# AIMDB

0.1

Generated by Doxygen 1.8.11

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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

# Chapter 2

# Class Index

## 2.1 Class List

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

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

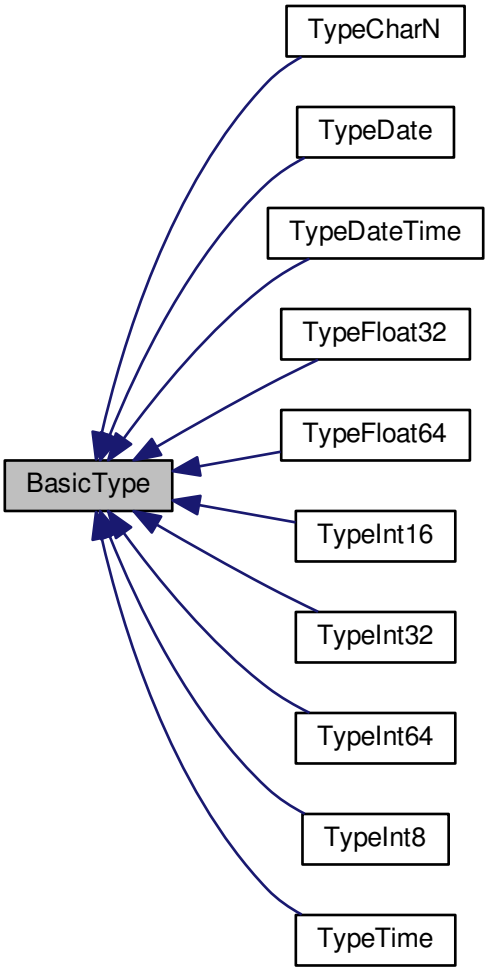
# Chapter 4

# Class Documentation

## 4.1 BasicType Class Reference

```
#include <datatype.h>
```

Inheritance diagram for BasicType:




## Public Member Functions

- BasicType (TypeCode typecode, int64_t typesize)
- virtual ~BasicType ()
- virtual bool cmpEQ (void *data1, void *data2)
- virtual bool cmpGE (void *data1, void *data2)
- virtual bool cmpGT (void *data1, void *data2)
- virtual bool cmpLE (void *data1, void *data2)
- virtual bool cmpLT (void *data1, void *data2)
- virtual int copy (void *dest, void *data)
- virtual int formatBin (void *dest, void *data)
- virtual int formatTxt (void *dest, void *data)
- virtual TypeCode getTypeCode (void)
- virtual int64_t getTypeSize (void)

### Protected Attributes

- TypeCode b_type_code
- int64_t b_type_size

### 4.1.1 Detailed Description

definition of class BasicType.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 BasicType::BasicType ( TypeCode *typecode,* int64_t *typesize* ) [inline]

constructor.

#### 4.1.2.2 virtual BasicType::~BasicType ( ) [inline], [virtual]

destructor.

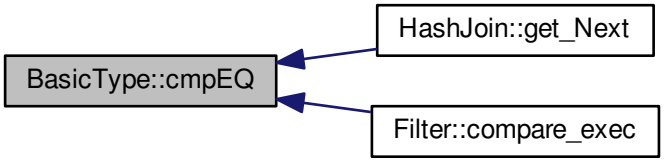
### 4.1.3 Member Function Documentation

#### 4.1.3.1 virtual bool BasicType::cmpEQ ( void * *data1,* void * *data2* ) [inline], [virtual]

equal to.

Reimplemented in TypeDateTime, TypeTime, TypeDate, TypeCharN, TypeFloat64, TypeFloat32, TypeInt64, TypeInt32, TypeInt16, and TypeInt8.

Here is the caller graph for this function:



#### 4.1.3.2 virtual bool BasicType::cmpGE ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than or equal to

Reimplemented in TypeDateTime, TypeTime, TypeDate, TypeCharN, TypeFloat64, TypeFloat32, TypeInt64, TypeInt32, TypeInt16, and TypeInt8.

Here is the caller graph for this function:


#### 4.1.3.3 virtual bool BasicType::cmpGT ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than.

Reimplemented in TypeDateTime, TypeTime, TypeDate, TypeCharN, TypeFloat64, TypeFloat32, TypeInt64, TypeInt32, TypeInt16, and TypeInt8.

Here is the caller graph for this function:


#### 4.1.3.4 virtual bool BasicType::cmpLE ( void * *data1*, void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented in TypeDateTime, TypeTime, TypeDate, TypeCharN, TypeFloat64, TypeFloat32, TypeInt64, TypeInt32, TypeInt16, and TypeInt8.

Here is the caller graph for this function:

#### 4.1.3.5 virtual bool BasicType::cmpLT ( void * *data1*, void * *data2* ) [inline], [virtual]

less than.

Reimplemented in TypeDateTime, TypeTime, TypeDate, TypeCharN, TypeFloat64, TypeFloat32, TypeInt64, TypeInt32, TypeInt16, and TypeInt8.

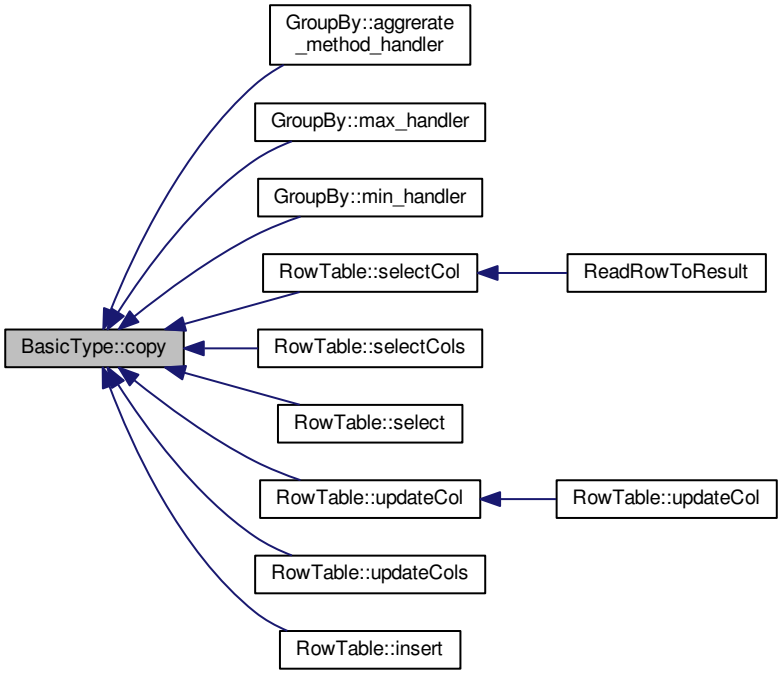
Here is the caller graph for this function:


#### 4.1.3.6 virtual int BasicType::copy ( void * *dest*, void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented in TypeDateTime, TypeTime, TypeDate, TypeCharN, TypeFloat64, TypeFloat32, TypeInt64, TypeInt32, TypeInt16, and TypeInt8.

Here is the caller graph for this function:

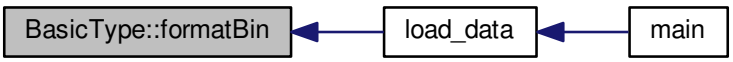


#### 4.1.3.7 virtual int BasicType::formatBin ( void * *dest*, void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented in TypeDateTime, TypeTime, TypeDate, TypeCharN, TypeFloat64, TypeFloat32, TypeInt64, TypeInt32, TypeInt16, and TypeInt8.

Here is the caller graph for this function:

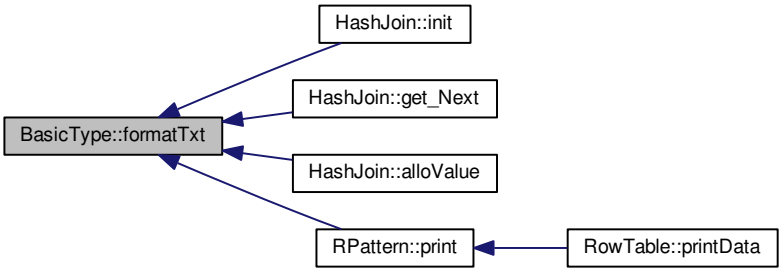



#### 4.1.3.8 virtual int BasicType::formatTxt ( void * *dest*, void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented in TypeDateTime, TypeTime, TypeDate, TypeCharN, TypeFloat64, TypeFloat32, TypeInt64, TypeInt32, TypeInt16, and TypeInt8.

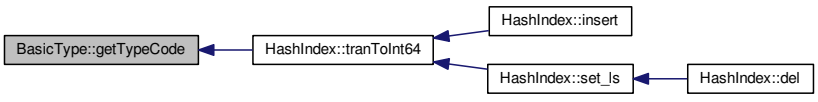
Here is the caller graph for this function:




#### 4.1.3.9 virtual TypeCode BasicType::getTypeCode ( void ) [inline], [virtual]

get type code of this data type.

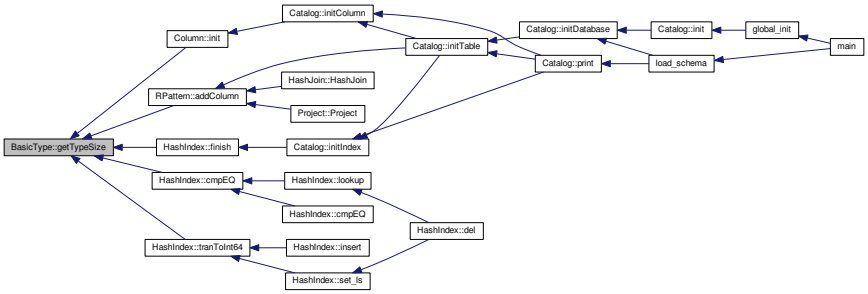
Here is the caller graph for this function:



#### 4.1.3.10 virtual int64_t BasicType::getTypeSize ( void ) [inline], [virtual]

get data size when stored in bin format.

Here is the caller graph for this function:




### 4.1.4 Member Data Documentation

#### 4.1.4.1 TypeCode BasicType::b_type_code [protected]

data type code

#### 4.1.4.2 int64_t BasicType::b_type_size [protected]

data type size

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

- system/datatype.h

## 4.2 Catalog Class Reference

`#include <catalog.h>`

### Public Member Functions

- bool createColumn (const char *name, ColumnType type, int64_t option_size, int64_t &c_id)
- bool createDatabase (const char *name, int64_t &d_id)
- bool createIndex (const char *name, IndexType type, Key i_key, int64_t &i_id)
- bool createTable (const char *name, TableType type, int64_t &t_id)
- Object * getObjById (int64_t o_id)
- Object * getObjByName (char *o_name)
- void init (void)
- bool initDatabase (int64_t d_id)
- void print (void)
- bool shut (void)
- bool shutDatabase (int64_t d_id)

### Private Member Functions

- bool initColumn (int64_t c_id)
- bool initIndex (int64_t i_id)
- bool initTable (int64_t t_id)
- int64_t obtainId (void)
- int64_t registerObj (Object *obj)

### Private Attributes

- std::vector< Object * > cl_id_obj
- std::unordered_map< std::string, Object * > cl_name_obj

## 4.2.1 Detailed Description

definition of class Catalog.

## 4.2.2 Member Function Documentation

#### 4.2.2.1 bool Catalog::createColumn ( const char * *name*, ColumnType *type*, int64_t *option_size*, int64_t & *c_id* )

create column.

