixtype, 48, 52, 53
 - _VDBL_tableflags, 77
 - _VDBL_viewflags, 90
 - row, 115
- operator==
 - _VDBL_alltype, 23
 - vdbl.triple.h, 159
- operator>
 - vdbl.triple.h, 159
- operator>=
 - vdbl.triple.h, 159
- pop_table
 - _VDBL_hierarchicalview, 45
 - hierarchical_view, 110
- print_it
 - internal, 16
- push_table
 - _VDBL_hierarchicalview, 45
 - hierarchical_view, 110
- read
 - _VDBL_aclentry, 20
- remove
 - _VDBL_hierarchicalview, 46
 - _VDBL_standardview, 68
 - hierarchical_view, 110
 - view, 133
- return_type
 - _VDBL_colbase, 27
 - col_base, 96
 - method, 111
- return_type_id
 - _VDBL_col, 26
 - _VDBL_colbase, 29
 - _VDBL_mthdcol, 56
 - col_base, 97
- row, 113
 - drop, 114
 - get_col, 114
 - has_col, 115
 - insert, 115
 - operator=, 115
 - row, 114
 - update, 115
- row_const_iterator
 - _VDBL_hierarchicalview, 43
 - _VDBL_standardtable, 61
 - _VDBL_standardview, 65
 - _VDBL_table, 72
 - _VDBL_view, 81
 - _VDBL_viewtable, 93
 - hierarchical_view, 107
 - standard_table, 118
 - table, 123
 - view, 130
- rowid
 - external, 12
- s
 - _VDBL_mixtype, 54
- second
 - triple, 125
- second_type
 - triple, 125
- seconds
 - _VDBL_date, 38
 - _VDBL_dateinterval, 40
- set
 - _VDBL_col, 26
 - _VDBL_stdcol, 70
 - typed_col, 127
- set_default
 - _VDBL_stdcol, 70
 - typed_col, 127
- setcontext
 - _VDBL_col, 24
 - _VDBL_colbase, 29
 - _VDBL_mthdcol, 57
 - _VDBL_stdcol, 70
 - col_base, 97
 - typed_col, 128
- standard_col
 - external, 12
- standard_table, 115
 - ~standard_table, 118
 - _T_colspec, 117
 - _T_ptrcolspec, 117
 - _next_col, 118
 - _next_row, 118
 - add_col, 119
 - col_const_iterator, 117
 - get_col_id, 119
 - get_colid, 119
 - get_colinfo, 119
 - get_row, 120
 - get_rowid, 120
 - insert, 120
 - insert_row, 120, 121
 - made_change, 121
 - row_const_iterator, 118
 - standard_table, 118
- table, 121

- `_T_colspec`, 122
- `_T_ptrcolspec`, 123
- `_VDBL_context`, 32
- `_next_col`, 123
- `_next_row`, 123
- `add_col`, 123
- `col_const_iterator`, 123
- `get_colid`, 123
- `get_colinfo`, 124
- `get_rowid`, 124
- `insert_row`, 124
- `made_change`, 124
- `row_const_iterator`, 123
- `table_privileges`
 - `_VDBL_userflags`, 79
- `tableflags`
 - `external`, 12
- `tableid`
 - `external`, 12
- `temporary`
 - `_VDBL_tableflags`, 78
- `third`
 - `triple`, 125
- `third_type`
 - `triple`, 125
- `timezone`
 - `_VDBL_date`, 38
- `triple`, 124, 141
 - `first`, 125
 - `first_type`, 125
 - `second`, 125
 - `second_type`, 125
 - `third`, 125
 - `third_type`, 125
 - `triple`, 125
