rdbxx

1.0.5

Generated by Doxygen 1.5.6

Tue Sep 15 11:07:15 2009

Contents

1 RDB User's Guide

1.1 Copyright

1.2 Introduction

- Purpose
- RDB format

1.3 Basic Interface

- Creating RDB objects for reading or writing
- RDB header and table information
- Accessing comments and columns
- Table I/O
- Rewinding tables
- Automatic indexing for column data
- Group columns

- Creating RDB objects
- Header and table size
- Comment initializers
- Comment accessors
- Column initializers
- Column accessors

1.4 Specialized Interface

- Data storage
  1. Default data storage
  2. User-specified data storage

  3. Auto-indexing

- Stream manipulation
  1. Effects on data storage
  2. Limitations

1.5 Examples

- Creating an output file
- A simple filter
- Using group columns

Author:

`M. Tibbetts`

2 Basic Interface

2.1 Creating RDB objects for reading or writing

The RDB library allows the user to read or write RDB formatted data. Users may attach an RDB object to either a file or a C++ stream. The object may be associated with a file or stream either at the time it is created, via the constructors RDB::RDB, or after the object is created, via the RDB::open methods.

```
// Open for input, using explicit open mode.
RDB irdb( "input.rdb", ios::in );

// Open for input, using default open mode.
RDB irdb( "input.rdb" );

// Open for input, using a pointer to an istream.
RDB irdb( &cin );

// Open for output.  The explicit open mode is required, otherwise
//   it will default to open for input.
RDB ordb( "output.rdb", ios::out );

// Create the object with no file or stream specified.
RDB ordb;
// Specify the stream via the RDB::open(ostream) method.
ordb.open( &cout );
// Or, specify the file via the RDB::open(string,ios::openmode) method.
ordb.open( "output.rdb", ios::out );
```

2.2 RDB header and table information

Information about the number of comments, columns, and rows are available using the RDB::nComments, RDB::nColumns(), and RDB::nRows methods. It is important to note that the first two are trivial queries, but determining the number of rows involves reading the entire table.

```
irdb.open( &cin );

for ( size_t idx = 0; idx < irdb.nColumns( ); idx++ ) {
  cout << idx << ":  " << irdb.getColumn( idx )->getName( ) << endl;

}

// This could take awhile...
cout << "There are " << irdb.nRows( ) << " rows" << endl;
```

2.3 Accessing comments and columns

An RDB object allows the user to add comments and columns as well as access existing comments and columns. When an RDB object is opened for reading, the RDB header is parsed for comments and columns. Any comment or column found is placed in the RDB object. When an RDB object is opened for writing, no comments or columns are present in the object.

To access an existing comment or column the user may reference the comment or column by index. Comments/Columns are indexed beginning at 0. If the user is requesting an index beyond the range of comments/columns contained in the RDB object, an RDBErrNotFound object is thrown or RDB::_errno is set. If the user is adding a comment/column and specifies an index beyond the range contained in the RDB object, the comment/column is added to the end of the list.

It is important to note that comments are returned as references to the underlying object while columns are returned as pointers to the underlying object. This is to allow columns of different datatypes to be stored in the same RDB object container.

```
// Foreach column...
for ( size_t idx = 0; idx < irdb.nColumns( ); idx++ ) {
  // Print the name and definition.
  cout << irdb.getColumn( idx )->getName( ) << " | "
       << irdb.getColumn( idx )->getDef( )  << endl;

       // Note the use of the '->', pointer dereference, operator.

}

// Set the error handling behavior to throw exceptions.
irdb.setThrow( true );
try {
  // Ask for on comment beyond what we have.
  irdb.getComment( irdb.nComments( ) );

       // NOTE:  the use of the '.', dot, operator.

} catch ( RDBErr& rdberr ) {
  // Catch the exception that will be thrown.
  cerr << "This will fail because comment indices start at 0." << endl;

}

// Here we check how many comments there are and then add one to
//   the end.
size_t ncomments_before = irdb.nComments( );
irdb.setComment( "Add one comment to the end..." );
if ( ncomments_before != irdb.nComments( ) ) {
  cout << "Now there's one more comment..." << endl;

}
```

It is also possible to access an existing comment/column or add a new comment/column by specifying the comment keyword/column name. Once, again if the comment/column is being added and no matching keyword/name is found, a new comment/column is added to the end of the list. If the comment/column is being returned and not matching keyword/name is found, an RDBErrNotFound object is thrown or RDB::_errno is set.

```
RDBColumn* col = irdb.getColumn( "x_err" );
```

Comments and columns are stored in RDBComment and RDBColumn objects respectively. The RDB object provides access to the RDBComment and RDBColumn objects it contains. For access to the constituent parts of either a comment or a column, the user may use the respective class interfaces.

```
// To access a column name, first get the column then ask it for
//   its name...
irdb.getColumn( 3 )->getName( );

// To reset a comment header variable, first get the comment then
//   set its value...
irdb.getComment( "foo" ).setValue( "Not bar" );
```

2.4 Table I/O

Table input and output are handled by the RDB::read and RDB::write methods. For input, RDB::read reads one row from the table, parsing it into the RDB object's columns. It checks to ensure the proper number of columns are found on each row,

2.5 Rewinding tables

With RDB objects, the user has the possibility of rewinding the RDB table. The object remebers the location of the first element of the first row or data. At any point, the object may be rewound, with the RDB::rewind method, to that point. RDBColumn::rewind is called on all columns associated with the RDB object.

It is important to note that if the RDB object was opened with a stream, rather than a filename, the stream must the ability to seek. Rewinding an RDB object opened with cin or cout will fail.

2.6 Automatic indexing for column data

2.7 Group columns

RDB objects use the RDB::newGroup method to monitor RDBColumns which have grouping activated. RDB::newGroup scans the RDBColumns associated with the object and calls RDBColumn::newGroup on each column. If any column returns true, RDB::newGroup returns true.

```
double sum;
int cnt;
string group;

irdb.getColumn( "break" )->setGroup( true );

while ( irdb.read( ) ) {
  if ( irdb.newGroup( ) ) {
    if ( irdb.getColumn( "_NR" )->getDataLong( ) ) {
      // Write out old group's stats...
      cout << "Avg for group " << group << " == " << sum / cnt << endl;

    }

    // Reset accumulators...
    sum   = 0.0;
    cnt   = 0;
    group = irdb.getColumn( "break" )->getDataString( );

  }

  // Accumulate statistics on other columns...
  sum += irdb.getColumn( "data" )->getDataDouble( );
  cnt++;

}
```

2.8 Creating RDB objects

RDB objects are created using any of the class constructors, RDB::RDB(const string&, const ios::open_mode), RDB::RDB(istream) or RDB::RDB(const RDB&). The filename constructor doubles as the default constructor for the class. If no filename is specifed, an object is created but no stream is attached to it. The RDB::open() method must be called to attach the object to a stream.

If the filename constructor is used with a filename specified, then the default is to open the file for reading. The user may explicitly set the open mode for output though.

The copy constructor does a shallow copy of the object provided as an argument. The new object is linked to the argument. Modification to the first objects RDBColumns will show up in the second objects RDBColumns.

2.9 Header and table size

Information about the number of comments, columns, rows are available using the RDB::nComments(), RDB::nColumns(), or RDB::nRows() methods.

2.10 Comment initializers

Various initialization functions are provided to set or add comments to an RDB. They methods allow the user to set or add entire RDBComment objects, or parts of RDBComments such as the keyword or value.

- RDB::setComment()
- RDB::setCommentText()
- RDB::setCommentKeyword()
- RDB::setCommentValue()

Each method allows the user to either specify the comment index, thereby replacing the existing RDBComment, or to leave out the index, thereby appending a new RDBComment to the end of the header.

2.10.1 Examples

Here an entire RDBComment object is added to the header.

```
RDBComment comment( "#: comment_variable = comment value" );
rdb.setComment( comment );
```

Here we overwrite the second comment with a new one.

```
rdb.setComment( comment, 2 );
```

2.11 Comment accessors

Comment accessors allow the user to retrieve all or part of the RDBComment. Once again there is a seperate method for each part of the RDBComment.

- RDB::getComment()
- RDB::getCommentText()
- RDB::getCommentKeyword()
- RDB::getCommentValue()

Each method has two signatures. The first signature allows the user to specify which comment to act upon by its index in the header. The second signature allows the user to specify which comment to act upon via keyword. It should be noted that if the comment does not explicityly contain a keyword = value construct no keyword is stored for the comment.

Each signature throws an exception if a comment matching either the index or keyword is not found. For functions specifying the index, an IndexOutOfRangeException is thrown. For the functions specifying the keyword, a KeyNotFoundException is thrown.

2.12 Column initializers

Various initialization functions are provided to set or add columns to an RDB. They methods allow the user to set or add entire RDBColumn objects, or parts of RDBColumns such as the name, definition, or value.

- RDB::setColumn()
- RDB::setColumnNameDef()
- RDB::setColumnName()
- RDB::setColumnDef()
- RDB::setColumnWidth()
- RDB::setColumnType()
- RDB::setColumnJust()

- RDB::setColumnDesc()
- RDB::setColumnValue()

Each method allows the user to either specify either the column index or the column name. The user may provide an entire RDBColumn object, as is the case with the RDB::setColumn() methods. If the index is out of range or the name is not the name of an existing column, the RDBColumn object is added to the RDB. Otherwise, the RDBColumn object replaces the existing object.

Each in turn throws an exception if the column is not found. In the case of columns specified by index, an IndexOutOfRangeException is thrown. In the cass of columns specified by name, a KeyNotFoundException is thrown. The methods which provide a column definition can also throw BadHeaderException if the definition is not of the proper form.

2.13 Column accessors

Column accessors allow the user to retrieve all or part of the RDBColumn. Once again there is a seperate method for each part of the RDBColumn.

- RDB::getColumn()
- RDB::getColumnNameDef()
- RDB::getColumnName()
- RDB::getColumnDef()
- RDB::getColumnWidth()
- RDB::getColumnType()
- RDB::getColumnJust()
- RDB::getColumnDesc()
- RDB::getColumnValue()

Each method allows the user to either specify either the column index or the column name.

Each in turn throws an exception if the column is not found. In the case of columns specified by index, an IndexOutOfRangeException is thrown. In the cass of columns specified by name, a KeyNotFoundException is thrown.

2.13.1 Examples

Here's a neat trick. You can link input and output tables by sharing RDBColumns. In this example the input is passed to the output. However, after linking the two RDB objects, the user doesn't have to explicitly set the values on the output table. The output will just use the values from the latest read from the input.

```
RDB irdb(&cin);
RDB ordb(&cout);

// This part takes a pointer to the column from input and gives it
//   to the output.  Now they point at the same RDBColumn...
for ( int i = 0; i < input.nColumns( ); i++ )
  ordb.setColumn( irdb.getColumn( i ) );

ordb.writeHeader( );

// Because the two tables share pointers to the same RDBColumns, a
//   call to read sets the values of the data in the output as well
//   as the input.  So now, a call to write will write the values
//   from the read.
while ( irdb.read( ) )
  ordb.write( );
```

3 Examples

example1.cc

example2.cc

example3.cc

The following example demonstrates the use of the clas simpleRDBTable. Given the following rdb file: hrc-size.cc

The rows of the rdb file can be initialized in the class ChipSize: ChipSize.h

ChipSize.cc ChipSize.cc

An example to invoke the simpleRDBTable class. testsimple.cc

4 Introduction

4.1 Purpose

The RDB library was created to handle input and output related tasks for RDB tables. The interface consists of multiple layers. The casual user may use the library through the RDB object interface exclusively. This interface allows the user to open, close, read from, and write to RDB tables. The interface handles provides default means of interpreting data types of the columns as well as default data storage management. Examples of basic table manipulations via the RDB object interface are available.

For some tasks the user may wish to modify the structure of a table or explicitly manage the location and memory allocation of the data. In these cases, the user will need to make use of the RDBColumn interface in addition to the RDB interface. Examples of more advance table manipulations using the RDBColumn interface are also available.

4.2 RDB format

RDB is an ASCII text format. An RDB table consists of three distinct elements:

- Comments (Comments are optional)
- Header (The header is mandatory)
- Data (The data are optional, though it would be a trivial table with no data...)

A comment is any line beginning zero or more spaces followed by a sharp sign (#). Characters following the sharp sign are considered to be the body of the comment. Comments may have additional structure within the body. It is possible to have comment variables, also known as header variables. A comment variable is said to exist if the sharp sign is immediately followed by a colon (:). The first word, non-whitespace and non-equal sign (=) characters, following the colon is the comment variable. The value of the comment variable is seperated from the variable by optional space (' ') and tab ('\t') characters and a mandatory equal sign. The variable value is then any text between the equal sign and the newline. It is common to restrict comments to the initial section of the file

The header consists of two lines. The first contains a tab seperated list of column names. Each column name is seperated by a single tab. The second line contains tab seperated column definitions. Definitions consist of an optional numeric value indicating the width, an S(tring), N(umeric), or M(onth) character denoting the type of the data, an optional < or > character indicating left or right justification of data, and an optional text description.

The data consist of tab seperated values, one per column name.

For a more definitive description of the format see `http://hea-www.harvard.edu/MST/simul/software/docs/html/perlrdb/rdb.html` .

5 Specialized Interface

5.1 Data storage

The default behavior of RDB object is to allocate space for data storage without pestering the user. When the object is used to read an RDB table, it parses the file and creates RDBComments for each comment line and RDBColumn of the appropriate datatype for each column.

5.1.1 Default data storage

When left to its own devices, an RDB object will manage memory usage on its own with no need for intervention from the user. This management essentially consists of allocating RDBComment objects and RDBColumn objects on demand and deleting them via the destructor.

Likewise, the underlying RDBColumn objects used by the RDB object handle their own memory needs by default. So, upon creation an RDBColumn object allocates a single element for data storage. RDBColumns handle the deletion of this data storage element in the call to their destructor.

Copies of the values can be accessed via the RDB::getValue() methods or the RDBColumn::getData() methods.

5.1.2 Examples

Using the default data storage and the RDB::getValue() method.

```
  while ( rdbtable.read( ) )
    cout << "Column 0 value == " << rdbtable.getValue( 0 ) << endl;
```

5.1.3 User-specified data storage

Sometimes the user may want to handle memory for data storage on their own. It is possible to supply the RDB object or RDBColumn a pointer to memory the user has alloacted via the RDBColumn::mapData() methods. By doing so the user can avoid calls to RDBColumn::getData() or RDB::getValue() and simply access the value via the pointer to the storage they provided.

Any memory the user provides via RDBColumn::mapData() must be managed, i.e. deleted, by the user.

By providing their own memory for data storage, the user can also make use of RDB object's ablity to auto-index arrays used for storage.

5.1.4 Examples

With user supplied data storage there's no need to use the RDB::getValue() method.

```
double d;
rdbtable.getColumn( 0 )->mapData( d );

while ( rdbtable.read( ) )
  cout << "Column 0 value == " << d << endl;
```

You can also store data from multiple rows.

```
double d_arr[10];
rdbtable.getColumn( 0 )->mapData( d, 10 );

int i = 0;
while ( rdbtable.read( ) ) {
  cout << "Column 0 current value  == " << d[i%10] << endl;
  cout << "Column 0 previous value == " << d[(i-1)%10] << endl;
  i++;
}
```

5.1.5 Auto-indexing

When the user supplies arrays of data storage via RDBColumn::mapData() RDB objects and RDBColumn will auto-index the arrays. As data is read or written, the default behavior is to increment the index into the user supplied array for data storage. It is possible then to store multiple rows of data.

RDB objects and RDBColumn increment the array index until the last element of the array provided by the user is reached. It then loops to the first element and begins over writing data.

It is possible to stop the auto-incrementing by calling the RDB::autoIdx() method. This temporarily turns off the auto-incrementing behavior until the next read or write.

5.1.6 Examples

In this example we want to store only the lines with negative data. So we use user supplied data storage and the RDB::autoIdx() method.

```
double* d = new double( rdbtable.nRows( ) );
rdbtable.getColumn( 0 )->mapData( d, rdbtable.nRows( ) );

int i = 0;
while ( rdbtable.read( ) ) {
  if ( 0 <= d[i] ) {
    rdbtable.noAdvance( );
  }
  else
    i++;
}
```

5.2 Stream manipulation

5.2.1 Effects on data storage

If the RDB object is attached to a file stream, it is possible to rewind the file to the first line of data. RDB::rewind() will move the file pointer back to the first data element. It also resets the array index for user supplied data storage. So, after a call to RDB::rewind(), all user supplied data storage will be pointing to the first element in the array.