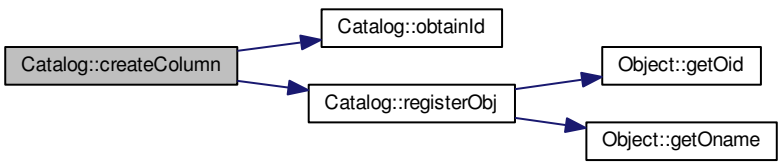
**Parameters**

| | |
|---|---|
| *name* | column name |
| *type* | column type: [INT8,INT16,...] |
| *option_size* | only work for column type CHARN(option_size) |
| *c_id* | reference of column identifier, position in cl_id_obj |

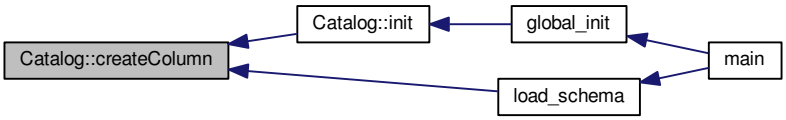
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:



Here is the caller graph for this function:




#### 4.2.2.2 bool Catalog::createDatabase ( const char * *name*, int64_t & *d_id* )

create database.

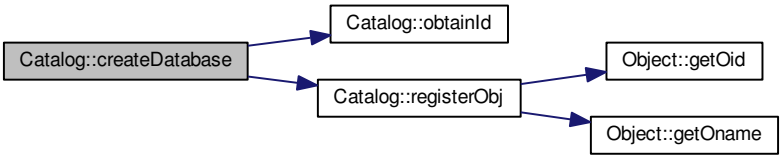
**Parameters**

| | |
|---|---|
| *name* | database name |
| *d_id* | reference of database identifier, position in cl_id_obj |

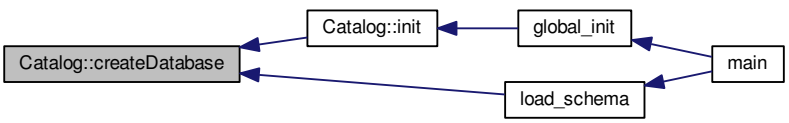
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:



#### 4.2.2.3 bool Catalog::createIndex ( const char * *name*, IndexType *type*, Key *i_key*, int64_t & *i_id* )

create index.

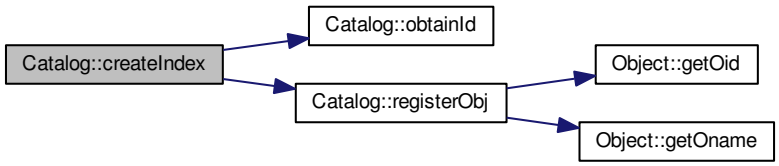
**Parameters**

| | |
|---|---|
| *name* | index name |
| *type* | index type: [HASHINDEX,BPTREEINDEX,ARTTREEINDEX] |
| *i_key* | stores column identifiers of this index |
| *i_id* | reference of index identifier, position in cl_id_obj |

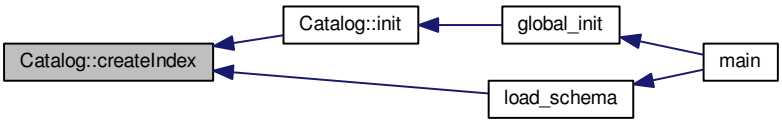
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.2.2.4 bool Catalog::createTable ( const char * *name*, TableType *type*, int64_t & *t_id* )

create table.

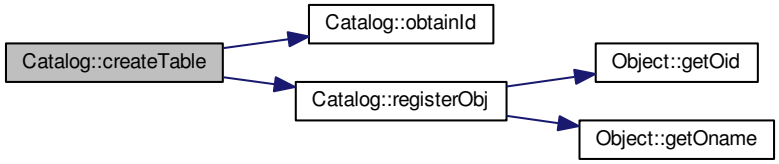
**Parameters**

| | |
|---|---|
| *name* | table name |
| *type* | tabletype: [ROWTABLE,COLUMNTABLE] |
| *t_id* | reference of table identifier, position in cl_id_obj |

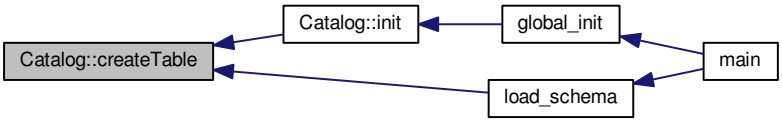
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.2.2.5 Object * Catalog::getObjById ( int64_t *o_id* )

get object[DATABASE,TABLE,COLUMN,INDEX] by identifier

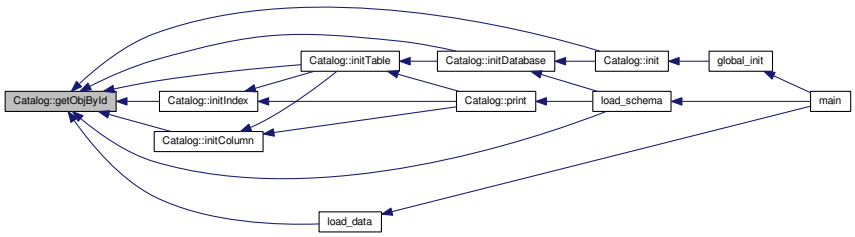
**Parameters**

| | |
|---|---|
| *o↩ _id* | identifier of object |

**Return values**

| | |
|---|---|
| *!=* | NULL available |
| *==* | NULL unavaliable, deleted or not exist |

Here is the caller graph for this function:




#### 4.2.2.6 Object * Catalog::getObjByName ( char * *o_name* )

get object[DATABASE,TABLE,COLUMN,INDEX] by object name

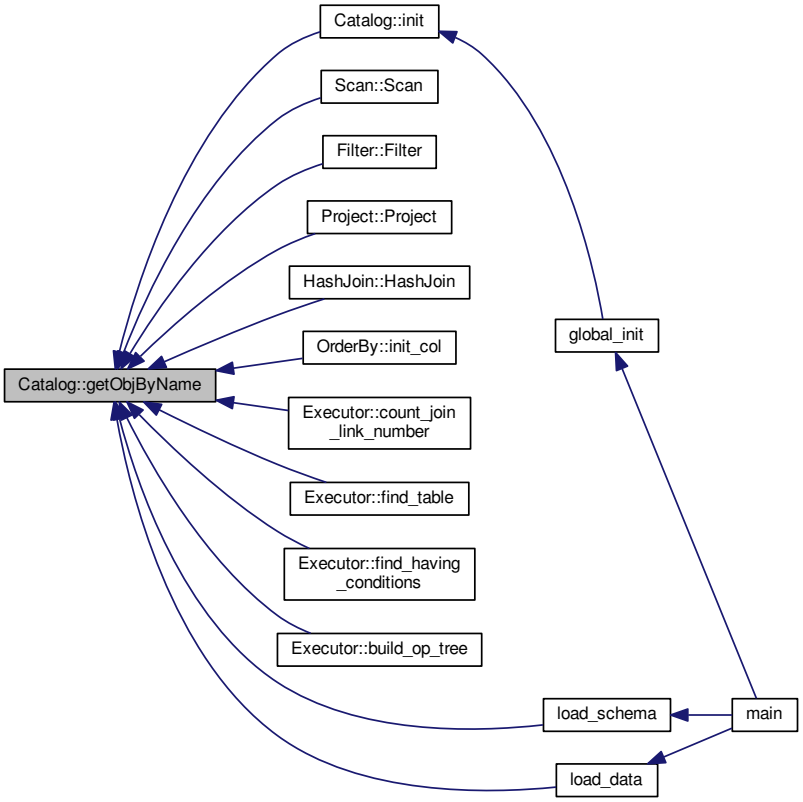
**Parameters**

| | |
|---|---|
| *name* | of an object |

**Return values**

| | |
|---|---|
| *!=* | NULL available |
| *==* | NULL unavaliable, deleted or not exist |

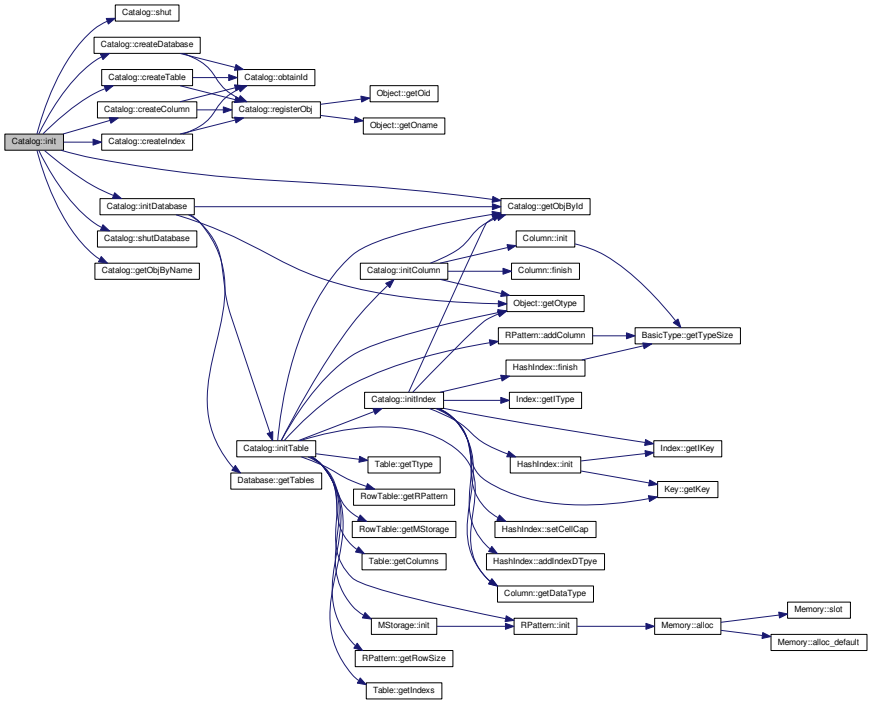
Here is the caller graph for this function:



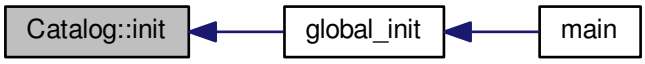

#### 4.2.2.7 void Catalog::init ( void ) [inline]

init operation.