- `typed_col`, 126
 - `def`, 127
 - `get_val`, 127
 - `new_copy`, 127
 - `set`, 127
 - `set_default`, 127
 - `setcontext`, 128
- `u`
 - `_VDBL_mixtype`, 54
- `unrestricted`
 - `_VDBL_tableflags`, 78
- `update`
 - `_VDBL_row`, 58
 - `row`, 115
- `userflags`
 - `external`, 12
- `userid`
 - `external`, 12
- `vdbl_alltype.h`, 142
 - `VDBL_MIXTYPE_ALLOCED_B`, 144
 - `VDBL_MIXTYPE_ALLOCED_D`, 144
 - `VDBL_MIXTYPE_ALLOCED_N`, 144
 - `VDBL_MIXTYPE_ALLOCED_NONE`, 144
 - `VDBL_MIXTYPE_ALLOCED_S`, 144
 - `VDBL_MIXTYPE_ALLOCED_U`, 144
 - `VDBL_MIXTYPE_EMPTY`, 144
- `vdbl_col.h`, 145
 - `operator<<`, 147
- `vdbl_config.h`, 147
- `vdbl_context.h`, 147
- `vdbl_database.h`, 149
- `vdbl_expression.h`, 150
- `vdbl_hrview.h`, 151
- `vdbl_index.h`, 152
- `vdbl_joinview.h`, 153
- `vdbl_method.h`, 154
- `VDBL_MIXTYPE_ALLOCED_B`
 - `vdbl_alltype.h`, 144
- `VDBL_MIXTYPE_ALLOCED_D`
 - `vdbl_alltype.h`, 144
- `VDBL_MIXTYPE_ALLOCED_N`
 - `vdbl_alltype.h`, 144
- `VDBL_MIXTYPE_ALLOCED_NONE`
 - `vdbl_alltype.h`, 144
- `VDBL_MIXTYPE_ALLOCED_S`
 - `vdbl_alltype.h`, 144
- `VDBL_MIXTYPE_ALLOCED_U`
 - `vdbl_alltype.h`, 144
- `VDBL_MIXTYPE_EMPTY`
 - `vdbl_alltype.h`, 144
- `vdbl_row.h`, 155
- `vdbl_stview.h`, 156
- `vdbl_table.h`, 157
- `vdbl_triple.h`, 158
 - `make_triple`, 160
 - `operator`
 - `=`, 159
 - `operator<`, 159
 - `operator<=`, 159
 - `operator==`, 159
 - `operator>`, 159
 - `operator>=`, 159
- `vdbl_types.h`, 160
- `vdbl_user.h`, 162
- `vdbl_view.h`, 162
- `vdbl_viewdbase.h`, 164
- `vdbl_vtable.h`, 165
- `view`, 128
 - `_T_colspec`, 130
 - `_V_c`, 133
 - `_V_r`, 133

- [_copy_col_data](#), 131
 - [_copy_def_data](#), 131
 - [_copy_row_data](#), 131
 - [_next_col](#), 131
 - [_next_def_col](#), 131
 - [_next_row](#), 131
 - [_prev_col](#), 132
 - [_prev_def_col](#), 132
 - [_prev_row](#), 132
 - [col_const_iterator](#), 130
 - [default_const_iterator](#), 130
 - [defaults_const_iterator](#), 129
 - [get](#), 132, 133
 - [get_change_ctr](#), 130
 - [get_col](#), 133
 - [get_colinfo](#), 133
 - [made_change](#), 129
 - [remove](#), 133
 - [row_const_iterator](#), 130
 - [view](#), 130
- [view_privileges](#)
 - [_VDBL_userflags](#), 79
- [viewdbase](#), 133, 165
 - [~viewdbase](#), 135
 - [get_tableid](#), 135
 - [get_view](#), 135
 - [has_view](#), 135, 136
 - [viewdbase](#), 134
- [viewflags](#)
 - [external](#), 13
- [viewid](#)
 - [external](#), 13
- [write](#)
 - [_VDBL_aclentry](#), 20
- [year](#)
 - [_VDBL_date](#), 38
- [years](#)
 - [_VDBL_dateinterval](#), 40