5.2.2 Limitations

As noted above, RDB::rewind only works when the RDB object is attached to a stream that allows rewinding. RDB::rewind() will fail if the RDB is attached to STDIN or STDOUT.

6 Directory Hierarchy

6.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

7 Class Index

7.1 Class Hierarchy

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

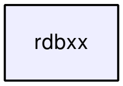
8 Class Index

8.1 Class List

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

9 Directory Documentation

9.1 rdbxx/ Directory Reference



Files

- file **RDB.cc**
- file **RDB.h**
- file **RDBColumn.cc**
- file **RDBColumn.h**
- file **RDBColumnTmplt.cc**
- file **RDBColumnTmplt.h**
- file **RDBComment.cc**
- file **RDBComment.h**
- file **RDBErr.cc**
- file **RDBErr.h**
- file **simpleRDBTable.cc**
- file **simpleRDBTable.h**

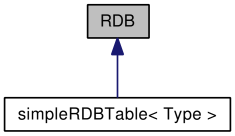
10 Class Documentation

10.1 RDB Class Reference

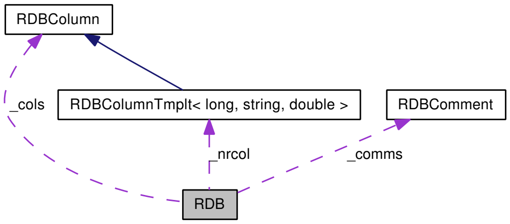
Provides interface for manipulating RDB tables.

`#include <RDB.h>`

Inheritance diagram for RDB:



Collaboration diagram for RDB:



Public Types

Enumerations for read/write/group return status.

- enum Status { **REOF** = 0x00, **REOL** = 0x01, **REOG** = 0x02, **RBOG** = 0x04 }

  Acceptable column justifications.

Public Member Functions

Constructing and destructing and initializing RDB objects.

- RDB (const string &name="", ios::openmode mode=ios::in) throw (RDBErr)

  Attaches RDB object to a file.

- RDB (istream ∗isptr) throw (RDBErr)

  Attaches RDB object to an istream.

- RDB (ostream ∗osptr) throw (RDBErr)

  Attaches RDB object to an ostream.

- RDB (const RDB &rdb) throw (RDBErr)

  Copies RDB object.

- ∼RDB (void)

  Deletes resources allocated by the RDB object.

I/O related operations.

- void open (const string &name, ios::openmode mode=ios::in) throw (RDBErr)

Attaches RDB object to a file.

- void open (istream ∗isptr) throw (RDBErr)
  Attaches RDB object to an istream.
- void open (ostream ∗osptr) throw (RDBErr)
  Attaches RDB object to an ostream.
- void open (const RDB &rdb) throw (RDBErr)
  Copies RDB object.
- void close (void)
  Closes the stream attached to RDB object.
- int read (void) throw (RDBErr)
  Read a line of data from the istream.
- bool write (void) throw (RDBErr)
  Write a line of data to the ostream.
- bool rewind (void) throw (RDBErr)
  Rewind the stream to the beginning of the first row of data.

Auto-indexing related methods.

- bool autoIdx (void) const
  Indicates if auto-indexing is activated.
- void autoIdx (const bool on)
  Activates/deactivates auto-indexing.
- void advanceIdx (void)
  Increments the indices in the RDBColumn data elements.

Column group manipulation (break columns)

- void setGroup (const string &name, bool group=true) throw (RDBErr)
  Turn on/off group status for the named column.
- void setGroup (const int idx, bool group=true) throw (RDBErr)
  Turn on/off group status for the indexed column.
- bool getGroup (const string &name) throw (RDBErr)
  Returns group status, true if its a new group, for the named column.

- bool getGroup (const int idx) throw (RDBErr)

  Returns group status, true if its a new group, for the indexed column.

- bool newGroup (void)

  Checks if any column indicates a new group.

Comment accessors.

- void setComment (const string &comm, const int idx=-1)

  Add RDBComment in header of RDB object.

- void setComment (RDBComment &comm, const int idx=-1)

  Add or replace RDBComment in header of RDB object.

- void setComment (RDBComment &comm, const string &name, const size_t idx=0)

  Add or replace RDBComment in header of RDB object.

- void setComment (const RDB &rdb)

  Copy all comments from an existing RDB object.

- RDBComment & getComment (const size_t idx) throw (RDBErr)

  Return RDBComment at given index.

- RDBComment & getComment (const string &name, const size_t idx=0) throw (RDBErr)

  Return RDBComment with given keyword.

Column accessors.

- void setColumn (const string &name, const string &def, const int idx=-1)

  Add an RDBColumn in RDB object.

- void setColumn (RDBColumn ∗col, const int idx=-1)

  Add or replace RDBColumn in RDB object.

- void setColumn (RDBColumn ∗col, const string &name, const size_t idx=0)

  Add of replace RDBColumn in RDB object.

- void setColumn (const RDB &rdb)

  Copy all columns from an existing RDB object.

- RDBColumn ∗ getColumn (const size_t idx) throw (RDBErr)

  Return pointer to RDBColumn at given index.

- RDBColumn ∗ getColumn (const string &name, const size_t idx=0) throw (RDBErr)

  Return pointer to RDBColumn with given name.

Column index based accessors.

- void setName (const size_t idx, const string &name) throw (RDBErr)

  Modify the name of the RDBColumn at idx.

- void setDef (const size_t idx, const string &def) throw (RDBErr)

  Modify the definition of the RDBColumn at idx.

- void setWidth (const size_t idx, const long width) throw (RDBErr)

  Modify the width of the RDBColumn at idx.

- void setType (const size_t idx, const RDBColumn::Type type) throw (RDBErr)

  Modify the type of the RDBColumn at idx.

- void setJust (const size_t idx, const RDBColumn::Just just) throw (RDBErr)

  Modify the justification of the RDBColumn at idx.

- void setDesc (const size_t idx, const string &desc) throw (RDBErr)

  Modify the description of the RDBColumn at idx.

- void mapData (const size_t idx, double data[ ], const size_t nelems=1) throw (RDBErr)

  Map RDBColumn data to user-supplied memory.

- void mapData (const size_t idx, long data[ ], const size_t nelems=1) throw (RDBErr)

  Map RDBColumn data to user-supplied memory.

- void mapData (const size_t idx, string data[ ], const size_t nelems=1) throw (RDBErr)

  Map RDBColumn data to user-supplied memory.

- void setData (const size_t idx, const double data) throw (RDBErr)

  Sets the data value of RDBColumn, converting as necessary.

- void setData (const size_t idx, const long data) throw (RDBErr)

  Sets the data value of RDBColumn, converting as necessary.

- void setData (const size_t idx, const string &data) throw (RDBErr)

  Sets the data value of RDBColumn, converting as necessary.

- void getName (const size_t idx, string &name) const throw (RDBErr)
  Return the name of the RDBColumn at idx.
- void getDef (const size_t idx, string &def) throw (RDBErr)
  Return the definition of the RDBColumn at idx.
- void getWidth (const size_t idx, long &width) const throw (RDBErr)
  Return the width of the RDBColumn at idx.
- void getType (const size_t idx, RDBColumn::Type &type) const throw (RDBErr)
  Return the type of the RDBColumn at idx.
- void getJust (const size_t idx, RDBColumn::Just &just) const throw (RDBErr)
  Return the just of the RDBColumn at idx.
- void getDesc (const size_t idx, string &desc) const throw (RDBErr)
  Return the description of the RDBColumn at idx.
- void getData (const size_t idx, double &data) throw (RDBErr)
  Return the data of the RDBColumn at idx, converting if necessary.
- void getData (const size_t idx, long &data) throw (RDBErr)
  Return the data of the RDBColumn at idx, converting if necessary.
- void getData (const size_t idx, string &data) throw (RDBErr)
  Return the data of the RDBColumn at idx, converting if necessary.
- string getName (const size_t idx) const throw (RDBErr)
  Return the name of the RDBColumn at idx.
- string getDef (const size_t idx) throw (RDBErr)
  Return the definition of the RDBColumn at idx.
- long getWidth (const size_t idx) const throw (RDBErr)
  Return the width of the RDBColumn at idx.
- RDBColumn::Type getType (const size_t idx) const throw (RDBErr)
  Return the type of the RDBColumn at idx.
- RDBColumn::Just getJust (const size_t idx) const throw (RDBErr)
  Return the justification of the RDBColumn at idx.
- string getDesc (const size_t idx) const throw (RDBErr)
  Return the description of the RDBColumn at idx.

- double getDataDouble (const size_t idx) throw (RDBErr)

  Return the data of the RDBColumn at idx, converting if necessary.

- long getDataLong (const size_t idx) throw (RDBErr)

  Return the data of the RDBColumn at idx, converting if necessary.

- string getDataString (const size_t idx) throw (RDBErr)

  Return the data of the RDBColumn at idx, converting if necessary.

Column name based accessors.

- void setName (const string &name, const string &newname) throw (RDBErr)

  Modify the RDBColumn name.

- void setDef (const string &name, const string &def) throw (RDBErr)

  Modify the RDBColumn definition.

- void setWidth (const string &name, const long width) throw (RDBErr)

  Modify the RDBColumn width.

- void setType (const string &name, const RDBColumn::Type type) throw (RDBErr)

  Modify the RDBColumn type.

- void setJust (const string &name, const RDBColumn::Just just) throw (RDBErr)

  Modify the RDBColumn justification.

- void setDesc (const string &name, const string &desc) throw (RDBErr)

  Modify the RDBColumn description.

- void mapData (const string &name, double data[ ], const size_t nelems=1) throw (RDBErr)

  Map RDBColum data to user-supplied memory.

- void mapData (const string &name, long data[ ], const size_t nelems=1) throw (RDBErr)

  Map RDBColum data to user-supplied memory.

- void mapData (const string &name, string data[ ], const size_t nelems=1) throw (RDBErr)

  Map RDBColum data to user-supplied memory.

- void setData (const string &name, const double data) throw (RDBErr)

Modify the RDBColumn data, converting if necessary.

- void setData (const string &name, const long data) throw (RDBErr)

  Modify the RDBColumn data, converting if necessary.

- void setData (const string &name, const string &data) throw (RDBErr)

  Modify the RDBColumn data, converting if necessary.

- void getName (const string &name, string &namefound) const throw (RDBErr)

  Return the name of the RDBColumn.

- void getDef (const string &name, string &def) throw (RDBErr)

  Return the definition of the RDBColumn.

- void getWidth (const string &name, long &width) const throw (RDBErr)

  Return the width of the RDBColumn.

- void getType (const string &name, RDBColumn::Type &type) const throw (RDBErr)

  Return the type of the RDBColumn.

- void getJust (const string &name, RDBColumn::Just &just) const throw (RDBErr)

  Return the justification of the RDBColumn.

- void getDesc (const string &name, string &desc) const throw (RDBErr)

  Return the description of the RDBColumn.

- void getData (const string &name, double &data) throw (RDBErr)

  Return the data of the RDBColumn, converting if necessary.

- void getData (const string &name, long &data) throw (RDBErr)

  Return the data of the RDBColumn, converting if necessary.

- void getData (const string &name, string &data) throw (RDBErr)

  Return the data of the RDBColumn, converting if necessary.

- string getName (const string &name) const throw (RDBErr)

  Return the name of the RDBColumn.

- string getDef (const string &name) throw (RDBErr)

  Return the definition of the RDBColumn.

- long getWidth (const string &name) const throw (RDBErr)

  Return the width of the RDBColumn.

- RDBColumn::Type getType (const string &name) const throw (RDBErr)

  Return the type of the RDBColumn.

- RDBColumn::Just getJust (const string &name) const throw (RDBErr)

  Return the justification of the RDBColumn.

- string getDesc (const string &name) const throw (RDBErr)

  Return the description of the RDBColumn.

- double getDataDouble (const string &name) throw (RDBErr)

  Return the data of the RDBColumn, converting if necessary.

- long getDataLong (const string &name) throw (RDBErr)

  Return the data of the RDBColumn, converting if necessary.

- string getDataString (const string &name) throw (RDBErr)

  Return the data of the RDBColumn, converting if necessary.

Table and header statistics.

- size_t nComments (void) const

  Return number of comments in RDB object.

- size_t nColumns (void) const

  Return number of columns in RDB object.

- size_t nRows (void)

  Return number of rows in RDB object.

Protected Member Functions

- void parseHeader (void) throw (RDBErr)

  Parse header, i.e. comments and column names and definitions.

- vector< string > parseLine (const string &line) const

  Parse fields in a row.

- size_t parseLine (bool &newgroup) throw (RDBErr)

  Parse fields in a row.

Protected Attributes

- string _filename

  Name of RDB file.

- ios::openmode _mode

  Open mode of the associated stream.

- istream ∗ _isptr

  Istream attached to data file.

- ostream ∗ _osptr

  Ostream attached to data file.

- bool _myisptr

  Indicates if RDB object is responsible for deallocating the istream.

- bool _myosptr

  Indicates if RDB object is responsible for deallocating the ostream.

- size_t _rewindto

  Position of beginning of first row of data.

- size_t _ncomms

  Number of comments.

- size_t _ncols

  Number of columns.

- size_t _nrows

  Number of rows.

- bool _knowrows

  Indicates if associated file must be scanned to determine number of rows.

- long _rownum

  Current ∗table∗ row number.

- long _frownum

  Current ∗file∗ row number.

- bool _autoidx

  Indicates if RDBColumn data elements should be advanced.

- bool _firstread

  Indicates if this is the first call to RDB::read.

- bool _lastread

  Indicates if this is the last call to RDB::read.

- bool _writehdr

  Indicates if the header has been output.

- RDBComment ∗ _comms

  Array of RDBComments.

- RDBColumn ∗∗ _cols

  Array of RDBColumns.

- RDBLongColumn _nrcol

  Hidden column, containing row number.

- bool ∗ _mycols

  Indicates if RDB object is responsible for deallocating given RDBColumn.

- string _line

  Line from RDB table.

Friends

Stream insertion and extraction operators.

- istream & operator>> (istream &is, RDB &rdb) throw (RDBErr)
  Read table from input stream.

- ostream & operator<< (ostream &os, RDB &rdb) throw (RDBErr)
  Write table to output stream.

10.1.1 Detailed Description

Provides interface for manipulating RDB tables.

Definition at line 43 of file RDB.h.

10.1.2 Member Enumeration Documentation

10.1.2.1 enum RDB::Status

Acceptable column justifications.

Definition at line 56 of file RDB.h.

10.1.3 Constructor & Destructor Documentation

10.1.3.1 RDB::RDB (const string & *name* = `""`, ios::openmode *mode* = `ios::in`) throw (RDBErr)

Attaches RDB object to a file.

Parameters:

- ***name*** the name of the RDB file.
- ***mode*** the ios::openmode of the file.

Exceptions:

- ***RDBErr*** error opening RDB file.
- ***RDBErr*** error parsing RDB comment or column name and definition.
- ***RDBErr*** error parsing RDB column definition.

Attaches RDB object's I/O stream to the file. If mode is set to ios::in, it calls RDB::parseHeader(void) to read comments, column names, and column definitions.

If no filename is specified, then the RDB::open() method may be used later to attach a file to the object.

Definition at line 118 of file RDB.cc.

10.1.3.2 RDB::RDB (istream $*$ *isptr*) throw (RDBErr)

Attaches RDB object to an istream.

Parameters:

- ***isptr*** input stream to attach to the RDB object.

Exceptions:

- ***RDBErr*** error parsing RDB comment or column name and definition.
- ***RDBErr*** error parsing RDB column definition.

Attaches the istream to the RDB object. RDB::parseHeader(void) is called to read comments, column names, and column definitions.

Definition at line 164 of file RDB.cc.

10.1.3.3 RDB::RDB (ostream $*$ *osptr*) throw (RDBErr)

Attaches RDB object to an ostream.

Parameters:

osptr output stream to attach to the RDB object.

Attaches the ostream to the RDB object.

Definition at line 199 of file RDB.cc.

10.1.3.4 RDB::RDB (const RDB & *rdb*) throw (RDBErr)

Copies RDB object.

Parameters:

rdb copy RDB object.

Makes a copy of the argument.

Definition at line 234 of file RDB.cc.

10.1.3.5 RDB::∼RDB (void)

Deletes resources allocated by the RDB object.

Deletes RDBComments and RDBColumns. Closes only the streams the object opened. User is responsible for closing streams they supply to the object.

Warning:

If the user supplies an RDBColumn object via the RDB::setColumn() method, the user is responsible for deleting that object.

Definition at line 271 of file RDB.cc.

References _cols, _comms, _isptr, _mycols, _ncols, _osptr, and close().