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.2.2.8 bool Catalog::initColumn ( int64_t *c_id* ) [private]

init column

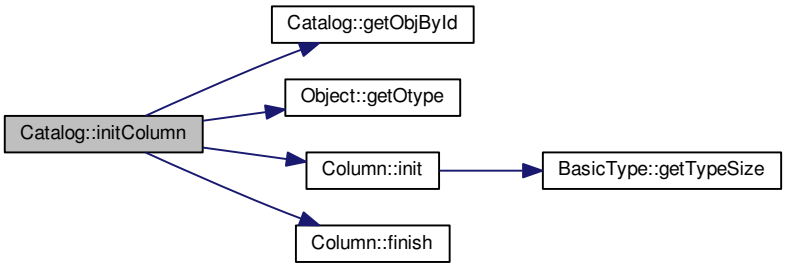
**Parameters**

| | |
|---|---|
| *c_id* | which column to prepare |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:


#### 4.2.2.9 bool Catalog::initDatabase ( int64_t *d_id* )

init database, very important, after all setting, call initDatabase to get this database in work

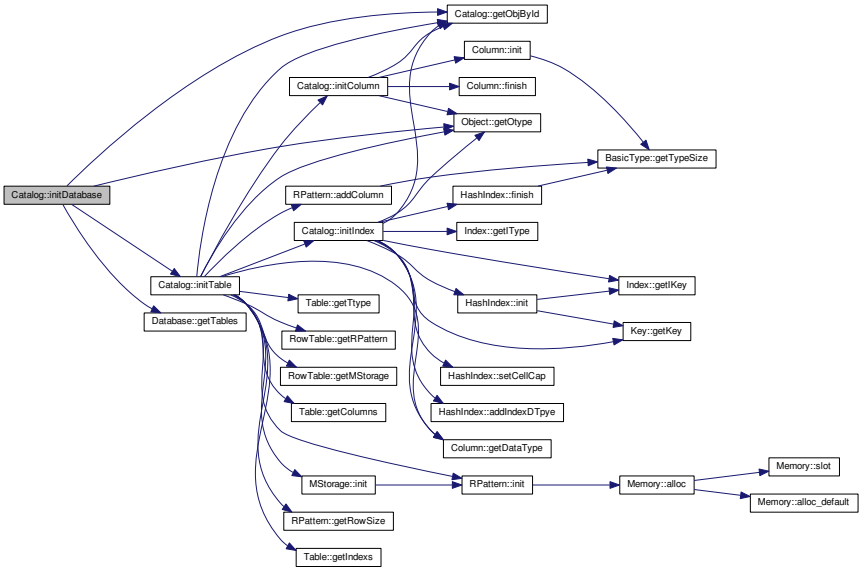
**Parameters**

| | |
|---|---|
| *d_id* | which database to prepare |

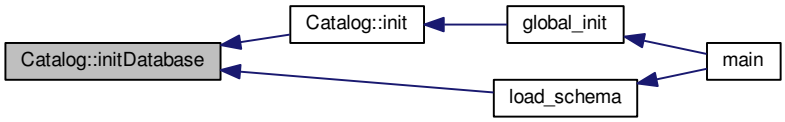
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.2.2.10 bool Catalog::initIndex ( int64_t *i_id* ) [private]

init index

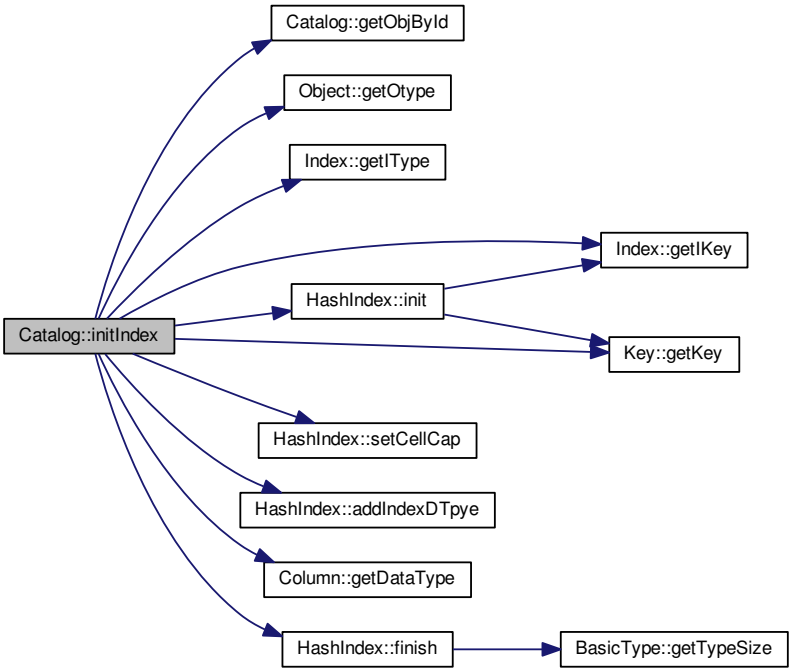
**Parameters**

| | |
|---|---|
| *i_id* | which index to prepare |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:


#### 4.2.2.11 bool Catalog::initTable ( int64_t *t_id* ) [private]

init table

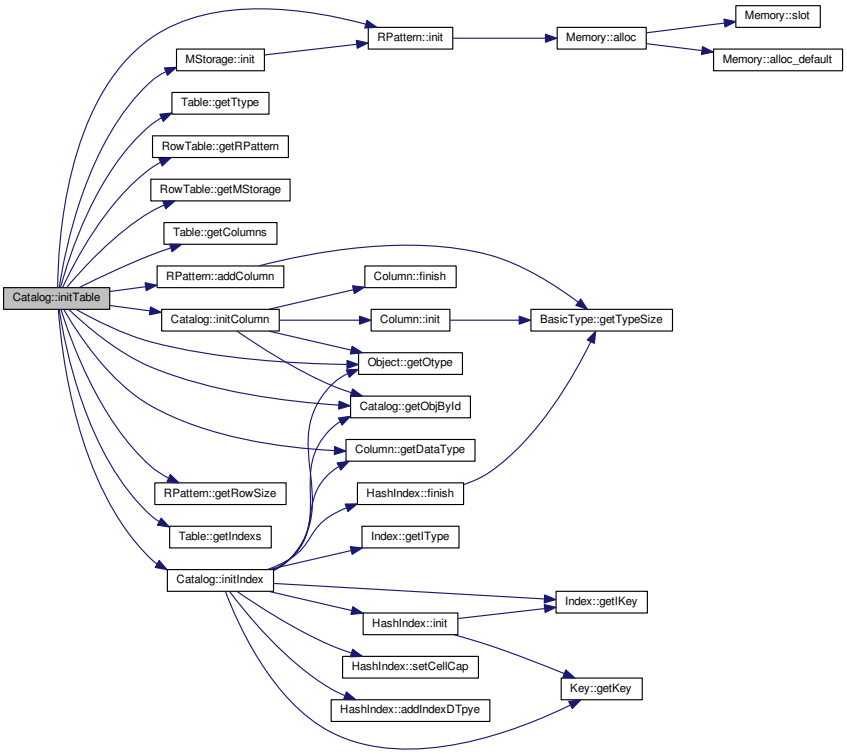
**Parameters**

| | |
|---|---|
| *t_id* | which table to prepare |

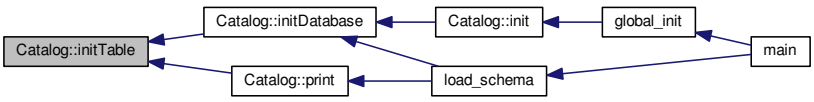
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:



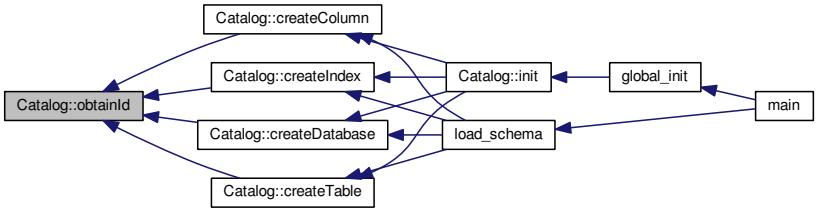

#### 4.2.2.12 int64_t Catalog::obtainId ( void ) [inline],[private]

get a free object identifier

**Return values**

| | |
|---|---|
| *>0* | success |
| *<=0* | failure |

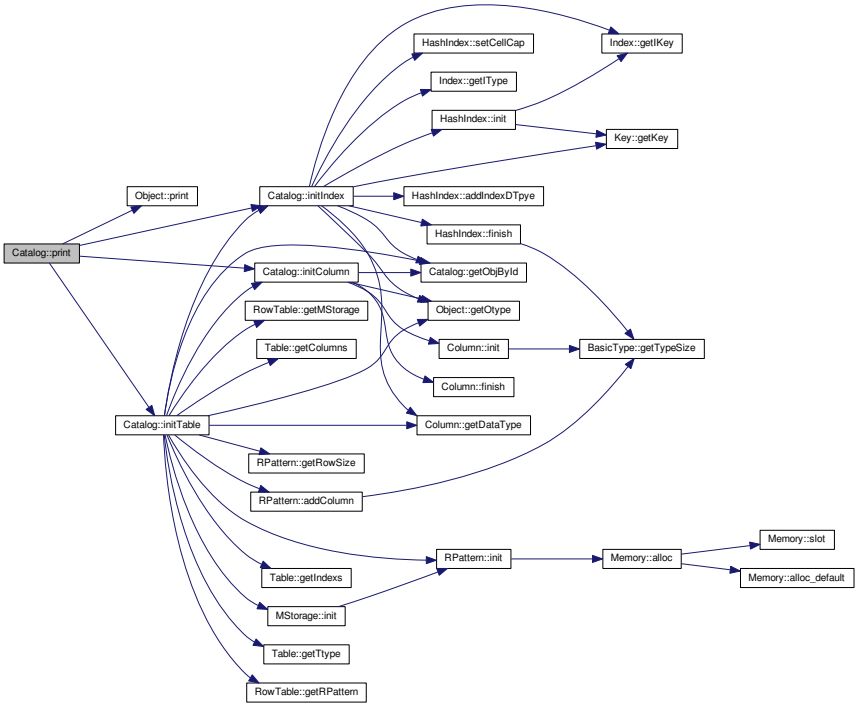
Here is the caller graph for this function:



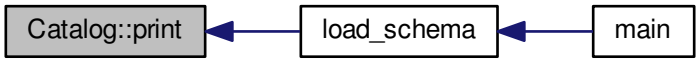

#### 4.2.2.13 void Catalog::print ( void ) [inline]

print the catalog

Here is the call graph for this function:




Here is the caller graph for this function:



#### 4.2.2.14 int64_t Catalog::registerObj ( Object * *obj* ) [inline],[private]

put object in cl_id_obj and cl_name_obj

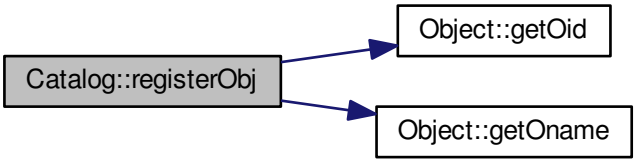
**Parameters**

| | |
|---|---|
| *obj* | pointer of object to put |

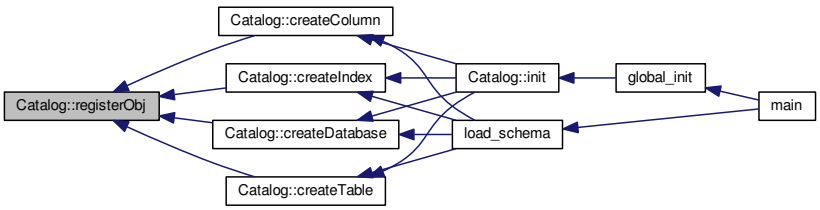
**Return values**

| | |
|---|---|
| *>0* | success |
| *<=0* | failure |

Here is the call graph for this function:



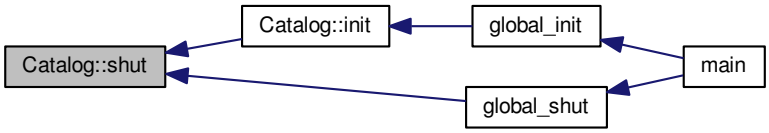

Here is the caller graph for this function:




#### 4.2.2.15 bool Catalog::shut ( void )

shut down, free all memory of Objects.

Here is the caller graph for this function:




#### 4.2.2.16 bool Catalog::shutDatabase ( int64_t *d_id* )

shutdowm database

**Parameters**

| | |
|---|---|
| *d_id* | which database to shut |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the caller graph for this function:


### 4.2.3 Member Data Documentation

#### 4.2.3.1 std::vector<Object *> Catalog::cl_id_obj [private]

container of Objects

#### 4.2.3.2 std::unordered_map<std::string, Object *> Catalog::cl_name_obj [private]

name map of Objects ,name should not be the same in the whole Catalog

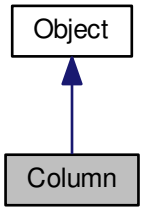
The documentation for this class was generated from the following files:

- system/catalog.h
- system/catalog.cc

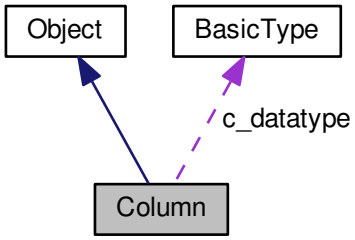
## 4.3 Column Class Reference

`#include <schema.h>`

Inheritance diagram for Column:




Collaboration diagram for Column:




### Public Member Functions

- Column (int64_t c_id, const char *c_name, ColumnType c_type, int64_t c_size=0)
- virtual ~Column (void)
- virtual bool finish (void)
- int64_t getCSize (void)
- ColumnType getCType (void)
- BasicType * getDataType (void)
- virtual bool init (void)
- virtual void print (void)
- virtual bool shut (void)

### Private Attributes

- BasicType * c_datatype
- int64_t c_size
- ColumnType c_type

### 4.3.1 Detailed Description

definition of class Column.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 Column::Column ( int64_t *c_id*, const char * *c_name*, ColumnType *c_type*, int64_t *c_size* = 0 ) [inline]

constructor.

**Parameters**

| | |
|---|---|
| *c_id* | column identiier |
| *c_name* | column name |
| *c_type* | column type |
| *c_size* | data type size |

#### 4.3.2.2 virtual Column::~Column ( void ) [inline], [virtual]

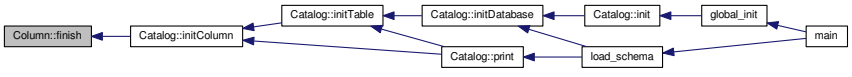
destructor.

### 4.3.3 Member Function Documentation

#### 4.3.3.1 virtual bool Column::finish ( void ) [inline], [virtual]

finish column setting.

Here is the caller graph for this function:




#### 4.3.3.2 int64_t Column::getCSize ( void ) [inline]

get data dize.

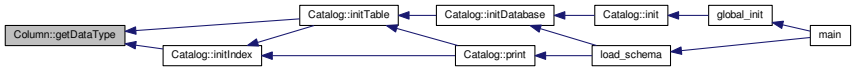
#### 4.3.3.3 ColumnType Column::getCType ( void ) [inline]

get column type.

#### 4.3.3.4 BasicType* Column::getDataType ( void ) [inline]

get data type of the column

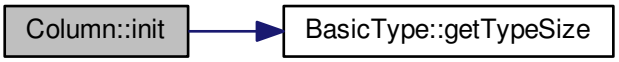
Here is the caller graph for this function:



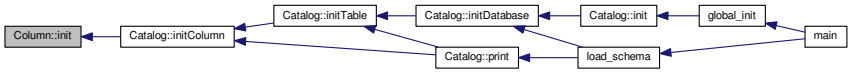

#### 4.3.3.5 virtual bool Column::init ( void ) [inline], [virtual]

init column.

Here is the call graph for this function:




Here is the caller graph for this function:

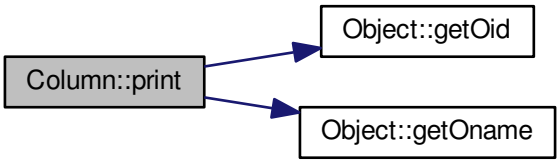



#### 4.3.3.6 virtual void Column::print ( void ) [inline], [virtual]

print column information

Reimplemented from Object.

Here is the call graph for this function:



#### 4.3.3.7 virtual bool Column::shut ( void ) [inline], [virtual]

shut down column.

Reimplemented from Object.

### 4.3.4 Member Data Documentation

#### 4.3.4.1 BasicType* Column::c_datatype [private]

column data type

#### 4.3.4.2 int64_t Column::c_size [private]

column size

#### 4.3.4.3 ColumnType Column::c_type [private]

column type

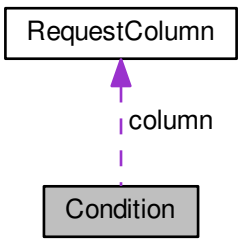
The documentation for this class was generated from the following file:

- system/schema.h

## 4.4 Condition Struct Reference

`#include <executor.h>`

Collaboration diagram for Condition:



### Public Attributes

- RequestColumn column
- CompareMethod compare
- char value [128]

### 4.4.1 Detailed Description

definition of compare condition.

### 4.4.2 Member Data Documentation

#### 4.4.2.1 RequestColumn Condition::column

which column

#### 4.4.2.2 CompareMethod Condition::compare

which method

#### 4.4.2.3 char Condition::value[128]

the value to compare with, if compare==LINK,value is another column's name; else it's the column's value

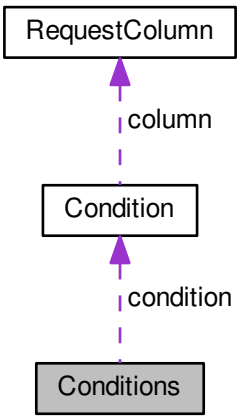
The documentation for this struct was generated from the following file:

- system/executor.h

## 4.5 Conditions Struct Reference

`#include <executor.h>`

Collaboration diagram for Conditions:



### Public Attributes

- Condition condition [4]
- int condition_num

### 4.5.1 Detailed Description

definition of conditions.

### 4.5.2 Member Data Documentation

#### 4.5.2.1 Condition Conditions::condition[4]

support maximum 4 & conditions

#### 4.5.2.2 int Conditions::condition_num

number of condition in use

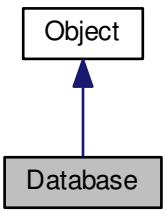
The documentation for this struct was generated from the following file:

- system/executor.h

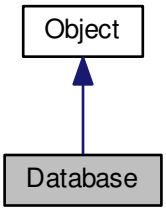
## 4.6 Database Class Reference

```
#include <schema.h>
```

Inheritance diagram for Database:



Collaboration diagram for Database:




### Public Member Functions

- Database (int64_t d_id, const char *d_name)
- virtual ~Database (void)
- virtual bool addTable (int64_t table_id)
- virtual bool finish (void)
- std::vector< int64_t > & getTables (void)
- virtual bool init (void)
- virtual bool insert (int64_t table_id, char *source)
- virtual bool insert (int64_t table_id, char *columns[ ])
- virtual bool loadData (int64_t table_id, const char *filename)
- virtual void print (void)
- virtual bool shut (void)

### Private Attributes

- std::vector< int64_t > d_table

## 4.6.1 Detailed Description

definition of class Database.

## 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 Database::Database ( int64_t *d_id,* const char * *d_name* ) [inline]

constructor.

**Parameters**

| | |
|---|---|
| *d_id* | database identifier |
| *d_name* | database name |

#### 4.6.2.2 virtual Database::~Database ( void ) [inline], [virtual]

destructor.

### 4.6.3 Member Function Documentation

#### 4.6.3.1 virtual bool Database::addTable ( int64_t *table_id* ) [inline], [virtual]

add table identifier to this database.

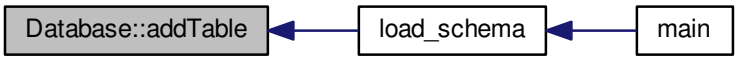
**Parameters**

| | |
|---|---|
| *table_id* | table identifier |

**Return values**

| | |
|---|---|
| *true* | success |
| *flase* | failure |

Here is the caller graph for this function:



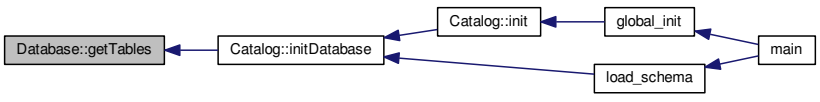

#### 4.6.3.2 virtual bool Database::finish ( void ) [inline], [virtual]

finish, important interface for son class

#### 4.6.3.3 std::vector< int64_t >& Database::getTables ( void ) [inline]

get table identifier container.

Here is the caller graph for this function:



#### 4.6.3.4 virtual bool Database::init ( void ) [inline], [virtual]

init, important interface for son class

#### 4.6.3.5 virtual bool Database::insert ( int64_t *table_id,* char * *source* ) [inline], [virtual]

insert a record to this database's table.

**Parameters**

| | |
|---|---|
| *table_id* | table identifier |
| *source* | buffer of data to insert |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

#### 4.6.3.6 virtual bool Database::insert ( int64_t *table_id,* char * *columns[ ]* ) [inline], [virtual]

insert a record to this database's table.

**Parameters**

| | |
|---|---|
| *table_id* | table identifier |
| *columns* | each element of columns is a pointer to data of the column |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

#### 4.6.3.7 virtual bool Database::loadData ( int64_t *table_id,* const char * *filename* ) [inline], [virtual]

load data into table in this database(not use).

**Parameters**

| | |
|---|---|
| *table_id* | table identifier |
| *filename* | data file to load |

**Return values**

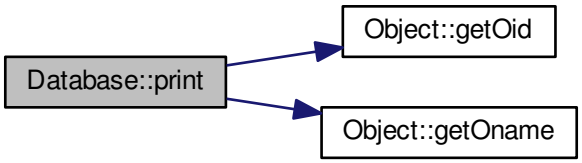
| | |
|---|---|
| *true* | success |
| *false* | failure |

#### 4.6.3.8 virtual void Database::print ( void ) [inline], [virtual]

print database information

Reimplemented from Object.

Here is the call graph for this function:




#### 4.6.3.9 virtual bool Database::shut ( void ) [inline], [virtual]

shut down this database, free all memory.

Reimplemented from Object.

### 4.6.4 Member Data Documentation

#### 4.6.4.1 std::vector< int64_t > Database::d_table [private]

table identifier container.

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

- system/schema.h

## 4.7 ErrorLog Class Reference

an array of source file names

`#include <errorlog.h>`

### Public Member Functions

- ErrorLog (const char *thread_name, int msg_cap=256 *1024)
- ~ErrorLog ()
- int getErrorCode (void)
- const char * getErrorMsg (void)
- void log (int level, const char *src_name, const int lineno,...)
- void reset ()

### Static Public Member Functions

- static void closeLog (void)
- static void flushLog (void)
- static const char * id2Name (int src_id)
- static void init (int level, const char *logfile)
- static int name2Id (const char *src_name)
- static void setLevel (int level)

### Static Public Attributes

- static int el_level
  *logging level*
- static const char * el_level_name [EL_SERIOUS+1]
  *level => name*

### Private Member Functions

- int getFuncNameGCC (char *bt_symbol)

### Private Attributes

- void * el_bt_buffer [256]
  *allow up to 256 levels of calls*
- char * el_demangle_buf
  *buffer needed for demangle*
- size_t el_demangle_len
  *demangle buffer length*
- int el_err_code
  *current error code*
- char * el_msg_buf
  *a buffer to hold the error message*
- int el_msg_cap
  *the message buffer size*
- char * el_msg_cur
  *point to the end of the message*
- char * el_thread_name
  *the thread name*
- time_t el_tloc
  *local time in seconds at last message*
- struct tm el_tm
  *broken down time at last message*

### Static Private Attributes

- static FILE * el_fp
  *file handle of el_logfile*
- static pthread_mutex_t el_lock
  *protect the global states*
- static char * el_logfile
  *file path to write log to*
- static std::unordered_map< std::string, int > * el_name_2_id
  *file name => id*

### 4.7.1 Detailed Description

an array of source file names

### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 ErrorLog::ErrorLog ( const char * *thread_name,* int *msg_cap* = 256 * 1024 )

constructor

**Parameters**

| | |
|---|---|
| *threadid* | the current thread id to generate the log for |
| *level* | the dynamic logging level |
| *logfile* | if not NULL then specify the log file path |

#### 4.7.2.2 ErrorLog::~ErrorLog ( )

destructor

### 4.7.3 Member Function Documentation

#### 4.7.3.1 static void ErrorLog::closeLog ( void ) [inline], [static]

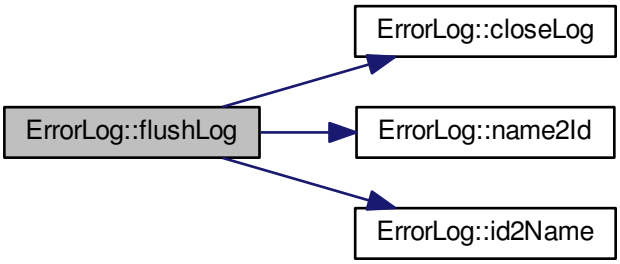
close the log file

Here is the caller graph for this function:


#### 4.7.3.2 static void ErrorLog::flushLog ( void ) [inline], [static]

flush the log file

Here is the call graph for this function:




#### 4.7.3.3 int ErrorLog::getErrorCode ( void ) [inline]

return the last error code

#### 4.7.3.4 const char* ErrorLog::getErrorMsg ( void ) [inline]

return the error message accumulated since last reset()

#### 4.7.3.5 int ErrorLog::getFuncNameGCC ( char * *bt_symbol* ) [private]

Parse the symbol returned from the backtrace_symbols() call. Then demangle the function name if necessary. Note this implementation is GCC specific.