10.1.4 Member Function Documentation

10.1.4.1 void RDB::open (const string & *name*, ios::openmode *mode* = `ios::in`) throw (RDBErr)

Attaches RDB object to a file.

Parameters:

name the name of the RDB file.

mode the ios::openmode of the file.

Exceptions:

RDBErr error opening RDB file.

RDBErr error parsing RDB comment or column name and definition.

RDBErr error parsing RDB column definition.

Attaches RDB object's I/O stream to the file. If mode is set to ios::in, it calls RDB::parseHeader(void) to read comments, column names, and column definitions.

Warning:

This method closes any previously opened streams.

Definition at line 316 of file RDB.cc.

References _filename, _isptr, _mode, _myisptr, _myosptr, _osptr, close(), and parseHeader().

10.1.4.2 void RDB::open (istream ∗ *isptr*) throw (RDBErr)

Attaches RDB object to an istream.

Parameters:

isptr input stream to attach to the RDB object.

Exceptions:

RDBErr error parsing RDB comments or column names and definitions

RDBErr error parsing RDB column definition.

Attaches the istream to the RDB object. RDB::parseHeader(void) is called to read comments, column names, and column definitions.

Warning:

This method closes any previously opened streams.

Definition at line 400 of file RDB.cc.

References _filename, _isptr, _mode, close(), and parseHeader().

10.1.4.3 void RDB::open (ostream ∗ *osptr*) throw (RDBErr)

Attaches RDB object to an ostream.

Parameters:

osptr output stream to attach to the RDB object.

Attaches the ostream to the RDB object.

Warning:

This method closes any previously opened streams.

Definition at line 432 of file RDB.cc.

References _filename, _mode, _osptr, and close().

10.1.4.4 void RDB::open (const RDB & *rdb*) throw (RDBErr)

Copies RDB object.

Parameters:

rdb copy RDB object.

Warning:

This method is currently not implemented.

Definition at line 451 of file RDB.cc.

References _filename, _isptr, _mode, _myisptr, _myosptr, _osptr, close(), and parseHeader().

10.1.4.5 void RDB::close (void)

Closes the stream attached to RDB object.

Closes any streams that the RDB object created.

Definition at line 526 of file RDB.cc.

References _autoidx, _cols, _comms, _filename, _isptr, _knowrows, _myisptr, _myosptr, _ncols, _ncomms, _nrows, _osptr, _rewindto, _rownum, _writehdr, RDBColumn::getDef(), getDef(), RDBColumn::getName(), and getName().

Referenced by open(), and ∼RDB().

10.1.4.6 int RDB::read (void) throw (RDBErr)

Read a line of data from the istream.

Exceptions:

- ***RDBErr*** error if the number of tokens found does not match the number of tokens expected.
- ***RDBErr*** error if non-numeric data is found in a numeric column.
- ***RDBErr*** error if a floating point number is being assigned to an integer column.

S ∗

Returns:

RDB::Status indicating if end of file, end of line, or end of group was reached.

Reads, parses, and stores data from the next line in the RDB file is an input stream has been provided. If RDB::_autoidx is set, this method will advance the indices in each RDBColumn's data pointer before reading and will increment the row number, RDB::_rownum, for the default '_NR' column.

Definition at line 589 of file RDB.cc.

References _autoidx, _cols, _filename, _firstread, _isptr, _lastread, _line, _mode, _ncols, _rownum, advanceIdx(), RDBColumn::advanceIdx(), RDBColumn::newGroup(), and RDBColumn::setData().

Referenced by simpleRDBTable< Type >::readRow().

10.1.4.7 bool RDB::write (void) throw (RDBErr)

Write a line of data to the ostream.

Returns:

false if EOF or if the ostream is bad.

Writes the next line of data to the RDB file, if an output stream has been provided. If the RDB header has not been written yet, it outputs the comments, column names and column definitions. If RDB::_autoidx is set, it will advance the indices in the RDBColumn's data pointer before reading and will increment the row number, RDB::_rownum, for the default '_NR' column.

After a successful write, the RDB::_autoidx flag is set to true.

Definition at line 712 of file RDB.cc.

References _autoidx, _cols, _comms, _mode, _ncols, _ncomms, _osptr, _rewindto, _rownum, _writehdr, advanceIdx(), RDBColumn::getDef(), getDef(), RDBColumn::getName(), and getName().

10.1.4.8 bool RDB::rewind (void) throw (RDBErr)

Rewind the stream to the beginning of the first row of data.

Returns:

> false if the stream associated with this RDB object cannot be rewound, i.e seekg() or seekp() is not defined.

Rewinds the stream associate with this object to the beginning of the first row of data. It also calls RDBColumn::rewind(void) for each RDBColumn. If the user supplied data storage to a particular column, the auto-incrementing index is rewound to point at the first element of the user supplied array.

Warning:

> If an istream or ostream, like cin or cout, was provided to the object, you only get one chance to read it. You can't rewind, you can't close and reopen.

Definition at line 772 of file RDB.cc.

References _cols, _isptr, _mode, _ncols, _osptr, _rewindto, _rownum, and RDBColumn::rewind().

10.1.4.9 bool RDB::autoIdx (void) const

Indicates if auto-indexing is activated.

Returns:

> State of the auto-indexing flag for this object.

Definition at line 843 of file RDB.cc.

References _autoidx.

10.1.4.10 void RDB::autoIdx (const bool *on*)

Activates/deactivates auto-indexing.

Parameters:

> ***on*** sets the auto-indexing behavior for the RDB object and its columns.

Sets RDBTable::_autoidx. This effectly turns off the auto-incrementing behavior for a single call to RDB::read(void) or RDB::write(void). For arrays of mapped memory, this will cause one line's data to overwrite the previous line's data.

Warning:

> RDB::_autoidx is set to true after each call to RDB::read(void) or RDB::write(void).

Definition at line 830 of file RDB.cc.

References _autoidx.

10.1.4.11 void RDB::advanceIdx (void)

Increments the indices in the RDBColumn data elements.

This method advances the auto-indices for all RDBColumns associated with this object.

Definition at line 857 of file RDB.cc.

References _cols, _ncols, _nrcol, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::advanceIdx(), and RDBColumn::advanceIdx().

Referenced by read(), and write().

10.1.4.12 void RDB::setGroup (const string & *name*, bool *group* = `true`) throw (RDBErr)

Turn on/off group status for the named column.

Parameters:

group	a bool indicating whether group is on or off.
name	the name of the column in this RDB object.

Exceptions:

RDBEr	error if there is no column with matching name.

Definition at line 878 of file RDB.cc.

References _autoidx, _cols, _filename, _ncols, _nrcol, getName(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroup(), and RDBColumn::setGroup().

10.1.4.13 void RDB::setGroup (const int *idx*, bool *group* = `true`) throw (RDBErr)

Turn on/off group status for the indexed column.

Parameters:

group	a bool indicating whether group is on or off.

idx the position of the column on which to operate.

Exceptions:

RDBErr error if there is no column with matching name.

Definition at line 911 of file RDB.cc.

References _autoidx, _cols, _filename, _ncols, and RDBColumn::setGroup().

10.1.4.14 bool RDB::getGroup (const string & *name*) throw (RDBErr)

Returns group status, true if its a new group, for the named column.

Parameters:

name the name of the column.

Exceptions:

RDBErrNotFnd error if there is no column with matching name.

Returns:

Status of the column's group flag.

Definition at line 938 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getGroup(), and getName().

10.1.4.15 bool RDB::getGroup (const int *idx*) throw (RDBErr)

Returns group status, true if its a new group, for the indexed column.

Parameters:

idx the column index.

Exceptions:

RDBErr error if the index is out of range.

Returns:

Status of the column's group flag.

Definition at line 967 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getGroup().

10.1.4.16 bool RDB::newGroup (void)

Checks if any column indicates a new group.

Returns:

True if any column indicates a new group, false otherwise.

A new group occurs when the data values between consecutive rows change.

Definition at line 989 of file RDB.cc.

References _cols, and _ncols.

Referenced by parseLine().

10.1.4.17 void RDB::setComment (const string & *comm*, const int *idx* = `-1`)

Add RDBComment in header of RDB object.

Parameters:

comm a comment string.
idx the position of the comment on which to operate.

Comments are numbered as they appear in the RDB file from top to bottom. If the index is not specified or does not fall within the range of existing comments, the RDBComment is appended to the list of comments.

Definition at line 1015 of file RDB.cc.

References _comms, and _ncomms.

Referenced by parseHeader(), and setComment().

10.1.4.18 void RDB::setComment (RDBComment & *comm*, const int *idx* = `-1`)

Add or replace RDBComment in header of RDB object.

Parameters:

comm a reference to the RDBComment object
idx the position of the comment on which to operate.

Comments are numbered as they appear in the RDB file from top to bottom. If the index is not specified or does not fall within the range of existing comments, the RDBComment is appended to the list of comments.

Definition at line 1053 of file RDB.cc.

References _comms, and _ncomms.

10.1.4.19 void RDB::setComment (RDBComment & *comm*, const string & *name*, const size_t *idx* = `0`)

Add or replace RDBComment in header of RDB object.

Parameters:

- ***comm*** a reference to the RDBComment object
- ***name*** the name of the comment keyword.
- ***idx*** currently not defined.

If no matching comment keyword is found, the comment is appended to the list of comments. Otherwise, comm replaces the existing comment.

Definition at line 1090 of file RDB.cc.

References _comms, _ncomms, and setComment().

10.1.4.20 void RDB::setComment (const RDB & *rdb*)

Copy all comments from an existing RDB object.

Parameters:

- ***rdb*** a reference to an RDB object

Copies all comments from the argument to this RDB object. Any comments previously associated with this object are removed.

Definition at line 1116 of file RDB.cc.

References _comms, and _ncomms.

10.1.4.21 RDBComment & RDB::getComment (const size_t *idx*) throw (RDBErr)

Return RDBComment at given index.

Parameters:

- ***idx*** the index of the RDBComment.

Exceptions:

- ***RDBErrNot*** error if the index is out of range.

Returns:

RDBComment with index equal to idx.

RDBComments are indexed in the order in which they appear in the RDB file starting with 0. Parameter idx must be between 0 and RDB::nComments(void) less one.

Definition at line 1148 of file RDB.cc.

References _comms, _filename, and _ncomms.

10.1.4.22 RDBComment & RDB::getComment (const string & *name*, const size_t *idx* = `0`) throw (RDBErr)

Return RDBComment with given keyword.

Parameters:

name the name of the comment keyword.

idx currently not defined.

Exceptions:

RDBErr error if there is no comment with matching keyword.

Returns:

RDBComment with matching keyword.

Definition at line 1173 of file RDB.cc.

References _comms, _filename, and _ncomms.

10.1.4.23 void RDB::setColumn (const string & *name*, const string & *def*, const int *idx* = `-1`)

Add an RDBColumn in RDB object.

Parameters:

name of the RDBColumn object

def definition of the RDBColumn object

idx the position of the column on which to operate.

Columns are numbered as they appear in the RDB file from left to right starting with 0. If the index is not specified or does not fall within the range of existing columns, an RDBColumn is created and is appended to the list of columns. The RDB object handles memory allocation and deletion.

Upon completion, the RDB object contains a pointer to the RDBColumn created. Any modifications to the RDBColumn outside of the RDB object will be reflected within the RDB object and vice versa.

Definition at line 1208 of file RDB.cc.

References _cols, _mycols, _ncols, RDBColumn::setDef(), and RDBColumn::setName().

Referenced by setColumn().

10.1.4.24 void RDB::setColumn (RDBColumn ∗ *col*, const int *idx* = `-1`)

Add or replace RDBColumn in RDB object.

Parameters:

col a reference to the RDBColumn object

idx the position of the column on which to operate.

Columns are numbered as they appear in the RDB file from left to right starting with 0. If the index is not specified or does not fall within the range of existing columns, the RDBColumn is appended to the list of columns.

Upon completion, the RDB object contains a pointer to the RDBColumn provided. Any modifications to the RDBColumn outside of the RDB object will be reflected within the RDB object and vice versa.

Warning:

The user is responsible for freeing the RDBColumn.

Definition at line 1274 of file RDB.cc.

References _cols, _mycols, and _ncols.

10.1.4.25 void RDB::setColumn (RDBColumn ∗ *col*, const string & *name*, const size_t *idx* = `0`)

Add of replace RDBColumn in RDB object.

Parameters:

col a reference to the RDBColumn object

name the name of the column in this RDB object.

idx currently not defined.

If no matching column is found, the column is appended to the list of columns after assigning it a name of 'name'. Otherwise, col replaces the existing column.

Upon completion, the RDB object contains a pointer to the RDBColumn provided. Any modifications to the RDBColumn outside of the RDB object will be reflected within the RDB object and vice versa.

Warning:

> The user is responsible for freeing the RDBColumn.

Definition at line 1331 of file RDB.cc.

References _cols, _mycols, _ncols, getName(), setColumn(), and RDBColumn::setName().

10.1.4.26 void RDB::setColumn (const RDB & *rdb*)

Copy all columns from an existing RDB object.

Parameters:

> ***rdb*** a reference to an RDB object

Copies all columns from the argument to this RDB object. Any columns previously associated with this object are removed.

Warning:

> The user is responsible for free any RDBColumns previously allocated and assigned to this RDB object via the RDB::setColumn methods.

Definition at line 1368 of file RDB.cc.

References _cols, _mycols, and _ncols.

10.1.4.27 RDBColumn ∗ RDB::getColumn (const size_t *idx*) throw (RDBErr)

Return pointer to RDBColumn at given index.

Parameters:

> ***idx*** the index of the RDBColumn.

Exceptions:

> ***RDBErr*** error if the index is out of range.

Returns:

> RDBColumn with index equal to idx.

RDBColumns are indexed in the order in which they appear in the RDB file starting with 0. Parameter idx must be between 0 and RDB::nColumns(void) less one.

Definition at line 1433 of file RDB.cc.

References _cols, _filename, and _ncols.

10.1.4.28 RDBColumn ∗ RDB::getColumn (const string & *name*, const size_t *idx* = `0`) throw (RDBErr)

Return pointer to RDBColumn with given name.

Parameters:

name the name of the column.

idx currently not defined.

Exceptions:

RDBErrNotFnd error if there is no column with matching name.

Returns:

RDBColumn with matching name.

Definition at line 1458 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, and getName().

10.1.4.29 void RDB::setName (const size_t *idx*, const string & *name*) throw (RDBErr)

Modify the name of the RDBColumn at idx.

Parameters:

idx index of RDBColumn.

name to assign to RDBColumn at idx.

Exceptions:

RDBErr if index is out of range.

Definition at line 1488 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setName().

10.1.4.30 void RDB::setDef (const size_t *idx*, const string & *def*) throw (RDBErr)

Modify the definition of the RDBColumn at idx.

Parameters:

idx index of RDBColumn.

def RDBColumn definition.

Exceptions:

RDBErr if index is out of range.
RDBErr if def is invalid RDBColumn definition.

Definition at line 1509 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setDef().

10.1.4.31 void RDB::setWidth (const size_t *idx*, const long *width*) throw (RDBErr)

Modify the width of the RDBColumn at idx.

Parameters:

idx index of RDBColumn.
width RDBColumn width.

Exceptions:

RDBErr if index is out of range.
RDBErr if width is invalid RDBColumn width.

Definition at line 1537 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setWidth().

10.1.4.32 void RDB::setType (const size_t *idx*, const RDBColumn::Type *type*) throw (RDBErr)

Modify the type of the RDBColumn at idx.

Parameters:

idx index of RDBColumn.
type RDBColumn type.

Exceptions:

RDBErr if index is out of range.
RDBErr if type is invalid RDBColumn type.

Definition at line 1565 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setType().

10.1.4.33 void RDB::setJust (const size_t *idx*, const RDBColumn::Just *just*) throw (RDBErr)

Modify the justification of the RDBColumn at idx.

Parameters:

idx index of RDBColumn.

just RDBColumn justification.

Exceptions:

RDBErr if index is out of range.

RDBErr if just is invalid RDBColumn justification.

Definition at line 1593 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setJust().

10.1.4.34 void RDB::setDesc (const size_t *idx*, const string & *desc*) throw (RDBErr)

Modify the description of the RDBColumn at idx.

Parameters:

idx index of RDBColumn.

desc RDBColumn description.

Exceptions:

RDBErr if index is out of range.

RDBErr if desc is invalid RDBColumn description.

Definition at line 1621 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setDesc().

10.1.4.35 void RDB::mapData (const size_t *idx*, double *data*[ ], const size_t *nelems* = `1`) throw (RDBErr)

Map RDBColumn data to user-supplied memory.