A symbol has the following format:

```
binary(function+offset) [return address]
```

**Parameters**

| | |
|---|---|
| *bt_symbol* | a symbol returned from backtrace_symbols() |

**Return values**

| | |
|---|---|
| *0* | success, output is in el_demangle_buf |
| *-1* | function name is not available |

Here is the caller graph for this function:


#### 4.7.3.6 const char * ErrorLog::id2Name ( int *src_id* ) [static]

get the file name for a given id

**Parameters**

| | |
|---|---|
| *src_id* | the file id |

**Return values**

| | |
|---|---|
| *!=NULL* | the file name |
| *==NULL* | the file id is out of range |

Here is the caller graph for this function:

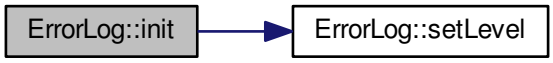

#### 4.7.3.7 void ErrorLog::init ( int *level,* const char * *logfile* ) [static]

global initiator

**Parameters**

| | |
|---|---|
| *level* | the dynamic logging level |
| *logfile* | if not NULL then specify the log file path |

Here is the call graph for this function:



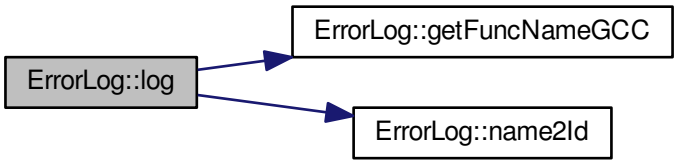

#### 4.7.3.8 void ErrorLog::log ( int *level*, const char * *src_name*, const int *lineno*, ... )

Log the message. Stack trace will be generated for error and serious messages.

**Parameters**

| | |
|---|---|
| *level* | the level of the message |
| *src_name* | the source file name |
| *lineno* | the line number in the source file |
| ... | printf-like format string and arguments |

Here is the call graph for this function:




#### 4.7.3.9 int ErrorLog::name2Id ( const char * *src_name* ) [static]

get fileid for a given file name

**Parameters**

| | |
|---|---|
| *src_name* | the file name |

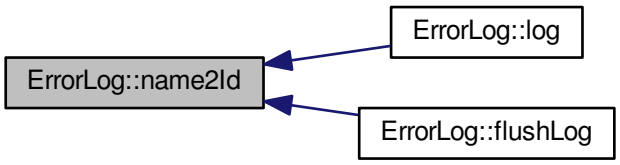
**Return values**

| | |
|---|---|
| *>=0* | the file id |

**Return values**

| | |
|---|---|
| *<0* | the file name does not exist |

Here is the caller graph for this function:




#### 4.7.3.10 void ErrorLog::reset ( )

Clear current error messages. Call this before executing an operation.

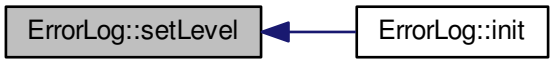
#### 4.7.3.11 void ErrorLog::setLevel ( int *level* ) [static]

set the dynamice logging level (this must be at least EL_LEVEL_COMPILE) This method is thread safe.

**Parameters**

| | |
|---|---|
| *level* | the dynamic logging level |

Here is the caller graph for this function:




### 4.7.4 Member Data Documentation

#### 4.7.4.1 void* ErrorLog::el_bt_buffer[256] [private]

allow up to 256 levels of calls

#### 4.7.4.2 char* ErrorLog::el_demangle_buf [private]

buffer needed for demangle

#### 4.7.4.3 size_t ErrorLog::el_demangle_len [private]

demangle buffer length

#### 4.7.4.4 int ErrorLog::el_err_code [private]

current error code

#### 4.7.4.5 FILE * ErrorLog::el_fp [static], [private]

file handle of el_logfile

#### 4.7.4.6 int ErrorLog::el_level [static]

logging level

#### 4.7.4.7 const char * ErrorLog::el_level_name [static]

level => name

#### 4.7.4.8 pthread_mutex_t ErrorLog::el_lock [static], [private]

protect the global states

#### 4.7.4.9 char * ErrorLog::el_logfile [static], [private]

file path to write log to

#### 4.7.4.10 char* ErrorLog::el_msg_buf [private]

a buffer to hold the error message

#### 4.7.4.11 int ErrorLog::el_msg_cap [private]

the message buffer size

#### 4.7.4.12 char* ErrorLog::el_msg_cur [private]

point to the end of the message

#### 4.7.4.13 std::unordered_map< std::string, int > * ErrorLog::el_name_2_id [static], [private]

file name => id

#### 4.7.4.14 char* ErrorLog::el_thread_name [private]

the thread name

#### 4.7.4.15 time_t ErrorLog::el_tloc [private]

local time in seconds at last message

#### 4.7.4.16 struct tm ErrorLog::el_tm [private]

broken down time at last message

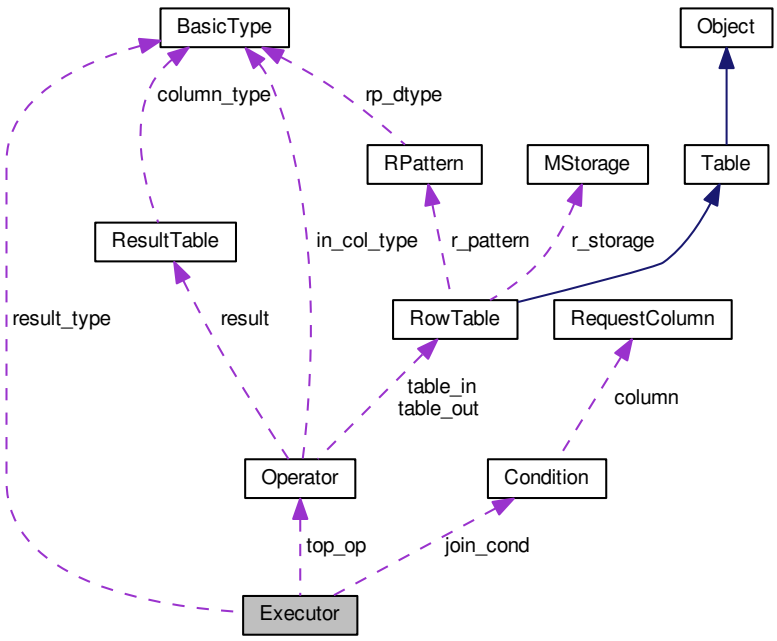
The documentation for this class was generated from the following files:

- system/errorlog.h
- system/errorlog.cc

## 4.8 Executor Class Reference

`#include <executor.h>`

Collaboration diagram for Executor:



### Public Member Functions

- virtual int close ()
  *close function.*
- virtual int exec (SelectQuery *query, ResultTable *result)
  *exec function.*

### Private Member Functions

- void build_op_tree (Operator **Op, SelectQuery *query)
  *build_op_tree*
- void count_join_link_number (SelectQuery *query)
  *count join link number*
- void find_having_conditions (SelectQuery *query)
  *find_having_conditions*
- void find_table (SelectQuery *query)
  *find tables that filter columns belong to*

### Private Attributes

- int64_t count
- int64_t * filter_tid
- int64_t * having_tid
- Condition ** join_cond
- int join_count
- int64_t * joinA_tid
- int64_t * joinB_tid
- BasicType ** result_type
- int64_t timesin
- Operator * top_op = NULL

## 4.8.1 Member Function Documentation

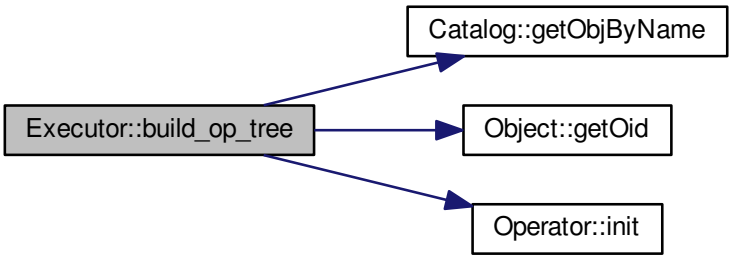
#### 4.8.1.1 void Executor::build_op_tree ( Operator ** *Op,* SelectQuery * *query* ) [inline], [private]

build_op_tree

**Parameters**

| | |
|---|---|
| *selected* | query, operator |

Here is the call graph for this function:




#### 4.8.1.2 int Executor::close ( void ) [virtual]

close function.

**Parameters**

| | |
|---|---|
| *None* | |

**Return values**

| | |
|---|---|
| *==0* | succeed to close |
| *!=0* | fail to close |

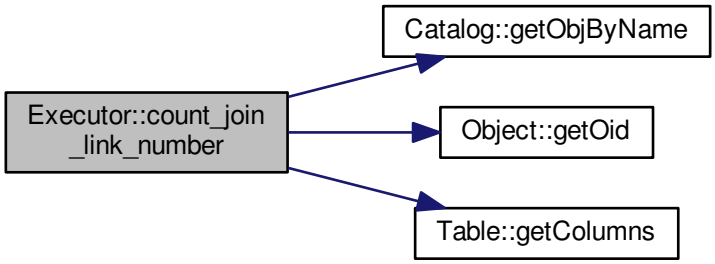
#### 4.8.1.3 void Executor::count_join_link_number ( SelectQuery * *query* ) [inline], [private]

count join link number

**Parameters**

| | |
|---|---|
| *selected* | query |

Here is the call graph for this function:




#### 4.8.1.4 int Executor::exec ( SelectQuery * *query*, ResultTable * *result* ) [virtual]

exec function.