Parameters:

idx index of RDBColumn.

data user-supplied memory for storing data.

nelems number of elements in user-supplied data.

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Definition at line 1650 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::mapData().

10.1.4.36 void RDB::mapData (const size_t *idx*, long *data*[ ], const size_t *nelems* = `1`) throw (RDBErr)

Map RDBColumn data to user-supplied memory.

Parameters:

idx index of RDBColumn.

data user-supplied memorgy to stored data.

nelems number of elems in user-supplied memory.

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Definition at line 1678 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::mapData().

10.1.4.37 void RDB::mapData (const size_t *idx*, string *data*[ ], const size_t *nelems* = `1`) throw (RDBErr)

Map RDBColumn data to user-supplied memory.

Parameters:

idx index of RDBColumn.

data user-supplied memory to store data.

nelems number of elements in user-supplied data.

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Definition at line 1707 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::mapData().

10.1.4.38 void RDB::setData (const size_t *idx*, const double *data*) throw (RDBErr)

Sets the data value of RDBColumn, converting as necessary.

Parameters:

idx index of RDBColumn.

data value to assign to RDBColumn data.

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid for RDBColumn.

Definition at line 1735 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setData().

10.1.4.39 void RDB::setData (const size_t *idx*, const long *data*) throw (RDBErr)

Sets the data value of RDBColumn, converting as necessary.

Parameters:

idx index of RDBColumn.

data value to assign to RDBColumn data.

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid for RDBColumn.

Definition at line 1763 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setData().

10.1.4.40 void RDB::setData (const size_t *idx*, const string & *data*) throw (RDBErr)

Sets the data value of RDBColumn, converting as necessary.

Parameters:

- ***idx*** index of RDBColumn/
- ***data*** value to assign RDBColumn data.

Exceptions:

- ***RDBErr*** if index is out of range.
- ***RDBErr*** if data is invalid for RDBColumn.

Definition at line 1791 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::setData().

10.1.4.41 void RDB::getName (const size_t *idx*, string & *name*) const throw (RDBErr)

Return the name of the RDBColumn at idx.

Parameters:

- ***idx*** index of RDBColumn.
- ***name*** retrieve name of RDBColumn.

Exceptions:

- ***RDBErr*** if index is out of range.

Definition at line 1818 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getName().

Referenced by close(), getColumn(), getData(), getDataDouble(), getDataLong(), getDataString(), getDef(), getDesc(), getGroup(), getJust(), getName(), getType(), getWidth(), mapData(), setColumn(), setData(), setDef(), setDesc(), setGroup(), setJust(), setName(), setType(), setWidth(), and write().

10.1.4.42 void RDB::getDef (const size_t *idx*, string & *def*) throw (RDBErr)

Return the definition of the RDBColumn at idx.

Parameters:

- ***idx*** index of RDBColumn.

def retrieve definition of RDBColumn.

Exceptions:

RDBErr if index is out of range.

Definition at line 1837 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDef().

Referenced by close(), and write().

10.1.4.43 void RDB::getWidth (const size_t *idx*, long & *width*) const throw (RDBErr)

Return the width of the RDBColumn at idx.

Parameters:

idx index of RDBColumn.

width retrieve width of RDBColumn.

Exceptions:

RDBErr if index is out of range.

Definition at line 1856 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getWidth().

10.1.4.44 void RDB::getType (const size_t *idx*, RDBColumn::Type & *type*) const throw (RDBErr)

Return the type of the RDBColumn at idx.

Parameters:

idx index of RDBColumn.

type retrieve type of RDBColumn.

Exceptions:

RDBErr if index is out of range.

Definition at line 1875 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getType().

10.1.4.45 void RDB::getJust (const size_t *idx*, RDBColumn::Just & *just*) const throw (RDBErr)

Return the just of the RDBColumn at idx.

Parameters:

- ***idx*** index of RDBColumn
- ***just*** retrieve just of RDBColumn.

Exceptions:

- ***RDBErr*** if index is out of range.

Definition at line 1894 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getJust().

10.1.4.46 void RDB::getDesc (const size_t *idx*, string & *desc*) const throw (RDBErr)

Return the description of the RDBColumn at idx.

Parameters:

- ***idx*** index of RDBColumn
- ***desc*** retrieve description of RDBColumn.

Exceptions:

- ***RDBErr*** if index is out of range.

Definition at line 1913 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDesc().

10.1.4.47 void RDB::getData (const size_t *idx*, double & *data*) throw (RDBErr)

Return the data of the RDBColumn at idx, converting if necessary.

Parameters:

- ***idx*** index of RDBColumn
- ***data*** retrieve data of RDBColumn.

Exceptions:

- ***RDBErr*** if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Definition at line 1933 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDataDouble().

10.1.4.48 void RDB::getData (const size_t *idx*, long & *data*) throw (RDBErr)

Return the data of the RDBColumn at idx, converting if necessary.

Parameters:

idx index of RDBColumn

data retrieve data of RDBColumn.

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Definition at line 1961 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDataLong().

10.1.4.49 void RDB::getData (const size_t *idx*, string & *data*) throw (RDBErr)

Return the data of the RDBColumn at idx, converting if necessary.

Parameters:

idx index of RDBColumn

data retrieve data of RDBColumn.

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Definition at line 1989 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDataString().

10.1.4.50 string RDB::getName (const size_t *idx*) const throw (RDBErr)

Return the name of the RDBColumn at idx.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

Returns:

name of RDBColumn.

Definition at line 2017 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getName().

10.1.4.51 string RDB::getDef (const size_t *idx*) throw (RDBErr)

Return the definition of the RDBColumn at idx.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

Returns:

definition of RDBColumn.

Definition at line 2037 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDef().

10.1.4.52 long RDB::getWidth (const size_t *idx*) const throw (RDBErr)

Return the width of the RDBColumn at idx.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

Returns:

width of RDBColumn.

Definition at line 2057 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getWidth().

10.1.4.53 RDBColumn::Type RDB::getType (const size_t *idx*) const throw (RDBErr)

Return the type of the RDBColumn at idx.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

Returns:

type of RDBColumn.

Definition at line 2077 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getType().

10.1.4.54 RDBColumn::Just RDB::getJust (const size_t *idx*) const throw (RDBErr)

Return the justification of the RDBColumn at idx.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

Returns:

justification of RDBColumn.

Definition at line 2097 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getJust().

10.1.4.55 string RDB::getDesc (const size_t *idx*) const throw (RDBErr)

Return the description of the RDBColumn at idx.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

Returns:

description of RDBColumn.

Definition at line 2117 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDesc().

10.1.4.56 double RDB::getDataDouble (const size_t *idx*) throw (RDBErr)

Return the data of the RDBColumn at idx, converting if necessary.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Returns:

data of RDBColumn.

Definition at line 2138 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDataDouble().

10.1.4.57 long RDB::getDataLong (const size_t *idx*) throw (RDBErr)

Return the data of the RDBColumn at idx, converting if necessary.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Returns:

Data of RDBColumn.

Definition at line 2167 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDataLong().

10.1.4.58 string RDB::getDataString (const size_t *idx*) throw (RDBErr)

Return the data of the RDBColumn at idx, converting if necessary.

Parameters:

idx index of RDBColumn

Exceptions:

RDBErr if index is out of range.

RDBErr if data is invalid datatype for RDBColumn.

Returns:

data of RDBColumn.

Definition at line 2196 of file RDB.cc.

References _cols, _filename, _ncols, and RDBColumn::getDataString().

10.1.4.59 void RDB::setName (const string & *name*, const string & *newname*) throw (RDBErr)

Modify the RDBColumn name.

Parameters:

name of RDBColumn

newname new name of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.

RDBErr if newname is invalid name for RDBColumn.

Definition at line 2224 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::setName().

10.1.4.60 void RDB::setDef (const string & *name*, const string & *def*) throw (RDBErr)

Modify the RDBColumn definition.

Parameters:

- ***name*** of RDBColumn
- ***def*** definition of RDBColumn.

Exceptions:

- ***RDBErr*** if no matching column name is found.
- ***RDBErr*** if def is invalid definition for RDBColumn.

Definition at line 2250 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::setDef().

10.1.4.61 void RDB::setWidth (const string & *name*, const long *width*) throw (RDBErr)

Modify the RDBColumn width.

Parameters:

- ***name*** of RDBColumn
- ***width*** width of RDBColumn.

Exceptions:

- ***RDBErr*** if no matching column name is found.
- ***RDBErr*** if width is invalid width for RDBColumn.

Definition at line 2285 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::setWidth().

10.1.4.62 void RDB::setType (const string & *name*, const RDBColumn::Type *type*) throw (RDBErr)

Modify the RDBColumn type.

Parameters:

name of RDBColumn

type type of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.

RDBErr if type is invalid type for RDBColumn.

Definition at line 2320 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::setType().

10.1.4.63 void RDB::setJust (const string & *name*, const RDBColumn::Just *just*) throw (RDBErr)

Modify the RDBColumn justification.

Parameters:

name of RDBColumn

just justification of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.

RDBErr if just is invalid justification for RDBColumn.

Definition at line 2355 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::setJust().

10.1.4.64 void RDB::setDesc (const string & *name*, const string & *desc*) throw (RDBErr)

Modify the RDBColumn description.

Parameters:

name of RDBColumn

desc description of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.
RDBErr if desc is invalid description for RDBColumn.

Definition at line 2390 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::setDesc().

10.1.4.65 void RDB::mapData (const string & *name*, double *data*[ ], const size_t *nelems* = `1`) throw (RDBErr)

Map RDBColum data to user-supplied memory.

Parameters:

name of RDBColumn
data user-supplied memory to store RDBColumn data.
nelems number of elems in user-supplied memory.

Exceptions:

RDBErr if no matching column name is found.
RDBErr if data is invalid datatype for RDBColumn.

Definition at line 2426 of file RDB.cc.

References _cols, _filename, _ncols, getName(), and RDBColumn::mapData().

10.1.4.66 void RDB::mapData (const string & *name*, long *data*[ ], const size_t *nelems* = `1`) throw (RDBErr)

Map RDBColum data to user-supplied memory.

Parameters:

name of RDBColumn
data user-supplied memory to store RDBColumn data.
nelems number of elems in user-supplied memory.

Exceptions:

RDBErr if no matching column name is found.
RDBErr if data is invalid datatype for RDBColumn.

Definition at line 2462 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::mapData(), and RDBColumn::mapData().

10.1.4.67 void RDB::mapData (const string & *name*, string *data*[ ], const size_t *nelems* = `1`) throw (RDBErr)

Map RDBColum data to user-supplied memory.

Parameters:

- ***name*** of RDBColumn
- ***data*** user-supplied memory to store RDBColumn data.
- ***nelems*** number of elems in user-supplied memory.

Exceptions:

- ***RDBErr*** if no matching column name is found.
- ***RDBErr*** if data is invalid datatype for RDBColumn.

Definition at line 2499 of file RDB.cc.

References _cols, _filename, _ncols, getName(), and RDBColumn::mapData().

10.1.4.68 void RDB::setData (const string & *name*, const double *data*) throw (RDBErr)

Modify the RDBColumn data, converting if necessary.

Parameters:

- ***name*** of RDBColumn
- ***data*** value to assign RDBColumn data.

Exceptions:

- ***RDBErr*** if no matching column name is found.
- ***RDBErr*** if data is invalid datatype for RDBColumn.

Definition at line 2534 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData(), and RDBColumn::setData().

10.1.4.69 void RDB::setData (const string & *name*, const long *data*) throw (RDBErr)

Modify the RDBColumn data, converting if necessary.

Parameters:

- ***name*** of RDBColumn

data value to assign RDBColumn data.

Exceptions:

RDBErr if no matching column name is found.
RDBErr if data is invalid datatype for RDBColumn.

Definition at line 2567 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData(), and RDBColumn::setData().

10.1.4.70 void RDB::setData (const string & *name*, const string & *data*) throw (RDBErr)

Modify the RDBColumn data, converting if necessary.

Parameters:

name of RDBColumn
data value to assign RDBColumn data.

Exceptions:

RDBErr if no matching column name is found.
RDBErr if data is invalid datatype for RDBColumn.

Definition at line 2600 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData(), and RDBColumn::setData().

10.1.4.71 void RDB::getName (const string & *name*, string & *namefound*) const throw (RDBErr)

Return the name of the RDBColumn.

Parameters:

name of RDBColumn
namefound name of RDBColumn if found.

Exceptions:

RDBErr if no matching column name is found.

Definition at line 2633 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getName(), and getName().

10.1.4.72 void RDB::getDef (const string & *name*, string & *def*) throw (RDBErr)

Return the definition of the RDBColumn.

Parameters:

- ***name*** of RDBColumn
- ***def*** definition of RDBColumn.

Exceptions:

- ***RDBErr*** if no matching column name is found.

Definition at line 2657 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getDef(), and getName().

10.1.4.73 void RDB::getWidth (const string & *name*, long & *width*) const throw (RDBErr)

Return the width of the RDBColumn.

Parameters:

- ***name*** of RDBColumn
- ***width*** width of RDBColumn.

Exceptions:

- ***RDBErr*** if no matching column name is found.

Definition at line 2681 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::getWidth().

10.1.4.74 void RDB::getType (const string & *name*, RDBColumn::Type & *type*) const throw (RDBErr)

Return the type of the RDBColumn.

Parameters:

- ***name*** of RDBColumn
- ***type*** type of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.

Definition at line 2705 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::getType().

10.1.4.75 void RDB::getJust (const string & *name*, RDBColumn::Just & *just*) const throw (RDBErr)

Return the justification of the RDBColumn.

Parameters:

name of RDBColumn

just justification of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.

Definition at line 2729 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getJust(), and getName().

10.1.4.76 void RDB::getDesc (const string & *name*, string & *desc*) const throw (RDBErr)

Return the description of the RDBColumn.

Parameters:

name of RDBColumn

desc description of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.

Definition at line 2753 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getDesc(), and getName().

10.1.4.77 void RDB::getData (const string & *name*, double & *data*) throw (RDBErr)

Return the data of the RDBColumn, converting if necessary.

Parameters:

name of RDBColumn

data data of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.

RDBErr if data is invalid datatype for RDBColumn.

Definition at line 2778 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataDouble(), RDBColumn::getDataDouble(), and getName().

10.1.4.78 void RDB::getData (const string & *name*, long & *data*) throw (RDBErr)

Return the data of the RDBColumn, converting if necessary.

Parameters:

name of RDBColumn

data data of RDBColumn.

Exceptions:

RDBErr if no matching column name is found.

RDBErr if data is invalid datatype for RDBColumn.

Definition at line 2812 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataLong(), RDBColumn::getDataLong(), and getName().

10.1.4.79 void RDB::getData (const string & *name*, string & *data*) throw (RDBErr)

Return the data of the RDBColumn, converting if necessary.

Parameters:

name of RDBColumn

data data of RDBColumn.

Exceptions:

RDBErr if data is invalid datatype for RDBColumn.

RDBErr if no matching column name is found.

Definition at line 2846 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataString(), RDBColumn::getDataString(), and getName().

10.1.4.80 string RDB::getName (const string & *name*) const throw (RDBErr)

Return the name of the RDBColumn.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

Returns:

name of RDBColumn.

Definition at line 2880 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getName(), and getName().

10.1.4.81 string RDB::getDef (const string & *name*) throw (RDBErr)

Return the definition of the RDBColumn.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

Returns:

definition of RDBColumn.

Definition at line 2904 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getDef(), and getName().

10.1.4.82 long RDB::getWidth (const string & *name*) const throw (RDBErr)

Return the width of the RDBColumn.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

Returns:

width of RDBColumn.

Definition at line 2928 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::getWidth().

10.1.4.83 RDBColumn::Type RDB::getType (const string & *name*) const throw (RDBErr)

Return the type of the RDBColumn.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

Returns:

type of RDBColumn.

Definition at line 2952 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, getName(), and RDBColumn::getType().

10.1.4.84 RDBColumn::Just RDB::getJust (const string & *name*) const throw (RDBErr)

Return the justification of the RDBColumn.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

Returns:

justification of RDBColumn.

Definition at line 2976 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getJust(), and getName().

10.1.4.85 string RDB::getDesc (const string & *name*) const throw (RDBErr)

Return the description of the RDBColumn.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

Returns:

description of RDBColumn.

Definition at line 3000 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumn::getDesc(), and getName().

10.1.4.86 double RDB::getDataDouble (const string & *name*) throw (RDBErr)

Return the data of the RDBColumn, converting if necessary.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

RDBErr if data is invalid datatype for RDBColumn.

Returns:

data of RDBColumn.

Definition at line 3025 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataDouble(), RDBColumn::getDataDouble(), and getName().

10.1.4.87 long RDB::getDataLong (const string & *name*) throw (RDBErr)

Return the data of the RDBColumn, converting if necessary.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

RDBErr if data is invalid datatype for RDBColumn.

Returns:

data of RDBColumn.

Definition at line 3058 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataLong(), RDBColumn::getDataLong(), and getName().

10.1.4.88 string RDB::getDataString (const string & *name*) throw (RDBErr)

Return the data of the RDBColumn, converting if necessary.

Parameters:

name of RDBColumn

Exceptions:

RDBErr if no matching column name is found.

RDBErr if data is invalid datatype for RDBColumn.

Returns:

data of RDBColumn.

Definition at line 3091 of file RDB.cc.

References _cols, _filename, _ncols, _nrcol, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataString(), RDBColumn::getDataString(), and getName().

10.1.4.89 size_t RDB::nComments (void) const

Return number of comments in RDB object.

Returns:

Number of comments.

Definition at line 3120 of file RDB.cc.

References _ncomms.

10.1.4.90 size_t RDB::nColumns (void) const

Return number of columns in RDB object.

Returns:

Number of columns.

Definition at line 3133 of file RDB.cc.

References _ncols.

10.1.4.91 size_t RDB::nRows (void)

Return number of rows in RDB object.

Returns:

Number of rows.

Warning:

This method reads through the entire RDB file to count the number of rows and then attempts to rewind the file to the location at which it began . If the stream attached to the RDB object is not seekable, this method will fail.

Definition at line 3151 of file RDB.cc.

References _isptr, _knowrows, _line, _mode, _nrows, and _rewindto.

10.1.4.92 void RDB::parseHeader (void) throw (RDBErr) `[protected]`

Parse header, i.e. comments and column names and definitions.

Parses out comments and column names and defintions from the RDB file header. This is used by RDB::open().

Definition at line 3185 of file RDB.cc.

References _cols, _isptr, _mycols, _ncols, _ncomms, _nrows, _rewindto, _rownum, and setComment().

Referenced by open().

10.1.4.93 vector< string > RDB::parseLine (const string & *line*) const `[protected]`

Parse fields in a row.

Parameters:

line the line to be split.

Returns:

Vector of tokens, one for each tab delimited field.

Splits the input line on tabs.

Definition at line 3284 of file RDB.cc.

10.1.4.94 size_t RDB::parseLine (bool & *newgroup*) throw (RDBErr) `[protected]`

Parse fields in a row.

Returns:

number of tokens parsed.

Splits the line on tabs and assigns the tokens to the RDBColumns.

Definition at line 3316 of file RDB.cc.

References _cols, _line, _rownum, newGroup(), and RDBColumn::setData().

10.1.5 Friends And Related Function Documentation

10.1.5.1 istream& operator>> (istream & *is*, RDB & *rdb*) throw (RDBErr) `[friend]`

Read table from input stream.

Parameters:

is the input stream.

rdb the table to fill.

Returns:

A reference to the input stream.

If the stream contains a valid rdbtable fill it... otherwise set ios::failbit.

Definition at line 38 of file RDB.cc.

10.1.5.2 ostream& operator<< (ostream & *os*, RDB & *rdb*) throw (RDBErr) `[friend]`

Write table to output stream.

Parameters:

os the output stream.

rdb the table to print.

Returns:

A reference to the output stream

Places the table on the output stream.

Definition at line 76 of file RDB.cc.

10.1.6 Member Data Documentation

10.1.6.1 string RDB::_filename `[protected]`

Name of RDB file.

Definition at line 303 of file RDB.h.

Referenced by close(), getColumn(), getComment(), getData(), getDataDouble(), getDataLong(), getDataString(), getDef(), getDesc(), getGroup(), getJust(), getName(), getType(), getWidth(), mapData(), open(), read(), setData(), setDef(), setDesc(), setGroup(), setJust(), setName(), setType(), and setWidth().

10.1.6.2 ios::openmode RDB::_mode `[protected]`

Open mode of the associated stream.

Definition at line 305 of file RDB.h.

Referenced by nRows(), open(), read(), rewind(), and write().

10.1.6.3 istream∗ RDB::_isptr `[protected]`

Istream attached to data file.

Definition at line 308 of file RDB.h.

Referenced by close(), nRows(), open(), parseHeader(), read(), simpleRDBTable< Type >::readRow(), rewind(), and ∼RDB().

10.1.6.4 ostream∗ RDB::_osptr `[protected]`

Ostream attached to data file.

Definition at line 310 of file RDB.h.

Referenced by close(), open(), rewind(), write(), and ∼RDB().

10.1.6.5 bool RDB::_myisptr `[protected]`

Indicates if RDB object is responsible for deallocating the istream.

Definition at line 312 of file RDB.h.

Referenced by close(), and open().

10.1.6.6 bool RDB::_myosptr `[protected]`

Indicates if RDB object is responsible for deallocating the ostream.

Definition at line 314 of file RDB.h.

Referenced by close(), and open().

10.1.6.7 size_t RDB::_rewindto `[protected]`

Position of beginning of first row of data.

Definition at line 317 of file RDB.h.

Referenced by close(), nRows(), parseHeader(), rewind(), and write().

10.1.6.8 size_t RDB::_ncomms `[protected]`

Number of comments.

Definition at line 319 of file RDB.h.

Referenced by close(), getComment(), nComments(), parseHeader(), setComment(), and write().

10.1.6.9 size_t RDB::_ncols `[protected]`

Number of columns.

Definition at line 321 of file RDB.h.

Referenced by advanceIdx(), close(), getColumn(), getData(), getDataDouble(), getDataLong(), getDataString(), getDef(), getDesc(), getGroup(), getJust(), getName(), getType(), getWidth(), mapData(), nColumns(), newGroup(), parseHeader(), read(), rewind(), setColumn(), setData(), setDef(), setDesc(), setGroup(), setJust(), setName(), setType(), setWidth(), write(), and ∼RDB().

10.1.6.10 size_t RDB::_nrows `[protected]`

Number of rows.

Definition at line 323 of file RDB.h.

Referenced by close(), nRows(), and parseHeader().

10.1.6.11 bool RDB::_knowrows `[protected]`

Indicates if associated file must be scanned to determine number of rows.

Definition at line 325 of file RDB.h.

Referenced by close(), and nRows().

10.1.6.12 long RDB::_rownum `[protected]`

Current ∗table∗ row number.

Definition at line 327 of file RDB.h.

Referenced by close(), parseHeader(), parseLine(), read(), rewind(), and write().

10.1.6.13 long RDB::_frownum `[protected]`

Current ∗file∗ row number.

Definition at line 329 of file RDB.h.

10.1.6.14 bool RDB::_autoidx `[protected]`

Indicates if RDBColumn data elements should be advanced.

Definition at line 332 of file RDB.h.

Referenced by autoIdx(), close(), read(), setGroup(), and write().

10.1.6.15 bool RDB::_firstread `[protected]`

Indicates if this is the first call to RDB::read.

Definition at line 334 of file RDB.h.

Referenced by read().

10.1.6.16 bool RDB::_lastread `[protected]`

Indicates if this is the last call to RDB::read.

Definition at line 336 of file RDB.h.

Referenced by read().

10.1.6.17 bool RDB::_writehdr `[protected]`

Indicates if the header has been output.

Definition at line 338 of file RDB.h.

Referenced by close(), and write().

10.1.6.18 RDBComment∗ RDB::_comms `[protected]`

Array of RDBComments.

Definition at line 341 of file RDB.h.

Referenced by close(), getComment(), setComment(), write(), and ∼RDB().

10.1.6.19 RDBColumn∗∗ RDB::_cols `[protected]`

Array of RDBColumns.

Definition at line 343 of file RDB.h.

Referenced by advanceIdx(), close(), getColumn(), getData(), getDataDouble(), getDataLong(), getDataString(), getDef(), getDesc(), getGroup(), getJust(), getName(), getType(), getWidth(), mapData(), newGroup(), parseHeader(), parseLine(), read(), rewind(), setColumn(), setData(), setDef(), setDesc(), setGroup(), setJust(), setName(), setType(), setWidth(), write(), and ∼RDB().

10.1.6.20 RDBLongColumn RDB::_nrcol `[protected]`

Hidden column, containing row number.

Definition at line 345 of file RDB.h.

Referenced by advanceIdx(), getColumn(), getData(), getDataDouble(), getDataLong(), getDataString(), getDef(), getDesc(), getGroup(), getJust(), getName(), getType(), getWidth(), mapData(), setData(), setDef(), setDesc(), setGroup(), setJust(), setName(), setType(), and setWidth().

10.1.6.21 bool∗ RDB::_mycols `[protected]`

Indicates if RDB object is responsible for deallocating given RDBColumn.

Definition at line 347 of file RDB.h.

Referenced by parseHeader(), setColumn(), and ∼RDB().

10.1.6.22 string RDB::_line `[protected]`

Line from RDB table.

Definition at line 350 of file RDB.h.

Referenced by nRows(), parseLine(), and read().

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

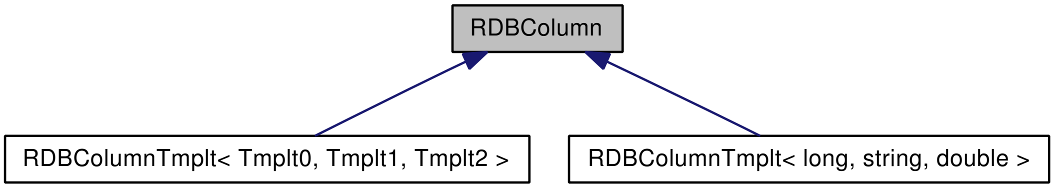
- RDB.h
- RDB.cc

10.2 RDBColumn Class Reference

Provides interface for general column related methods.

`#include <RDBColumn.h>`

Inheritance diagram for RDBColumn:


RDBColumn
RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >
RDBColumnTmplt< long, string, double >

Public Types

Enumerations for column definitions and error conditions.

- enum Just { **LEFT** = '<', **RIGHT** = '>' }
  Acceptable column justifications.
- enum Type { **MONTH** = 'M', **NUMERIC** = 'N', **STRING** = 'S' }
  Acceptable column types.

- enum Err {
  NOERR = 0, **LOSTPRECISION** = EDOM, **NONNUMERIC** = EINVAL, **OUTOFRANGE** = ERANGE,
  NODATA = ENODATA }
  Possible error conditions.
- enum Status { **CMOG** = 0x01, **CEOG** = 0x02, **CBOG** = 0x04 }
  Acceptable column justifications.

Public Member Functions

Constructing, destructing, and initializing RDBColumn objects.

- RDBColumn (const string &name="", const string &def="") throw (RDBErr)
  Assigns name and definition to RDBColumn object.
- RDBColumn (const RDBColumn &col)
  Copies RDBColumn object.
- virtual ∼RDBColumn (void)
  Deletes resources allocated by RDBColumn object.
- RDBColumn & operator= (const RDBColumn &col)
  Copies RDBColumn object.
- virtual RDBColumn & operator= (const double &data)=0 throw (RDBErr)
  Assigns data to RDBColumn object's _data member, converting as necessary.
- virtual RDBColumn & operator= (const long &data)=0 throw (RDBErr)
  Assigns data to RDBColumn object's _data member, converting as necessary.
- virtual RDBColumn & operator= (const string &data)=0 throw (RDBErr)
  Assigns data to RDBColumn object's _data member, converting as necessary.

Auto-indexing control methods.

- virtual void advanceIdx (void)=0
  Increments index to the RDBColumn's data elements.
- virtual void rewind (void)=0
  Rewinds index to the RDBColumn's data elements.

Group information ("break" column) methods.

- virtual void setGroup (bool group)

  Turn on/off group tracking for this column object.

- bool getGroup (void) const
- virtual void setGroupValue (void)=0

  Sets the group value to the current data value.

- virtual int newGroup (void)=0

  Returns the group status of this column object.

Data member initializers.

- void setName (const string &name)

  Sets the name.

- void setDef (const string &def) throw (RDBErr)

  Sets the definition.

- void setWidth (const long width)

  Sets the width.

- void setType (const RDBColumn::Type type)

  Sets the type.

- void setJust (const RDBColumn::Just just)

  Sets the justification.

- void setDesc (const string &desc)

  Sets the description.

- void setPrecision (const int precision)

  Sets the precision for numeric output and numeric to string conversion.

- void setThrow (const bool t=true)

  Sets the excpeption throwing behavior.

- void setErrNo (const int no=0)

  Sets the error status.

- virtual bool setData (const double &data)=0 throw (RDBErr)

  Sets the data value, converting as necessary.

- virtual bool setData (const long &data)=0 throw (RDBErr)

  Sets the data value, converting as necessary.

- virtual bool setData (const string &data)=0 throw (RDBErr)
  Sets the data value, converting as necessary.

Methods to map RDBColumn's data to user-supplied memory.

- virtual void mapData (double data[ ], const size_t nelems=1) throw (RDBErr)
  Maps data to user-supplied memory, if possible.
- virtual void mapData (long data[ ], const size_t nelems=1) throw (RDBErr)
  Maps data to user-supplied memory, if possible.
- virtual void mapData (string data[ ], const size_t nelems=1) throw (RDBErr)
  Maps data to user-supplied memory, if possible.

Data member accessors.

- string getName (void) const
  Returns the name.
- string getDef (void)
  Returns the definition.
- long getWidth (void) const
  Returns the width.
- RDBColumn::Type getType (void) const
  Returns the type.
- RDBColumn::Just getJust (void) const
  Returns the justification.
- string getDesc (void) const
  Returns the description.
- int getPrecision (void) const
  Returns the precision.
- bool getThrow (void) const
  Returns the state of the exception throwing behavior.
- char ∗ getErr (void) const

Returns a brief description of the error condition.

- int getErrNo (void) const
  Returns the error status.
- virtual void ∗ getData (void)=0
  Returns a pointer to the current data element.
- virtual bool getData (double &data)=0 throw (RDBErr)
  Returns the value of the current data element, converting if necessary.
- virtual bool getData (long &data)=0 throw (RDBErr)
  Returns the value of the current data element, converting if necessary.
- virtual bool getData (string &data)=0 throw (RDBErr)
  Returns the value of the current data element, converting if necessary.
- virtual double getDataDouble (void)=0 throw (RDBErr)
  Returns the value of the current data element, converting if necessary.
- virtual long getDataLong (void)=0 throw (RDBErr)
  Returns the value of the current data element, converting if necessary.
- virtual string getDataString (void)=0 throw (RDBErr)
  Returns the value of the current data element, converting if necessary.

Protected Member Functions

- void convert (const double &idata, double &odata) throw (RDBErr)
  Used to converted data based on user requests.
- void convert (const double &idata, long &odata) throw (RDBErr)
  Used to converted data based on user requests.
- void convert (const double &idata, string &odata) throw (RDBErr)
  Used to converted data based on user requests.
- void convert (const long &idata, double &odata) throw (RDBErr)
  Used to converted data based on user requests.
- void convert (const long &idata, long &odata) throw (RDBErr)
  Used to converted data based on user requests.
- void convert (const long &idata, string &odata) throw (RDBErr)

Used to converted data based on user requests.

- void convert (const string &idata, double &odata) throw (RDBErr)

  Used to converted data based on user requests.

- void convert (const string &idata, long &odata) throw (RDBErr)

  Used to converted data based on user requests.

- void convert (const string &idata, string &odata) throw (RDBErr)

  Used to converted data based on user requests.

- virtual istream & read (istream &is)=0 throw (RDBErr)

  Called by the stream insertion operator.

- virtual istream & extract (istream &is, double &data) throw (RDBErr)

  Overridden in the subclass of this datatype.

- virtual istream & extract (istream &is, long &data) throw (RDBErr)

  Overridden in the subclass of this datatype.

- virtual istream & extract (istream &is, string &data) throw (RDBErr)

  Overridden in the subclass of this datatype.

- virtual ostream & write (ostream &os) const =0

  Called by the stream extraction operator.

- virtual ostream & insert (ostream &os, double &data) const

  Overridden in the subclass of this datatype.

- virtual ostream & insert (ostream &os, long &data) const

  Overridden in the subclass of this datatype.

- virtual ostream & insert (ostream &os, string &data) const

  Overridden in the subclass of this datatype.

Protected Attributes

- string _name

  Name.

- string _def

Definition.

- long _width

  Width.

- RDBColumn::Type _type

  Data type.

- RDBColumn::Just _just

  Justification.

- string _desc

  Description.

- bool _changed

  Indicates state for the definition field.