**Parameters**

| | |
|---|---|
| *query* | to execute, if NULL, execute query at last time |

**Returns**

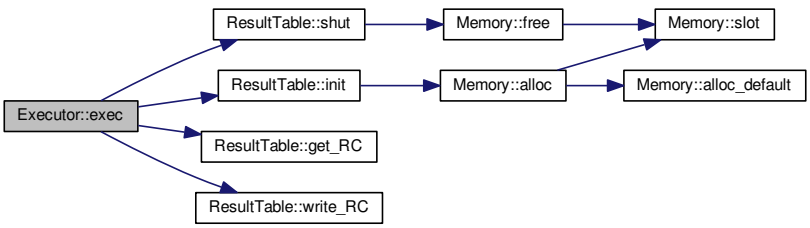
result table generated by an execution, store result in pattern defined by the result table

**Return values**

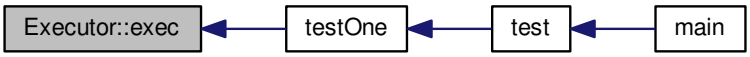
| | |
|---|---|
| *>0* | number of result rows stored in result |
| *<=0* | no more result |

exeutor function

Here is the call graph for this function:



Here is the caller graph for this function:



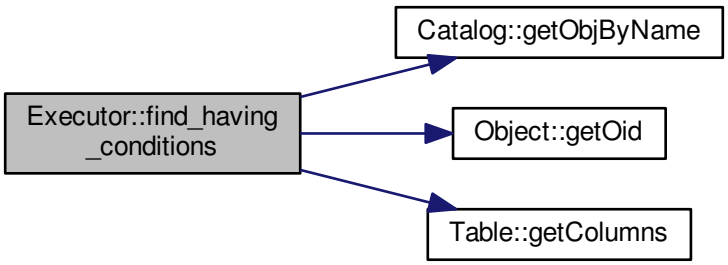

#### 4.8.1.5 void Executor::find_having_conditions ( SelectQuery * *query* ) [inline], [private]

find_having_conditions

**Parameters**

| | |
|---|---|
| *selected* | query |

Here is the call graph for this function:



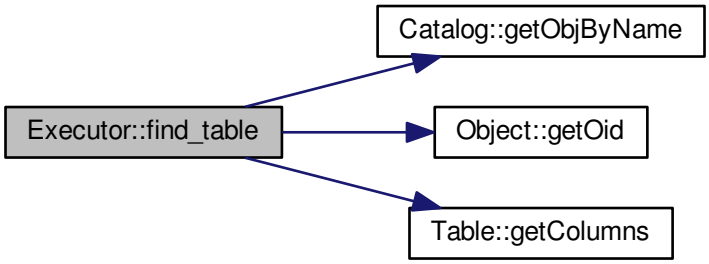

#### 4.8.1.6 void Executor::find_table ( SelectQuery * *query* ) [inline], [private]

find tables that filter columns belong to

**Parameters**

| | |
|---|---|
| *selected* | query |

Here is the call graph for this function:




### 4.8.2 Member Data Documentation

#### 4.8.2.1 int64_t Executor::count [private]

count records for debug uses

#### 4.8.2.2 int64_t* Executor::filter_tid [private]

Table id of filter conditions.

#### 4.8.2.3 int64_t* Executor::having_tid [private]

inside class use

#### 4.8.2.4 Condition** Executor::join_cond [private]

Conditions for join.

#### 4.8.2.5 int Executor::join_count [private]

inside class use

#### 4.8.2.6 int64_t* Executor::joinA_tid [private]

inside class use

#### 4.8.2.7 int64_t* Executor::joinB_tid [private]

inside class use

#### 4.8.2.8 BasicType** Executor::result_type [private]

Type of every column in result table.

#### 4.8.2.9 int64_t Executor::timesin [private]

count for enter func exec times

#### 4.8.2.10 Operator* Executor::top_op = NULL [private]

Top operator of the operator tree.

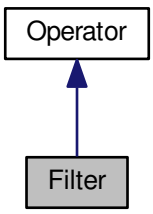
The documentation for this class was generated from the following files:

- system/executor.h
- system/executor.cc

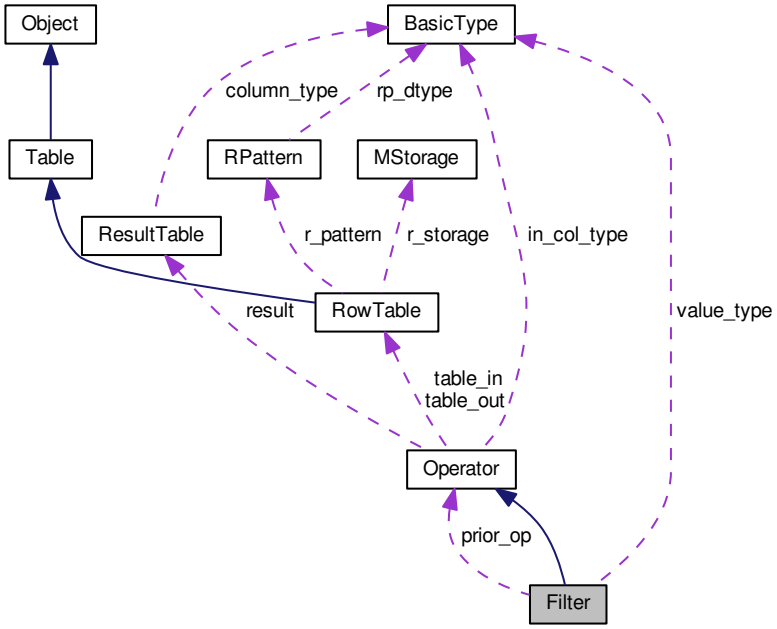
## 4.9 Filter Class Reference

```
#include <executor.h>
```

Inheritance diagram for Filter:



Collaboration diagram for Filter:




## Public Member Functions

- Filter (Operator *Op, Condition *condi)
- bool close ()
- bool compare_exec (char *cmpSrcA_ptr, char *cmpSrcB_ptr)
- bool get_Next (ResultTable *result)
- bool init ()
- bool is_End ()
- void MemShut ()
- bool resCopy (bool flag, ResultTable *result)
- void resinit ()

## Private Attributes

- int64_t col_rank
- CompareMethod compare_method
- Operator * prior_op
- char value [128]
- BasicType * value_type

## Additional Inherited Members

### 4.9.1 Detailed Description

definition of filter operaotr

### 4.9.2 Constructor & Destructor Documentation

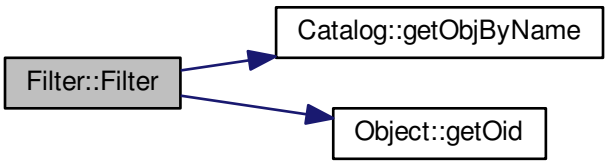
#### 4.9.2.1 Filter::Filter ( Operator * *Op*, Condition * *condi* )

construction of filter Operator

**Parameters**

| | |
|---|---|
| *Op* | prior_op that needs to inherit |
| *condi* | the filter conditions. |

Here is the call graph for this function:




### 4.9.3 Member Function Documentation

#### 4.9.3.1 bool Filter::close ( void ) [virtual]

close operator and release memory

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

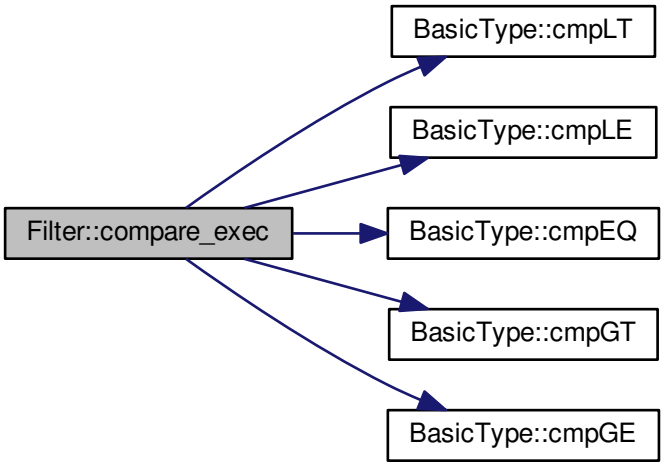
#### 4.9.3.2 bool Filter::compare_exec ( char * *cmpSrcA_ptr*, char * *cmpSrcB_ptr* ) [inline]

check the result of cmp

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:




#### 4.9.3.3 bool Filter::get_Next ( ResultTable * *result* ) [virtual]

get next record

**Parameters**

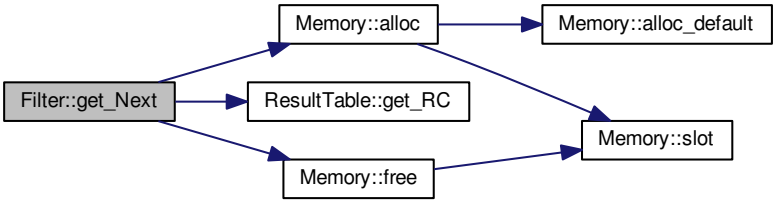
| | |
|---|---|
| *result* | the buffer to store result. |

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

Here is the call graph for this function:



#### 4.9.3.4 bool Filter::init ( void ) [virtual]

init operator

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

#### 4.9.3.5 bool Filter::is_End ( void ) [virtual]

judge whether is end

**Return values**

| | |
|---|---|
| *false* | not end |
| *true* | run end |

Implements Operator.

#### 4.9.3.6 void Filter::MemShut ( ) [inline]

shut memory

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

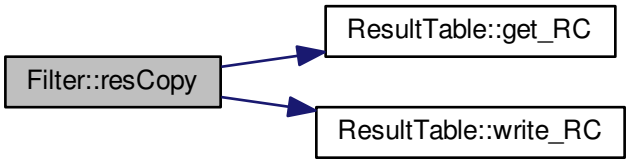
#### 4.9.3.7 bool Filter::resCopy ( bool *flag*, ResultTable * *result* ) [inline]

copy result

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:




#### 4.9.3.8 void Filter::resinit ( ) [inline]

init filter resulttable

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:


### 4.9.4 Member Data Documentation

#### 4.9.4.1 int64_t Filter::col_rank [private]

which column need to be filtered.

#### 4.9.4.2 CompareMethod Filter::compare_method [private]

compare methods

#### 4.9.4.3 Operator* Filter::prior_op [private]

operator of prior operation.