- bool _throw

  State of the exception throwing behavior.

- int _errno

  Error state.

- int _precision

  Precision used for stream output or numeric to string conversion.

- stringstream _strstrm

  Used for numeric to string conversion.

- bool _group

  This is a group column.

- bool _initgroup

  Group been initialized.

Friends

Stream insertion and extraction operators.

- istream & operator>> (istream &is, RDBColumn &col) throw (RDBErr)

  Read column from input stream.

- istream & operator>> (istream &is, RDBColumn ∗col) throw (RDBErr)
  Read column from input stream.
- ostream & operator<< (ostream &os, const RDBColumn &col)
  Write column to output stream.
- ostream & operator<< (ostream &os, const RDBColumn ∗col)
  Write column to output stream.

10.2.1 Detailed Description

Provides interface for general column related methods.

Definition at line 43 of file RDBColumn.h.

10.2.2 Member Enumeration Documentation

10.2.2.1 enum RDBColumn::Just

Acceptable column justifications.

Definition at line 60 of file RDBColumn.h.

10.2.2.2 enum RDBColumn::Type

Acceptable column types.

Definition at line 62 of file RDBColumn.h.

10.2.2.3 enum RDBColumn::Err

Possible error conditions.

Definition at line 64 of file RDBColumn.h.

10.2.2.4 enum RDBColumn::Status

Acceptable column justifications.

Definition at line 65 of file RDBColumn.h.

10.2.3 Constructor & Destructor Documentation

10.2.3.1 RDBColumn::RDBColumn (const string & *name* = "", const string & *def* = "") throw (RDBErr)

Assigns name and definition to RDBColumn object.

Parameters:

name the column name.

def the column definition.

Initializes the RDBColumn name and definition. Sets the exception throwing behavior to true.

Definition at line 136 of file RDBColumn.cc.

10.2.3.2 RDBColumn::RDBColumn (const RDBColumn & *col*)

Copies RDBColumn object.

Parameters:

col the RDBColumn object to copy.

Copies the argument.

Definition at line 165 of file RDBColumn.cc.

10.2.3.3 RDBColumn::∼RDBColumn (void) `[virtual]`

Deletes resources allocated by RDBColumn object.

Nothing to do.

Definition at line 180 of file RDBColumn.cc.

10.2.4 Member Function Documentation

10.2.4.1 RDBColumn & RDBColumn::operator= (const RDBColumn & *col*)

Copies RDBColumn object.

Parameters:

col the RDBColumn object to copy.

Definition at line 191 of file RDBColumn.cc.

References _changed, _def, _desc, _errno, _group, _just, _name, _precision, _throw, _type, and _width.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator=().

10.2.4.2 virtual RDBColumn& RDBColumn::operator= (const double & *data*) throw (RDBErr) `[pure virtual]`

Assigns data to RDBColumn object's _data member, converting as necessary.

Implemented in RDBColumnTmplt< long, string, double >.

10.2.4.3 virtual RDBColumn& RDBColumn::operator= (const long & *data*) throw (RDBErr) `[pure virtual]`

Assigns data to RDBColumn object's _data member, converting as necessary.

Implemented in RDBColumnTmplt< long, string, double >.

10.2.4.4 virtual RDBColumn& RDBColumn::operator= (const string & *data*) throw (RDBErr) `[pure virtual]`

Assigns data to RDBColumn object's _data member, converting as necessary.

Implemented in RDBColumnTmplt< long, string, double >.

10.2.4.5 virtual void RDBColumn::advanceIdx (void) `[pure virtual]`

Increments index to the RDBColumn's data elements.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

Referenced by RDB::advanceIdx(), and RDB::read().

10.2.4.6 virtual void RDBColumn::rewind (void) `[pure virtual]`

Rewinds index to the RDBColumn's data elements.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

Referenced by RDB::rewind().

10.2.4.7 void RDBColumn::setGroup (bool *group*) `[virtual]`

Turn on/off group tracking for this column object.

Parameters:

group incidcates whether or not this is a group column.

Reimplemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

Definition at line 219 of file RDBColumn.cc.

References _group.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroup(), and RDB::setGroup().

10.2.4.8 bool RDBColumn::getGroup (void) const

Returns group status, RBOG if at beginning of a group, REOG if at ned of a group, or REOL if in the middle of a group.

Returns:

Whether or not this is a group column.

Definition at line 232 of file RDBColumn.cc.

References _group.

Referenced by RDB::getGroup().

10.2.4.9 virtual void RDBColumn::setGroupValue (void) `[pure virtual]`

Sets the group value to the current data value.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

10.2.4.10 virtual int RDBColumn::newGroup (void) `[pure virtual]`

Returns the group status of this column object.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

Referenced by RDB::read().

10.2.4.11 void RDBColumn::setName (const string & *name*)

Sets the name.

Parameters:

name the RDBColumn name to use.

Sets the RDBColumn name.

Definition at line 247 of file RDBColumn.cc.

References _name.

Referenced by RDB::setColumn(), and RDB::setName().

10.2.4.12 void RDBColumn::setDef (const string & *def*) throw (RDBErr)

Sets the definition.

Parameters:

> ***def*** the RDBColumn definition to use.

Exceptions:

> ***RDBErrColFmt*** error if the definition is an unrecognized format.

Sets the RDBColumn definition, as well as the width, type, justification, and description.

Definition at line 266 of file RDBColumn.cc.

References _changed, _def, _desc, _just, _name, _type, and _width.

Referenced by RDB::setColumn(), and RDB::setDef().

10.2.4.13 void RDBColumn::setWidth (const long *width*)

Sets the width.

Parameters:

> ***width*** the width of the RDBColumn.

The width has no effect on the storage or printing of the data. The RDB docs indicate that it is used only by the ptbl command.

Definition at line 340 of file RDBColumn.cc.

References _changed, and _width.

Referenced by RDB::setWidth().

10.2.4.14 void RDBColumn::setType (const RDBColumn::Type *type*)

Sets the type.

Parameters:

> ***type*** of the RDBColumn.

Possible values are 'M' for month, 'N' for numeric, 'S' for string. String is the default type.

Definition at line 357 of file RDBColumn.cc.

References _changed, and _type.

Referenced by RDB::setType().

10.2.4.15 void RDBColumn::setJust (const RDBColumn::Just *just*)

Sets the justification.

Parameters:

just the justification of the data when printing.

Possible values are '<' for left justification and '>' for right justification. Default is for months and strings to be left justified and numeric data to be right justified.

Definition at line 375 of file RDBColumn.cc.

References _changed, and _just.

Referenced by RDB::setJust().

10.2.4.16 void RDBColumn::setDesc (const string & *desc*)

Sets the description.

Parameters:

desc is the column documentation.

Sets the RDBColumn description.

Definition at line 391 of file RDBColumn.cc.

References _changed, and _desc.

Referenced by RDB::setDesc().

10.2.4.17 void RDBColumn::setPrecision (const int *precision*)

Sets the precision for numeric output and numeric to string conversion.

Parameters:

precision Sets the RDBColumn precision which controls the output precision of floating point data.

Definition at line 408 of file RDBColumn.cc.

References _precision.

10.2.4.18 void RDBColumn::setThrow (const bool *t* = `true`)

Sets the excpeption throwing behavior.

Parameters:

t state of the exception throwing behavior.

Sets the RDBColumn exception throwing behavior.

Definition at line 423 of file RDBColumn.cc.

References _throw.

10.2.4.19 void RDBColumn::setErrNo (const int *no* = `0`)

Sets the error status.

Parameters:

no Sets the RDBColumn error status.

Definition at line 438 of file RDBColumn.cc.

References _errno.

10.2.4.20 virtual bool RDBColumn::setData (const double & *data*) throw (RDBErr) `[pure virtual]`

Sets the data value, converting as necessary.

Implemented in RDBColumnTmplt< long, string, double >.

Referenced by RDB::parseLine(), RDB::read(), and RDB::setData().

10.2.4.21 virtual bool RDBColumn::setData (const long & *data*) throw (RDBErr) `[pure virtual]`

Sets the data value, converting as necessary.

Implemented in RDBColumnTmplt< long, string, double >.

10.2.4.22 virtual bool RDBColumn::setData (const string & *data*) throw (RDBErr) `[pure virtual]`

Sets the data value, converting as necessary.

Implemented in RDBColumnTmplt< long, string, double >.

10.2.4.23 void RDBColumn::mapData (double *data*[ ], const size_t *nelems* = `1`) throw (RDBErr) `[virtual]`

Maps data to user-supplied memory, if possible.

Definition at line 448 of file RDBColumn.cc.

References _name.

Referenced by RDB::mapData().

10.2.4.24 void RDBColumn::mapData (long *data*[ ], const size_t *nelems* = `1`) throw (RDBErr) `[virtual]`

Maps data to user-supplied memory, if possible.

Reimplemented in RDBColumnTmplt< long, string, double >.

Definition at line 459 of file RDBColumn.cc.

References _name.

10.2.4.25 void RDBColumn::mapData (string *data*[ ], const size_t *nelems* = `1`) throw (RDBErr) `[virtual]`

Maps data to user-supplied memory, if possible.

Definition at line 470 of file RDBColumn.cc.

References _name.

10.2.4.26 string RDBColumn::getName (void) const

Returns the name.

Returns the RDBColumn name.

Definition at line 484 of file RDBColumn.cc.

References _name.

Referenced by RDB::close(), RDB::getName(), and RDB::write().

10.2.4.27 string RDBColumn::getDef (void)

Returns the definition.

Returns the RDBColumn definition, reconstructing it from its constituent parts if any of them have changed.

Definition at line 498 of file RDBColumn.cc.

References _changed, _def, _just, _strstrm, _type, and _width.

Referenced by RDB::close(), RDB::getDef(), and RDB::write().

10.2.4.28 long RDBColumn::getWidth (void) const

Returns the width.

Returns the RDBColumn width.

Definition at line 540 of file RDBColumn.cc.

Referenced by RDB::getWidth().

10.2.4.29 RDBColumn::Type RDBColumn::getType (void) const

Returns the type.

Returns the RDBColumn type.

Definition at line 553 of file RDBColumn.cc.

Referenced by RDB::getType().

10.2.4.30 RDBColumn::Just RDBColumn::getJust (void) const

Returns the justification.

Returns the RDBColumn justification.

Definition at line 566 of file RDBColumn.cc.

Referenced by RDB::getJust().

10.2.4.31 string RDBColumn::getDesc (void) const

Returns the description.

Returns the RDBColumn description.

Definition at line 579 of file RDBColumn.cc.

Referenced by RDB::getDesc().

10.2.4.32 int RDBColumn::getPrecision (void) const

Returns the precision.

Returns the RDBColumn precision used for stream output.

Definition at line 592 of file RDBColumn.cc.

10.2.4.33 bool RDBColumn::getThrow (void) const

Returns the state of the exception throwing behavior.

Returns the RDBColumn exception throwing flag.

Definition at line 605 of file RDBColumn.cc.

10.2.4.34 char ∗ RDBColumn::getErr (void) const

Returns a brief description of the error condition.

Returns a string describing the RDBColumn error flag.

Definition at line 618 of file RDBColumn.cc.

10.2.4.35 int RDBColumn::getErrNo (void) const

Returns the error status.

Returns the RDBColumn error flag.

Definition at line 631 of file RDBColumn.cc.

10.2.4.36 virtual void∗ RDBColumn::getData (void) `[pure virtual]`

Returns a pointer to the current data element.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

10.2.4.37 virtual bool RDBColumn::getData (double & *data*) throw (RDBErr) `[pure virtual]`

Returns the value of the current data element, converting if necessary.

Implemented in RDBColumnTmplt< long, string, double >.

10.2.4.38 virtual bool RDBColumn::getData (long & *data*) throw (RDBErr) `[pure virtual]`

Returns the value of the current data element, converting if necessary.

Implemented in RDBColumnTmplt< long, string, double >.

10.2.4.39 virtual bool RDBColumn::getData (string & *data*) throw (RDBErr) `[pure virtual]`

Returns the value of the current data element, converting if necessary.

Implemented in RDBColumnTmplt< long, string, double >.

10.2.4.40 virtual double RDBColumn::getDataDouble (void) throw (RDBErr) `[pure virtual]`

Returns the value of the current data element, converting if necessary.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

Referenced by RDB::getData(), and RDB::getDataDouble().

10.2.4.41 virtual long RDBColumn::getDataLong (void) throw (RDBErr) `[pure virtual]`

Returns the value of the current data element, converting if necessary.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

Referenced by RDB::getData(), and RDB::getDataLong().

10.2.4.42 virtual string RDBColumn::getDataString (void) throw (RDBErr) `[pure virtual]`

Returns the value of the current data element, converting if necessary.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

Referenced by RDB::getData(), and RDB::getDataString().

10.2.4.43 void RDBColumn::convert (const double & *idata*, double & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

- ***idata*** input data.
- ***odata*** output data.

Handles the trivial conversion for double to double for child classes.

Definition at line 647 of file RDBColumn.cc.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataDouble(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataLong(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataString(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator=(), and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData().

10.2.4.44 void RDBColumn::convert (const double & *idata*, long & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

idata input data.

odata output data.

Handles the trivial conversion for double to long for child classes.

Definition at line 664 of file RDBColumn.cc.

10.2.4.45 void RDBColumn::convert (const double & *idata*, string & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

idata input data.

odata output data.

Handles the non-trivial conversion for double to string for child classes.

Warning:

This is slow. Like dirt. Don't do it often.

Definition at line 684 of file RDBColumn.cc.

10.2.4.46 void RDBColumn::convert (const long & *idata*, double & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

idata input data.

odata output data.

Handles the trivial conversion for long to double for child classes.

Definition at line 705 of file RDBColumn.cc.

10.2.4.47 void RDBColumn::convert (const long & *idata*, long & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

idata input data.

odata output data.

Handles the trivial conversion for long to long for child classes.

Definition at line 722 of file RDBColumn.cc.

10.2.4.48 void RDBColumn::convert (const long & *idata*, string & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

idata input data.

odata output data.

Handles the non-trivial conversion for long to string for child classes.

Warning:

This is slow. Like dirt. Don't do it often.

Definition at line 741 of file RDBColumn.cc.

10.2.4.49 void RDBColumn::convert (const string & *idata*, double & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

idata input data.

odata output data.

Exceptions:

RDBErr error if idata is non-numeric.

Handles the non-trivial conversion for string to double for child classes.

Definition at line 764 of file RDBColumn.cc.

10.2.4.50 void RDBColumn::convert (const string & *idata*, long & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

idata input data.

odata output data.

Exceptions:

RDBErr error if idata is non-numeric.

RDBErr error if idata represents a floating point number and precision is lost converting to integer number.

Handles the non-trivial conversion for string to long for child classes.

Definition at line 796 of file RDBColumn.cc.

10.2.4.51 void RDBColumn::convert (const string & *idata*, string & *odata*) throw (RDBErr) `[protected]`

Used to converted data based on user requests.

Parameters:

idata input data.

odata output data.

Handles the trivial conversion for string to string for child classes.

Definition at line 848 of file RDBColumn.cc.

10.2.4.52 virtual istream& RDBColumn::read (istream & *is*) throw (RDBErr) `[protected, pure virtual]`

Called by the stream insertion operator.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

10.2.4.53 istream & RDBColumn::extract (istream & *is*, double & *data*) throw (RDBErr) `[protected, virtual]`

Overridden in the subclass of this datatype.

Parameters:

is input string.

data double data.

Returns:

istream the input stream.

Extracts a double from the input stream.

Definition at line 867 of file RDBColumn.cc.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::read().

10.2.4.54 istream & RDBColumn::extract (istream & *is*, long & *data*) throw (RDBErr) `[protected, virtual]`

Overridden in the subclass of this datatype.

Parameters:

is input string.

data long data.

Returns:

istream the input stream.

Extracts a long from the input stream.

Definition at line 891 of file RDBColumn.cc.

10.2.4.55 istream & RDBColumn::extract (istream & *is*, string & *data*) throw (RDBErr) `[protected, virtual]`

Overridden in the subclass of this datatype.

Parameters:

is input string.

data string data.

Returns:

istream the input stream.

Extracts a string from the input stream. Extraction stops at the first tab or newline character.

Definition at line 917 of file RDBColumn.cc.

10.2.4.56 virtual ostream& RDBColumn::write (ostream & *os*) const `[protected, pure virtual]`

Called by the stream extraction operator.

Implemented in RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >, and RDBColumnTmplt< long, string, double >.

10.2.4.57 ostream & RDBColumn::insert (ostream & *os*, double & *data*) const `[protected, virtual]`

Overridden in the subclass of this datatype.

Parameters:

os output string.

data double data.

Returns:

ostream the output stream.

Inserts a double into the output stream.

Definition at line 953 of file RDBColumn.cc.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::write().

10.2.4.58 ostream & RDBColumn::insert (ostream & *os*, long & *data*) const `[protected, virtual]`

Overridden in the subclass of this datatype.

Parameters:

os output string.

data double data.

Returns:

ostream the output stream.

Inserts a long into the output stream.

Definition at line 972 of file RDBColumn.cc.

10.2.4.59 ostream & RDBColumn::insert (ostream & *os*, string & *data*) const `[protected, virtual]`

Overridden in the subclass of this datatype.

Parameters:

os output string.

data double data.

Returns:

ostream the output stream.

Inserts a string into the output stream.

Definition at line 991 of file RDBColumn.cc.

10.2.5 Friends And Related Function Documentation

10.2.5.1 istream& operator>> (istream & *is*, RDBColumn & *col*) throw (RDBErr) `[friend]`

Read column from input stream.

Parameters:

is input stream.

col RDBColumn to fill.

Returns:

A reference to the input stream.

Reads one data element into this object.

Definition at line 38 of file RDBColumn.cc.

10.2.5.2 istream& operator>> (istream & *is*, RDBColumn ∗ *col*) throw (RDBErr) `[friend]`

Read column from input stream.

Parameters:

is input stream.
col RDBColumn to fill.

Returns:

A reference to the input stream.

Reads one data element into this object.

Definition at line 69 of file RDBColumn.cc.

10.2.5.3 ostream& operator<< (ostream & *os*, const RDBColumn & *col*) `[friend]`

Write column to output stream.

Parameters:

os output stream.
col RDBColumn to fill.

Returns:

A reference to the output stream.

Writes one data element from this object.

Definition at line 100 of file RDBColumn.cc.

10.2.5.4 ostream& operator<< (ostream & *os*, const RDBColumn ∗ *col*) `[friend]`

Write column to output stream.

Parameters:

- ***os*** output stream.
- ***col*** RDBColumn to fill.

Returns:

A reference to the output stream.

Writes one data element from this object.

Definition at line 119 of file RDBColumn.cc.

10.2.6 Member Data Documentation

10.2.6.1 string RDBColumn::_name `[protected]`

Name.

Definition at line 225 of file RDBColumn.h.

Referenced by getName(), mapData(), operator=(), setDef(), and setName().

10.2.6.2 string RDBColumn::_def `[protected]`

Definition.

Definition at line 227 of file RDBColumn.h.

Referenced by getDef(), operator=(), and setDef().

10.2.6.3 long RDBColumn::_width `[protected]`

Width.

Definition at line 229 of file RDBColumn.h.

Referenced by getDef(), operator=(), setDef(), and setWidth().

10.2.6.4 RDBColumn::Type RDBColumn::_type `[protected]`

Data type.

Definition at line 231 of file RDBColumn.h.

Referenced by getDef(), operator=(), setDef(), and setType().

10.2.6.5 RDBColumn::Just RDBColumn::_just `[protected]`

Justification.

Definition at line 233 of file RDBColumn.h.

Referenced by getDef(), operator=(), setDef(), and setJust().

10.2.6.6 string RDBColumn::_desc `[protected]`

Description.

Definition at line 235 of file RDBColumn.h.

Referenced by operator=(), setDef(), and setDesc().

10.2.6.7 bool RDBColumn::_changed `[protected]`

Indicates state for the definition field.

Definition at line 237 of file RDBColumn.h.

Referenced by getDef(), operator=(), setDef(), setDesc(), setJust(), setType(), and setWidth().

10.2.6.8 bool RDBColumn::_throw `[protected]`

State of the exception throwing behavior.

Definition at line 240 of file RDBColumn.h.

Referenced by operator=(), and setThrow().

10.2.6.9 int RDBColumn::_errno `[protected]`

Error state.

Definition at line 242 of file RDBColumn.h.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getData(), operator=(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData(), and setErrNo().

10.2.6.10 int RDBColumn::_precision `[protected]`

Precision used for stream output or numeric to string conversion.

Definition at line 244 of file RDBColumn.h.

Referenced by operator=(), and setPrecision().

10.2.6.11 stringstream RDBColumn::_strstrm `[protected]`

Used for numeric to string conversion.

Definition at line 246 of file RDBColumn.h.

Referenced by getDef().

10.2.6.12 bool RDBColumn::_group `[protected]`

This is a group column.

Definition at line 248 of file RDBColumn.h.

Referenced by getGroup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::mapData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::newGroup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator=(), operator=(), setGroup(), and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroupValue().

10.2.6.13 bool RDBColumn::_initgroup `[protected]`

Group been initialized.

Definition at line 250 of file RDBColumn.h.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::newGroup().

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

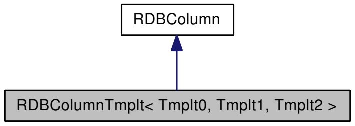
- RDBColumn.h
- RDBColumn.cc

10.3 RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 > Class Template Reference

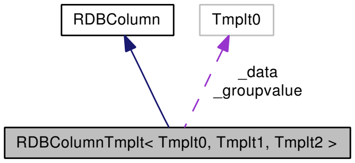
Parameterizes RDBColumn interface for many data types.

`#include <RDBColumnTmplt.h>`

Inheritance diagram for RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >:



Collaboration diagram for RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >:



Public Member Functions

Constructing, destructing, and initializing RDB columns.

- RDBColumnTmplt (const string &name="", const string &def="") throw (RDBErr)

  Assigns name and definition.

- RDBColumnTmplt (const RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 > &rdbcolumntmplt)

  Copies RDBColumnTmplt object.

- ∼RDBColumnTmplt (void)

  Deletes resources allocated by RDBColumnTmplt object.

- RDBColumn & operator= (const RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 > &rdbcolumntmplt)

  Copies RDBColumnTmplt object.

- virtual RDBColumn & operator= (const Tmplt0 &data) throw (RDBErr)

  Assigns data to RDBColumn object's _data member, converting as necessary.

- virtual RDBColumn & operator= (const Tmplt1 &data) throw (RDBErr)

  Assigns data to RDBColumn object's _data member, converting as necessary.

- virtual RDBColumn & operator= (const Tmplt2 &data) throw (RDBErr)

  Assigns data to RDBColumn object's _data member, converting as necessary.

Auto-indexing control methods.

- virtual void advanceIdx (void)

  Increments index to the RDBColumn's data elements.

- virtual void rewind (void)

  Rewinds index to the RDBColumn's data elements.

Group information ("break" column) methods.

- virtual void setGroup (bool group)
- virtual void setGroupValue (void)

  Sets the group value to the current data value.
- virtual int newGroup (void)

  Returns the group status of this column object.

Data member initializers.

- virtual bool setData (const Tmplt0 &data) throw (RDBErr)

  Sets the data value, converting as necessary.
- virtual bool setData (const Tmplt1 &data) throw (RDBErr)

  Sets the data value, converting as necessary.
- virtual bool setData (const Tmplt2 &data) throw (RDBErr)

  Sets the data value, converting as necessary.

Methods to map RDBColumn's data to user-supplied memory.

- virtual void mapData (Tmplt0 data[ ], const size_t nelems) throw (RDBErr)

  Maps data to user-supplied memory.

Data member accessors.

- void $*$ getData (void)

  Returns a pointer to the current data element.
- virtual bool getData (Tmplt0 &data) throw (RDBErr)

  Returns the value of the current data element, converting if necessary.
- virtual bool getData (Tmplt1 &data) throw (RDBErr)

  Returns the value of the current data element, converting if necessary.
- virtual bool getData (Tmplt2 &data) throw (RDBErr)

  Returns the value of the current data element, converting if necessary.
- virtual double getDataDouble (void) throw (RDBErr)

  Returns the value of the current data element, converting if necessary.
- virtual long getDataLong (void) throw (RDBErr)

Returns the value of the current data element, converting if necessary.

- virtual string getDataString (void) throw (RDBErr)

  Returns the value of the current data element, converting if necessary.

Protected Member Functions

- virtual istream & read (istream &is) throw (RDBErr)

  Called by the stream insertion operator.

- virtual ostream & write (ostream &os) const

  Called by the stream extraction operator.

- void cleanup (void)

  Deletes resources allocated by RDBColumnTmplt object.

Protected Attributes

- Tmplt0 $*$ _data

  Pointer to the data managed by object.

- size_t _idx

  Index into the data.

- size_t _nelems

  Number of elements of data.

- bool _mine

  Indicates that RDBColumnTmplt is responsible for deallocating the data.

- Tmplt0 _groupvalue

  Current group value.

10.3.1 Detailed Description

template< class Tmplt0, class Tmplt1, class Tmplt2> class RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >

Parameterizes RDBColumn interface for many data types.

Definition at line 38 of file RDBColumnTmplt.h.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 template<class Tmplt0, class Tmplt1, class Tmplt2> RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::RDBColumnTmplt (const string & *name* = `""`, const string & *def* = `""`) throw (RDBErr) `[inline]`

Assigns name and definition.

Parameters:

name the column name.

def the column definition.

Allocates space for a single data element of type Tmplt1.

Definition at line 37 of file RDBColumnTmplt.cc.

10.3.2.2 template<class Tmplt0, class Tmplt1, class Tmplt2> RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::RDBColumnTmplt (const RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 > & *rdbcolumntmplt*) `[inline]`

Copies RDBColumnTmplt object.

Parameters:

rdbcolumntmplt col the RDBColumn object to copy.

Makes a shallow copy of the argument. The two RDBColumn objects share data elements.

Definition at line 58 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_mine, and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_nelems.

10.3.2.3 template<class Tmplt0, class Tmplt1, class Tmplt2> RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::~RDBColumnTmplt (void) `[inline]`

Deletes resources allocated by RDBColumnTmplt object.

Responsible for freeing data element memory allocated by the RDBColumn object.

Definition at line 76 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::cleanup().

10.3.3 Member Function Documentation

10.3.3.1 template<class Tmplt0, class Tmplt1, class Tmplt2> RDBColumn & RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator= (const RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 > & *col*) `[inline]`

Copies RDBColumnTmplt object.

Parameters:

col the RDBColumn object to copy.

Makes a shallow copy of the argument. The two RDBColumn objects share data elements.

Definition at line 93 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_group, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_mine, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_nelems, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::cleanup(), and RDBColumn::operator=().

10.3.3.2 template<class Tmplt0, class Tmplt1, class Tmplt2> RDBColumn & RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator= (const Tmplt0 & *data*) throw (RDBErr) `[inline, virtual]`

Assigns data to RDBColumn object's _data member, converting as necessary.

Parameters:

data Assigns the value to the current RDBColumn data element.

Definition at line 124 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

10.3.3.3 template<class Tmplt0, class Tmplt1, class Tmplt2> RDBColumn & RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator= (const Tmplt1 & *data*) throw (RDBErr) `[inline, virtual]`

Assigns data to RDBColumn object's _data member, converting as necessary.

Parameters:

data

Exceptions:

RDBErr error if the user attempts to convert non-numeric string data to a numeric column.

RDBErr error if the user attempts to convert string data representing a floating point number to an integer column.

Assigns the value to the current RDBColumn data element.

Definition at line 155 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

10.3.3.4 template<class Tmplt0, class Tmplt1, class Tmplt2> RDBColumn & RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator= (const Tmplt2 & *data*) throw (RDBErr) `[inline, virtual]`

Assigns data to RDBColumn object's _data member, converting as necessary.

Parameters:

data

Exceptions:

RDBErr error if the user attempts to convert non-numeric string data to a numeric column.

RDBErr error if the user attempts to convert string data representing a floating point number to an integer column.

Assigns the value to the current RDBColumn data element.

Definition at line 186 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

10.3.3.5 template<class Tmplt0, class Tmplt1, class Tmplt2> void RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::advanceIdx (void) `[inline, virtual]`

Increments index to the RDBColumn's data elements.

Advances the automatic index for the data elements by one.

Implements RDBColumn.

Definition at line 212 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_nelems.

Referenced by RDB::advanceIdx().

10.3.3.6 template<class Tmplt0, class Tmplt1, class Tmplt2> void RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::rewind (void) `[inline, virtual]`

Rewinds index to the RDBColumn's data elements.

Rewinds the automatic index for the data elements to the first element.

Implements RDBColumn.

Definition at line 228 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx.

10.3.3.7 template<class Tmplt0, class Tmplt1, class Tmplt2> void RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroup (bool *group*) `[inline, virtual]`

Parameters:

group	incidcates whether or not this is a group column.

Reimplemented from RDBColumn.

Definition at line 242 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_mine, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_nelems, and RDBColumn::setGroup().

Referenced by RDB::setGroup().

10.3.3.8 template<class Tmplt0, class Tmplt1, class Tmplt2> void RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroupValue (void) `[inline, virtual]`

Sets the group value to the current data value.

Sets the group for this object to the current data element.

Implements RDBColumn.

Definition at line 268 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_group, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_groupvalue, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_nelems.

10.3.3.9 template<class Tmplt0, class Tmplt1, class Tmplt2> int RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::newGroup (void) `[inline, virtual]`

Returns the group status of this column object.

Returns:

True if the data element is in a new group.

Implements RDBColumn.

Definition at line 285 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_group, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_groupvalue, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, RDBColumn::_initgroup, and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_nelems.

10.3.3.10 template<class Tmplt0, class Tmplt1, class Tmplt2> bool RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData (const Tmplt0 & *data*) throw (RDBErr) `[inline, virtual]`

Sets the data value, converting as necessary.

Parameters:

data

Returns:

True upon successful conversion, false otherwise.

Assigns the value to the current RDBColumn data element.

Definition at line 343 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_errno, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

Referenced by RDB::setData().

10.3.3.11 template<class Tmplt0, class Tmplt1, class Tmplt2> bool RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData (const Tmplt1 & *data*) throw (RDBErr) `[inline, virtual]`

Sets the data value, converting as necessary.

Parameters:

data

Exceptions:

RDBErr error if the user attempts to convert non-numeric string data to a numeric column.

RDBErr error if the user attempts to convert string data representing a floating point number to an integer column.

Returns:

True upon successful conversion, false otherwise.

Assigns the value to the current RDBColumn data element.

Definition at line 376 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_errno, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

10.3.3.12 template<class Tmplt0, class Tmplt1, class Tmplt2> bool RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData (const Tmplt2 & *data*) throw (RDBErr) `[inline, virtual]`

Sets the data value, converting as necessary.

Parameters:

data

Exceptions:

RDBErr error if the user attempts to convert non-numeric string data to a numeric column.

RDBErr error if the user attempts to convert string data representing a floating point number to an integer column.

Returns:

True upon successful conversion, false otherwise.

Assigns the value to the current RDBColumn data element.

Definition at line 409 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_errno, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

10.3.3.13 template<class Tmplt0, class Tmplt1, class Tmplt2> void RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::mapData (Tmplt0 *data*[ ], const size_t *nelems*) throw (RDBErr) `[inline, virtual]`

Maps data to user-supplied memory.

Parameters:

data pointer to a data element of type Tmplt0.

nelems number of data elements in the array pointed by data.

This method associates user allocated memory pointed to by data with this RDBColumn object. The used is responsible for freeing the memory pointed to by data after the RDBColumn is destroyed.

Definition at line 440 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_group, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_mine, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_nelems, and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::cleanup().

Referenced by RDB::mapData().

10.3.3.14 template<class Tmplt0, class Tmplt1, class Tmplt2> void * RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getData (void) `[inline, virtual]`

Returns a pointer to the current data element.

Returns:

Pointer to the current data element.

This method returns a pointer to the current data element. Modifications to the data returned are evident within this object.

Implements RDBColumn.

Definition at line 466 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx.

10.3.3.15 template<class Tmplt0, class Tmplt1, class Tmplt2> bool RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getData (Tmplt0 & *data*) throw (RDBErr) `[inline, virtual]`

Returns the value of the current data element, converting if necessary.

Parameters:

data assigned the value of the current data element in this object.

Returns:

True if the conversion was successful, false otherwise.

Assigns the value of the current RDBColumn data element to the arguement.

Definition at line 484 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_errno, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

10.3.3.16 template<class Tmplt0, class Tmplt1, class Tmplt2> bool RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getData (Tmplt1 & *data*) throw (RDBErr) `[inline, virtual]`

Returns the value of the current data element, converting if necessary.

Parameters:

data assigned the value of the current data element in this object.

Exceptions:

RDBErr error if the user attempts to convert a string column with non-numeric data to a numeric arguemnt.

RDBErr error if the user attempts to convert a string column representing a floating point number to an integer argument.