#### 4.9.4.4 char Filter::value[128] [private]

const value

#### 4.9.4.5 BasicType* Filter::value_type [private]

column types of filter columns

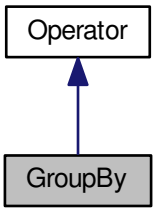
The documentation for this class was generated from the following files:

- system/executor.h
- system/executor.cc

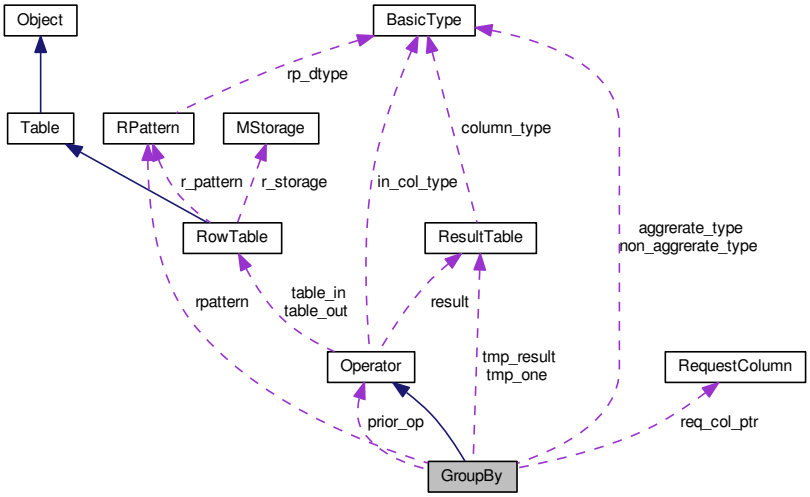
## 4.10 GroupBy Class Reference

`#include <executor.h>`

Inheritance diagram for GroupBy:




Collaboration diagram for GroupBy:



## Public Member Functions

- GroupBy (Operator *Op, int groupby_num, RequestColumn req_col[4])

  *construction of OrderBy*
- bool close ()

  *close operator and release memory.*
- bool get_Next (ResultTable *result)

  *get next record of operator*
- bool init ()

  *init of project*
- bool is_End ()

  *judge whether is end*

## Private Member Functions

- bool aggrerate (AggrerateMethod method, uint64_t probe_result, int agg_i)

  *do aggregation on a column*
- void aggrerate_method_handler (int64_t hash_count)

  *aggrerate_method handler*
- bool avg_handler ()

  *avg handler*
- int64_t get_prior_col_rank (int64_t object_oid)

  *get prior column rank*
- Operator * get_prior_operator ()

  *get pointer to prior Operator*
- RPattern get_prior_rpattern ()

  *get prior column rank*
- RowTable * get_prior_tableout ()

  *get prior tableout*
- RPattern get_rpattern ()

  *get rpattern*
- int64_t hash (char *str, int64_t length)

  *get hash from a string with length*
- void init_aggre (int64_t col_rank)

  *init aggrerate*
- void init_non_aggre (int64_t col_rank)

  *init non_aggre*
- void max_handler ()
- void min_handler ()
- void sum_handler ()

  *sum handler*

### Private Attributes

- int agg_i
- int aggrerate_index = 0
- AggrerateMethod aggrerate_method [4]
- int aggrerate_num = 0
- size_t aggrerate_off [4]
- BasicType * aggrerate_type [4]
- int col_num = 0
- char * datain
- char * dataout
- int index = 0
- int non_aggrerate_index = 0
- int non_aggrerate_num = 0
- size_t non_aggrerate_off [4]
- BasicType * non_aggrerate_type [4]
- int64_t * pattern_count_ptr
- Operator * prior_op
- RequestColumn * req_col_ptr
- int64_t row_size
- RPattern rpattern
- ResultTable tmp_one
- ResultTable tmp_result

### Additional Inherited Members

## 4.10.1 Detailed Description

definition of GroupBy

## 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 GroupBy::GroupBy ( Operator * *Op,* int *groupby_num,* RequestColumn *req_col[4]* )

construction of OrderBy

**Parameters**

| | |
|---|---|
| *Op* | prior operators |
| *groupby_num* | number of condtions to groupby |
| *req_col* | names of columns |

## 4.10.3 Member Function Documentation

#### 4.10.3.1 bool GroupBy::aggrerate ( AggrerateMethod *method,* uint64_t *probe_result,* int *agg_i* ) [private]

do aggregation on a column

**Parameters**

| | |
|---|---|
| *method* | Aggregation method |
| *probe_result* | the result of hashtable::probe |
| *agg_i* | the index of aggregation |

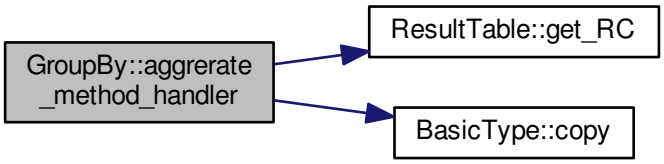
**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

#### 4.10.3.2 void GroupBy::aggrerate_method_handler ( int64_t *hash_count* ) [inline], [private]

aggrerate_method handler

Here is the call graph for this function:




#### 4.10.3.3 bool GroupBy::avg_handler ( ) [inline], [private]

avg handler

#### 4.10.3.4 bool GroupBy::close ( void ) [virtual]

close operator and release memory.

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

#### 4.10.3.5 bool GroupBy::get_Next ( ResultTable * *result* ) [virtual]

get next record of operator

**Parameters**

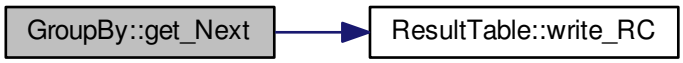
| | |
|---|---|
| *result* | buffer to store result |

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

Here is the call graph for this function:




#### 4.10.3.6 int64_t GroupBy::get_prior_col_rank ( int64_t *object_oid* ) [inline], [private]

get prior column rank

**Return values**

| | |
|---|---|
| *prior* | column rank |

#### 4.10.3.7 Operator* GroupBy::get_prior_operator ( ) [inline], [private]

get pointer to prior Operator

**Return values**

| | |
|---|---|
| *pointer* | to prior Operator |

#### 4.10.3.8 RPattern GroupBy::get_prior_rpattern ( ) [inline], [private]

get prior column rank

**Return values**

| | |
|---|---|
| *prior* | column rank |

#### 4.10.3.9 RowTable* GroupBy::get_prior_tableout ( ) [inline], [private]

get prior tableout

**Return values**

| | |
|---|---|
| *prior* | tableout |

#### 4.10.3.10 RPattern GroupBy::get_rpattern ( ) [inline], [private]

get rpattern

**Return values**

| | |
|---|---|
| *rpattern* | inside |

#### 4.10.3.11 int64_t GroupBy::hash ( char * *str*, int64_t *length* ) [private]

get hash from a string with length

**Parameters**

| | |
|---|---|
| *str* | input a string |
| *length* | the length of string |

**Return values**

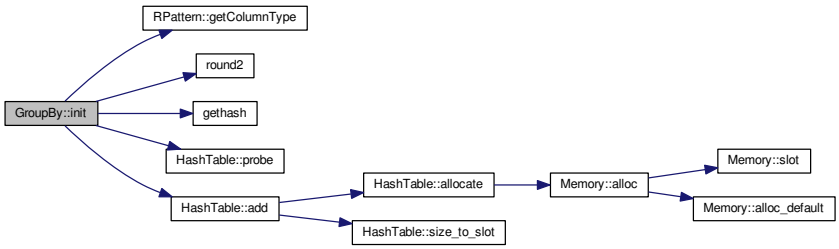
| | |
|---|---|
| *hash* | number |

#### 4.10.3.12 bool GroupBy::init ( void ) [virtual]

init of project

Implements Operator.

Here is the call graph for this function:




#### 4.10.3.13 void GroupBy::init_aggre ( int64_t *col_rank* ) [inline], [private]

init aggrerate

#### 4.10.3.14 void GroupBy::init_non_aggre ( int64_t *col_rank* ) [inline], [private]

init non_aggre

#### 4.10.3.15 bool GroupBy::is_End ( void ) [virtual]

judge whether is end

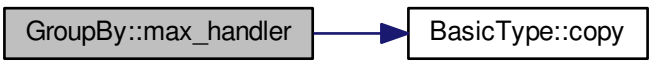
**Return values**

| | |
|---|---|
| *false* | not end |
| *true* | run end |

Implements Operator.

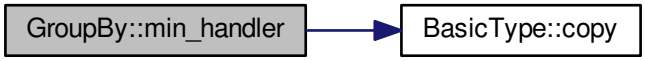
#### 4.10.3.16 void GroupBy::max_handler ( ) [inline], [private]

Here is the call graph for this function:



#### 4.10.3.17 void GroupBy::min_handler ( ) [inline], [private]

Here is the call graph for this function:




#### 4.10.3.18 void GroupBy::sum_handler ( ) [inline], [private]

sum handler

### 4.10.4 Member Data Documentation

#### 4.10.4.1 int GroupBy::agg_i [private]

inside class use

#### 4.10.4.2 int GroupBy::aggrerate_index = 0 [private]

index has been group

#### 4.10.4.3 AggrerateMethod GroupBy::aggrerate_method[4] [private]

aggrerate methods

#### 4.10.4.4 int GroupBy::aggrerate_num = 0 [private]

aggrerate number

#### 4.10.4.5 size_t GroupBy::aggrerate_off[4] [private]

aggrerate offset

#### 4.10.4.6 BasicType* GroupBy::aggrerate_type[4] [private]

type of each aggrerate

#### 4.10.4.7 int GroupBy::col_num = 0 [private]

number of column

#### 4.10.4.8 char* GroupBy::datain [private]

inside class use

#### 4.10.4.9 char* GroupBy::dataout [private]

inside class use

#### 4.10.4.10 int GroupBy::index = 0 [private]

index of current

#### 4.10.4.11 int GroupBy::non_aggrerate_index = 0 [private]

index of non aggrerated

#### 4.10.4.12 int GroupBy::non_aggrerate_num = 0 [private]

number of non aggrerate

#### 4.10.4.13 size_t GroupBy::non_aggrerate_off[4] [private]

non aggrerate method offset

#### 4.10.4.14 BasicType* GroupBy::non_aggrerate_type[4] [private]

type of each non aggrerate

#### 4.10.4.15 int64_t* GroupBy::pattern_count_ptr [private]

inside class use

#### 4.10.4.16 Operator* GroupBy::prior_op [private]

operator of prior oprators

#### 4.10.4.17 RequestColumn* GroupBy::req_col_ptr [private]

inside class use

#### 4.10.4.18 int64_t GroupBy::row_size [private]

size of row

#### 4.10.4.19 RPattern GroupBy::rpattern [private]

inside class use

#### 4.10.4.20 ResultTable GroupBy::tmp_one [private]

buffer to store one result

#### 4.10.4.21 ResultTable GroupBy::tmp_result [private]

buffer to store tmp result

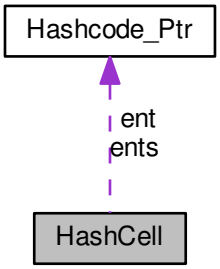
The documentation for this class was generated from the following files:

- system/executor.h
- system/executor.cc

## 4.11 HashCell Class Reference

`#include <hashtable.h>`

Collaboration diagram for HashCell:



### Public Attributes

- int hc_num
- union {
  Hashcode_Ptr ent
  struct {
  int capacity
  Hashcode_Ptr * ents
  } num_2_or_more
  } hc_union

## 4.11.1 Detailed Description

definition of class HashCell.

## 4.11.2 Member Data Documentation

#### 4.11.2.1 int HashCell::capacity

maximun number of array in this structure

#### 4.11.2.2 Hashcode_Ptr HashCell::ent

Hashcode_Ptr, hit,decrease cache miss

#### 4.11.2.3 Hashcode_Ptr* HashCell::ents

pointer of Hashcode_Ptr array

#### 4.11.2.4 int HashCell::hc_num

Hashcode_Ptr number in this HashCell

#### 4.11.2.5 union { ... } HashCell::hc_union

if hc_num == 1,store one Hashcode_Ptr; if hc_num>1,store num_2_or_more

#### 4.11.2.6 struct { ... } HashCell::num_2_or_more

struct of a Hashcode_Ptr array,not hit,seach a array,not a link-list,decrease cache miss

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

- system/hashtable.h

## 4.12 Hashcode_Ptr Class Reference

`#include <hashtable.h>`

### Public Attributes

- int64_t hash_code
- char * tuple

### 4.12.1 Detailed Description

File Name: baseline/hashTable.h Written By: Shimin Chen, Sept, 2002 Description: in-memory hash table implementation.

The hash table stores hash codes and pointers to the tuples. It has an array of hash cells, which contains a hash code and a pointer. In case of confliction, a variable sized array of hash code and pointer is allocated.

Modified By liugang (liugang@ict.ac.cn)definition of class Hashcode_Ptr.