Returns:

True if the conversion was successful, false otherwise.

Assigns the value of the current RDBColumn data element to the arguement.

Definition at line 517 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_errno, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

10.3.3.17 template<class Tmplt0, class Tmplt1, class Tmplt2> bool RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getData (Tmplt2 & *data*) throw (RDBErr) `[inline, virtual]`

Returns the value of the current data element, converting if necessary.

Parameters:

data assigned the value of the current data element in this object.

Exceptions:

RDBErr error if the user attempts to convert a string column with non-numeric data to a numeric arguemnt.

RDBErr error if the user attempts to convert a string column representing a floating point number to an integer argument.

Returns:

True if the conversion was successful, false otherwise.

Assigns the value of the current RDBColumn data element to the arguement.

Definition at line 550 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumn::_errno, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

10.3.3.18 template<class Tmplt0, class Tmplt1, class Tmplt2> double RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataDouble (void) throw (RDBErr) `[inline, virtual]`

Returns the value of the current data element, converting if necessary.

Exceptions:

RDBErr error if the user attempts to convert a string column with non-numeric data to a numeric arguemnt.

Returns:

The data element as a double.

Implements RDBColumn.

Definition at line 578 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

Referenced by RDB::getData(), and RDB::getDataDouble().

10.3.3.19 template<class Tmplt0, class Tmplt1, class Tmplt2> long RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataLong (void) throw (RDBErr) `[inline, virtual]`

Returns the value of the current data element, converting if necessary.

Exceptions:

RDBErr error if the user attempts to convert a string column with non-numeric data to a numeric arguemnt.

RDBErr error if the user attempts to convert a string column representing a floating point number to an integer argument.

Returns:

The data element as a long.

Implements RDBColumn.

Definition at line 608 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

Referenced by RDB::getData(), and RDB::getDataLong().

10.3.3.20 template<class Tmplt0, class Tmplt1, class Tmplt2> string RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataString (void) throw (RDBErr) `[inline, virtual]`

Returns the value of the current data element, converting if necessary.

Returns:

The data element as a string.

Implements RDBColumn.

Definition at line 635 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::convert().

Referenced by RDB::getData(), and RDB::getDataString().

10.3.3.21 template<class Tmplt0, class Tmplt1, class Tmplt2> istream & RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::read (istream & *is*) throw (RDBErr) `[inline, protected, virtual]`

Called by the stream insertion operator.

Parameters:

is input stream.

Returns:

The input stream.

Called by the RDBColumn stream extraction operator.

Implements RDBColumn.

Definition at line 666 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::extract().

10.3.3.22 template<class Tmplt0, class Tmplt1, class Tmplt2> ostream & RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::write (ostream & *os*) const `[inline, protected, virtual]`

Called by the stream extraction operator.

Parameters:

os output stream.

Returns:

The output stream.

Called by the RDBColumn stream insertion operator.

Implements RDBColumn.

Definition at line 696 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx, and RDBColumn::insert().

10.3.3.23 template<class Tmplt0, class Tmplt1, class Tmplt2> void RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::cleanup (void) `[inline, protected]`

Deletes resources allocated by RDBColumnTmplt object.

Frees memory allocated by this object.

Definition at line 710 of file RDBColumnTmplt.cc.

References RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data, and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_mine.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::mapData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator=(), and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::~RDBColumnTmplt().

10.3.4 Member Data Documentation

10.3.4.1 template<class Tmplt0, class Tmplt1, class Tmplt2> Tmplt0* RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_data `[protected]`

Pointer to the data managed by object.

Definition at line 120 of file RDBColumnTmplt.h.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::cleanup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataDouble(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataLong(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataString(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::mapData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::newGroup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator=(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::RDBColumnTmplt(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::read(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroupValue(), and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::write().

10.3.4.2 template<class Tmplt0, class Tmplt1, class Tmplt2> size_t RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_idx `[protected]`

Index into the data.

Definition at line 122 of file RDBColumnTmplt.h.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::advanceIdx(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataDouble(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataLong(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::getDataString(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::mapData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::newGroup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator=(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::RDBColumnTmplt(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::read(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::rewind(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroupValue(), and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::write().

10.3.4.3 template<class Tmplt0, class Tmplt1, class Tmplt2> size_t RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_nelems `[protected]`

Number of elements of data.

Definition at line 124 of file RDBColumnTmplt.h.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::advanceIdx(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::mapData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::newGroup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator=(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::RDBColumnTmplt(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroup(), and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroupValue().

10.3.4.4 template<class Tmplt0, class Tmplt1, class Tmplt2> bool RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_mine `[protected]`

Indicates that RDBColumnTmplt is responsible for deallocating the data.

Definition at line 126 of file RDBColumnTmplt.h.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::cleanup(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::mapData(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::operator=(), RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::RDBColumnTmplt(), and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroup().

10.3.4.5 template<class Tmplt0, class Tmplt1, class Tmplt2> Tmplt0 RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::_groupvalue `[protected]`

Current group value.

Definition at line 129 of file RDBColumnTmplt.h.

Referenced by RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::newGroup(), and RDBColumnTmplt< Tmplt0, Tmplt1, Tmplt2 >::setGroupValue().

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

- RDBColumnTmplt.h
- RDBColumnTmplt.cc

10.4 RDBComment Class Reference

Provides interface for manipulating RDB comments.

```
#include <RDBComment.h>
```

Public Member Functions

Constructing, destructing, and initializing RDBComment objects.

- RDBComment (const string &comment="") throw (RDBErr)
  Parses comment string for RDB comment structure.
- RDBComment (const RDBComment &rdbcomment)
  Copy RDBComment object.
- ∼RDBComment (void)
  Destructor has nothing to do.
- const RDBComment & operator= (const RDBComment &rdbcomment)
  Copy RDBComment object.
- const RDBComment & operator= (const string &rdbcomment)
  Copy string to RDBComment object.

Data member initializers.

- void setComment (const string &comment) throw (RDBErr)
  Parses comment string for RDB comment elements.
- void setCommentText (const string &comment)
  Parses string for RDB comment elements.
- void setKeyword (const string &keyword) throw (RDBErr)
  Set just the comment keyword.
- void setValue (const string &value) throw (RDBErr)
  Set just the comment value.

Data member accessors.

- string getComment (void) const
  Return the full comment.
- string getCommentText (void) const
  Return the full comment text.
- string getKeyword (void) const

Return the keyword, if any.

- string getValue (void) const

  Return the keyword's value, if any.

Friends

Stream insertion and extraction operators.

- istream & operator>> (istream &is, RDBComment &rdbcomment) throw (RDBErr)

  Read comment from input stream.

- ostream & operator<< (ostream &os, const RDBComment &rdbcomment)

  Write comment to output stream.

10.4.1 Detailed Description

Provides interface for manipulating RDB comments.

Definition at line 35 of file RDBComment.h.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 RDBComment::RDBComment (const string & *comment* = "") throw (RDBErr)

Parses comment string for RDB comment structure.

Parameters:

comment the comment string.

Assigns the string comment to the RDBComment's data member. Parses for keyword=value pairs.

Definition at line 102 of file RDBComment.cc.

10.4.2.2 RDBComment::RDBComment (const RDBComment & *rdbcomment*)

Copy RDBComment object.

Parameters:

rdbcomment a reference to the RDBComment to copy.

Copies the RDBComment into this object.

Definition at line 125 of file RDBComment.cc.

References operator=().

10.4.2.3 RDBComment::∼RDBComment (void)

Destructor has nothing to do.

Nothing to do.

Definition at line 137 of file RDBComment.cc.

10.4.3 Member Function Documentation

10.4.3.1 const RDBComment & RDBComment::operator= (const RDBComment & *rdbcomment*)

Copy RDBComment object.

Parameters:

rdbcomment a reference to the RDBComment to copy.

Returns:

A reference to this.

Definition at line 150 of file RDBComment.cc.

References _comment, _keyword, and _value.

Referenced by RDBComment().

10.4.3.2 const RDBComment & RDBComment::operator= (const string & *comment*)

Copy string to RDBComment object.

Parameters:

comment a string comment to copy.

Returns:

A reference to this.

Definition at line 172 of file RDBComment.cc.

References setCommentText().

10.4.3.3 void RDBComment::setComment (const string & *comment*) throw (RDBErr)

Parses comment string for RDB comment elements.

Parameters:

comment Strips leading '#' character if present. If next character is ':', then it searches for a keyword=value pair.

Definition at line 190 of file RDBComment.cc.

References setCommentText().

Referenced by setKeyword(), and setValue().

10.4.3.4 void RDBComment::setCommentText (const string & *comment*)

Parses string for RDB comment elements.

Parameters:

comment Strips leading '#' character if present. If next character is ':', then it searches for a keyword=value pair.

Definition at line 222 of file RDBComment.cc.

Referenced by operator=(), and setComment().

10.4.3.5 void RDBComment::setKeyword (const string & *keyword*) throw (RDBErr)

Set just the comment keyword.

Parameters:

keyword the keyword part of a keyword=value pair.

Updates just the keyword half of a keyword=value pair.

Definition at line 278 of file RDBComment.cc.

References setComment().

10.4.3.6 void RDBComment::setValue (const string & *value*) throw (RDBErr)

Set just the comment value.

Parameters:

value the value part of a keyword=value pair.

Updates just the value half of a keyword=value pair.

Definition at line 302 of file RDBComment.cc.

References setComment().

10.4.3.7 string RDBComment::getComment (void) const

Return the full comment.

Returns:

The comment string.

Definition at line 324 of file RDBComment.cc.

10.4.3.8 string RDBComment::getCommentText (void) const

Return the full comment text.

Returns:

The comment string.

Definition at line 346 of file RDBComment.cc.

10.4.3.9 string RDBComment::getKeyword (void) const

Return the keyword, if any.

Returns:

The keyword half of a keyword=value pair.

Definition at line 359 of file RDBComment.cc.

10.4.3.10 string RDBComment::getValue (void) const

Return the keyword's value, if any.

Returns:

The value half of a keyword=value pair.

Definition at line 372 of file RDBComment.cc.

10.4.4 Friends And Related Function Documentation

10.4.4.1 istream& operator>> (istream & *is*, RDBComment & *rdbcomment*) throw (RDBErr) `[friend]`

Read comment from input stream.

Parameters:

is the input stream.

rdbcomment the comment to fill.

Returns:

A reference to the input stream.

If the line is an RDB comment line, i.e. it starts with a '#', then fill the comment. Otherwise, set ios::failbit.

Definition at line 38 of file RDBComment.cc.

10.4.4.2 ostream& operator<< (ostream & *os*, const RDBComment & *rdbcomment*) `[friend]`

Write comment to output stream.

Parameters:

os the output stream.

rdbcomment the comment to print.

Returns:

A reference to the output stream

Places the comment on the output stream. Appends a '#' character if its a plain comment or a "#:" string if its a keyword/value comment.

Definition at line 73 of file RDBComment.cc.

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

- RDBComment.h
- RDBComment.cc

10.5 RDBErr Class Reference

The parent class for all RDB related exceptions.

`#include <RDBErr.h>`

Public Member Functions

- RDBErr (const string &msg)

  Constructor.

- RDBErr ()

  Constructor.

- ∼RDBErr () throw ( )

  Destructor.

10.5.1 Detailed Description

The parent class for all RDB related exceptions.

Definition at line 33 of file RDBErr.h.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 RDBErr::RDBErr (const string & *msg*)

Constructor.

Definition at line 27 of file RDBErr.cc.

10.5.2.2 RDBErr::RDBErr ()

Constructor.

Definition at line 29 of file RDBErr.cc.

10.5.2.3 RDBErr::∼RDBErr () throw ( )

Destructor.

Definition at line 31 of file RDBErr.cc.

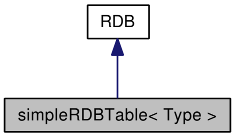
The documentation for this class was generated from the following files:

- RDBErr.h
- RDBErr.cc

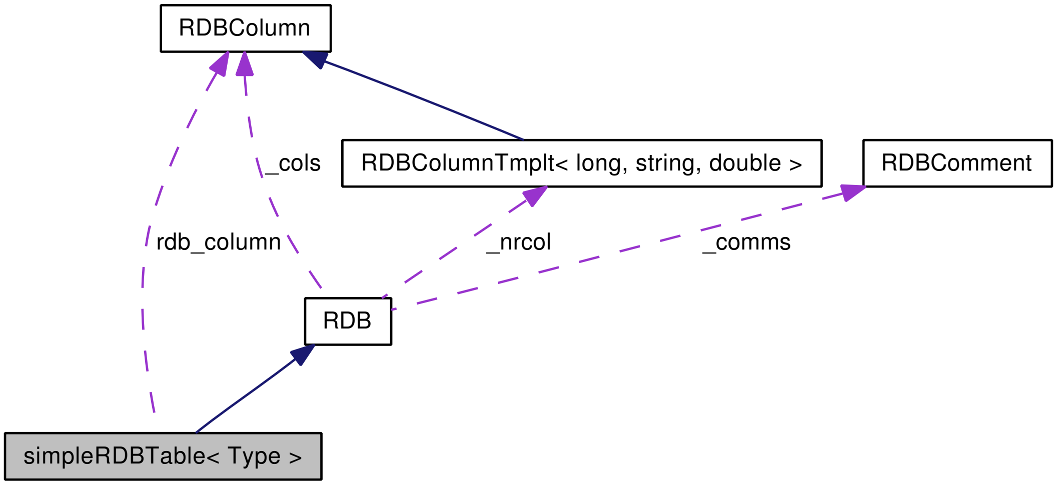
10.6 simpleRDBTable< Type > Class Template Reference

`#include <simpleRDBTable.h>`

Inheritance diagram for simpleRDBTable< Type >:



Collaboration diagram for simpleRDBTable< Type >:



Public Member Functions

- **simpleRDBTable** (const string &fname, const int mode, const char ∗header[ ]) throw (Exception)
- void **print** (ostream &os) const
- Type ∗ readRow () throw (Exception)

  Read a row of the RDB table.

Static Public Member Functions

- static void getData (RDBColumn ∗rdbcol, string &val) throw (Exception)

  Given an RDBColumn, get a string value.

- static void getData (RDBColumn ∗rdbcol, int &val) throw (Exception)

  Given an RDBColumn, get an integer value.

- static void getData (RDBColumn ∗rdbcol, long &val) throw (Exception)

  Given an RDBColumn, get a long value.

- static void getData (RDBColumn ∗rdbcol, double &val) throw (Exception)

  Given an RDBColumn, get a double value.

Protected Attributes

- const char ∗∗ **rdb_header**
- RDBColumn ∗∗ **rdb_column**

Friends

- ostream & **operator**<< (ostream &os, simpleRDBTable< Type > &a)
- ostream & **operator**<< (ostream &os, simpleRDBTable< Type > ∗a)

10.6.1 Detailed Description

template<class Type> class simpleRDBTable< Type >

A simple interface to the RDB++ read a row of an rdb table to initialize a class Type. The class Type must have a constructor with the following prototype:

Type(const char∗ header[], RDBColumn∗∗ rdb_column) throw(Exception)

Definition at line 48 of file simpleRDBTable.h.

10.6.2 Member Function Documentation

10.6.2.1 template<class Type> void simpleRDBTable< Type >::getData (RDBColumn ∗ *rdbcol*, string & *val*) throw (Exception) `[inline, static]`

Given an RDBColumn, get a string value.

Definition at line 86 of file simpleRDBTable.cc.

10.6.2.2 template<class Type> void simpleRDBTable< Type >::getData (RDBColumn ∗ *rdbcol*, int & *val*) throw (Exception) `[inline, static]`

Given an RDBColumn, get an integer value.

Definition at line 98 of file simpleRDBTable.cc.

10.6.2.3 template<class Type> void simpleRDBTable< Type >::getData (RDBColumn ∗ *rdbcol*, long & *val*) throw (Exception) `[inline, static]`

Given an RDBColumn, get a long value.

Definition at line 122 of file simpleRDBTable.cc.

10.6.2.4 template<class Type> void simpleRDBTable< Type >::getData (RDBColumn ∗ *rdbcol*, double & *val*) throw (Exception) `[inline, static]`

Given an RDBColumn, get a double value.

Definition at line 145 of file simpleRDBTable.cc.

10.6.2.5 template<class Type> Type ∗ simpleRDBTable< Type >::readRow () throw (Exception) `[inline]`

Read a row of the RDB table.

Definition at line 191 of file simpleRDBTable.cc.

References RDB::_isptr, and RDB::read().

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

- simpleRDBTable.h
- simpleRDBTable.cc

Index