### 4.12.2 Member Data Documentation

#### 4.12.2.1 int64_t Hashcode_Ptr::hash_code

hash_code of specific data

#### 4.12.2.2 char* Hashcode_Ptr::tuple

pointer of a record tuple

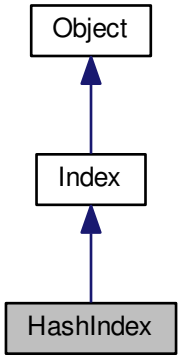
The documentation for this class was generated from the following file:

- system/hashtable.h

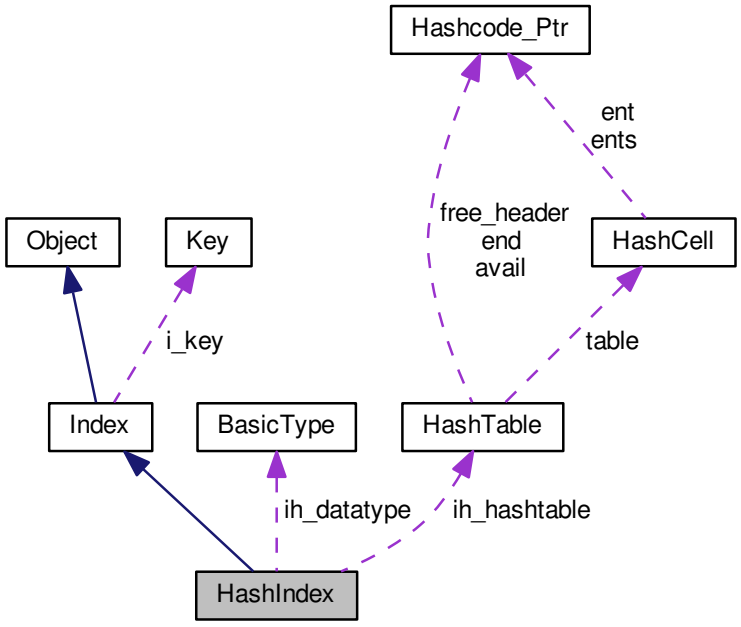
## 4.13 HashIndex Class Reference

`#include <hashindex.h>`

Inheritance diagram for HashIndex:




Collaboration diagram for HashIndex:



### Public Member Functions

- HashIndex (int64_t h_id, const char *i_name, Key &i_key)
- bool addIndexDTpye (BasicType *i_dt)
- bool del (void *i_data)
- bool del (void *i_data[])
- bool finish (void)
- bool init (void)
- bool insert (void *i_data, void *p_in)
- bool insert (void *i_data[], void *p_in)
- bool lookup (void *i_data, void *info, void *&result)
- bool lookup (void *i_data[], void *info, void *&result)
- bool set_ls (void *i_data1, void *i_data2, void *info)
- bool set_ls (void *i_data1[], void *i_data2[], void *info)
- void setCellCap (int64_t cell_capbits)
- bool shut (void)

### Private Member Functions

- bool cmpEQ (void *i_data[], void *result)
- bool cmpEQ (void *i_data, void *result)
- int64_t hash (char *str, int64_t maxlen)
- void print (void)
- int64_t tranToInt64 (void *i_data)
- int64_t tranToInt64 (void *i_data[])

### Private Attributes

- int64_t ih_cell_capbits
- int64_t ih_column_cap
- int64_t ih_column_num
- BasicType ** ih_datatype
- int64_t * ih_hash_bits
- HashTable * ih_hashtable

## 4.13.1 Detailed Description

definition of HashIndex.

## 4.13.2 Constructor & Destructor Documentation

### 4.13.2.1 HashIndex::HashIndex ( int64_t *h_id*, const char * *i_name*, Key & *i_key* ) [inline]

constructor.

**Parameters**

| | |
|---|---|
| *h_id* | hash index identifier |
| *i_name* | index name |
| *i_key* | key of this index |

### 4.13.3 Member Function Documentation

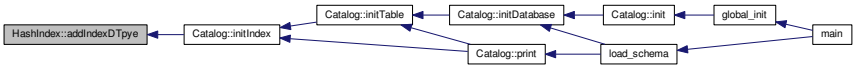
#### 4.13.3.1 bool HashIndex::addIndexDTpye ( BasicType * *i_dt* )

add indexed column's data type.

**Parameters**

| | |
|---|---|
| *i_dt* | data type of indexed column |

Here is the caller graph for this function:




#### 4.13.3.2 bool HashIndex::cmpEQ ( void * *i_data[ ]*, void * *result* ) [private]

whether i_data is result got from hash table

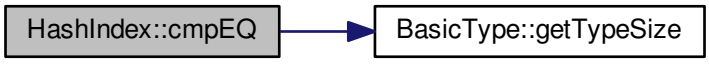
**Parameters**

| | |
|---|---|
| *i_data* | pointers of columns |
| *result* | got from hash table |

**Return values**

| | |
|---|---|
| *true* | equal |
| *false* | not equal |

Here is the call graph for this function:



Here is the caller graph for this function:


#### 4.13.3.3 bool HashIndex::cmpEQ ( void * *i_data*, void * *result* ) [private]

whether i_data is result got from hash table

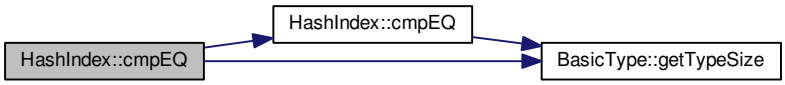
**Parameters**

| | |
|---|---|
| *i_data* | buffer data of columns |
| *result* | got from hash table |

**Return values**

| | |
|---|---|
| *true* | equal |
| *false* | not equal |

Here is the call graph for this function:




#### 4.13.3.4 bool HashIndex::del ( void * *i_data* ) [virtual]

del an entry in hash index.

**Parameters**

| | |
|---|---|
| *i_data* | buffer of column data |

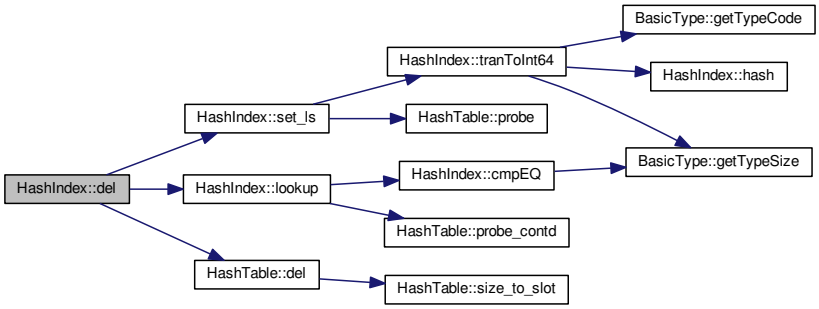
**Return values**

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Index.

Here is the call graph for this function:




#### 4.13.3.5 bool HashIndex::del ( void * *i_data[ ]* ) [virtual]

del an entry in hash index.

**Parameters**

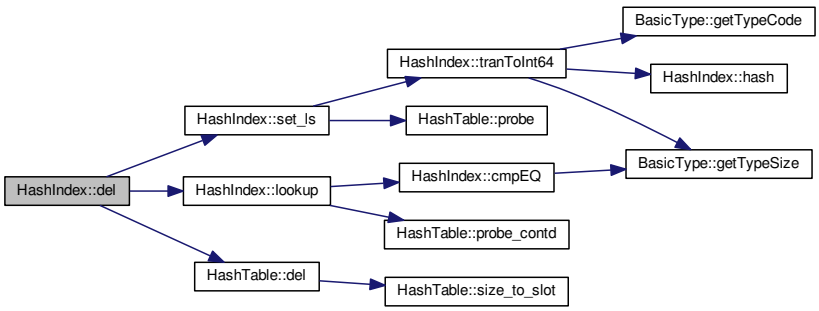
| | |
|---|---|
| *i_data* | pointers of column data |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Index.

Here is the call graph for this function:




#### 4.13.3.6 bool HashIndex::finish ( void ) [virtual]

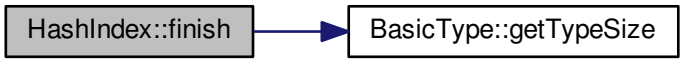
init of hash table, heart of hash index ,most important.

**Return values**

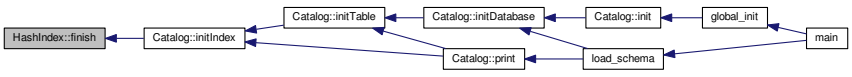
| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Index.

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.13.3.7 int64_t HashIndex::hash ( char * *str,* int64_t *maxlen* ) [private]

hash method for CHARN.

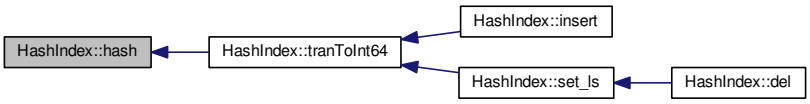
**Parameters**

| | |
|---|---|
| *str* | pointer of string |
| *maxlen* | maximum length of str |

**Return values**

| | |
|---|---|
| *int64* | hash value of CHARN |

Here is the caller graph for this function:




#### 4.13.3.8 bool HashIndex::init ( void ) [virtual]

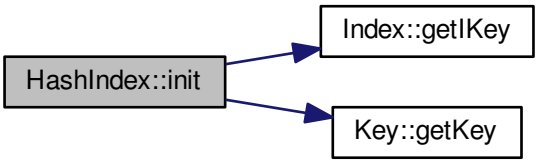
init hashindex, to calculate initial value.

**Return values**

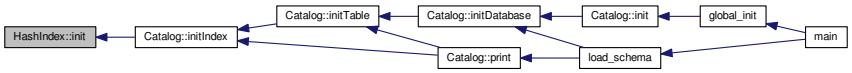
| | |
|---|---|
| *true* | init success |

Reimplemented from Index.

Here is the call graph for this function:



Here is the caller graph for this function:




#### 4.13.3.9 bool HashIndex::insert ( void * *i_data*, void * *p_in* ) [virtual]

insert an entry to hash index.

**Parameters**

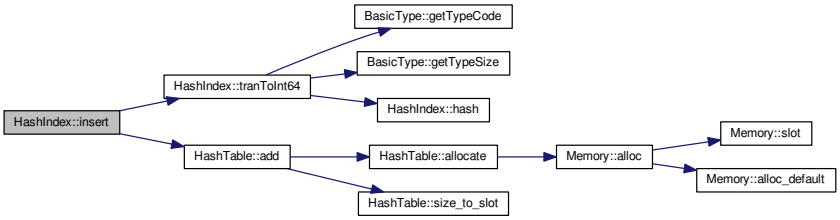
| | |
|---|---|
| *i_data* | buffer of column data in pattren |
| *p_in* | pointer of record to make index |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Index.

Here is the call graph for this function:




#### 4.13.3.10 bool HashIndex::insert ( void * *i_data[ ]*, void * *p_in* ) [virtual]

insert an entry to hash index.

**Parameters**

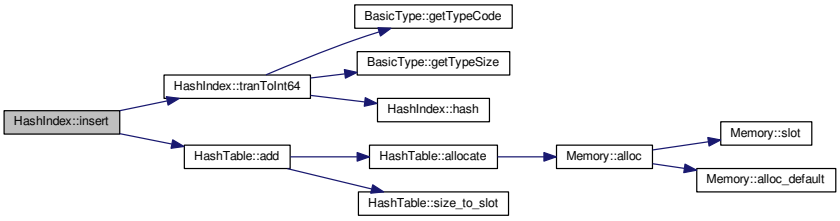
| | |
|---|---|
| *i_data* | each element of i_data pointed to a column data |
| *p_in* | pointer of record to make index |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Index.

Here is the call graph for this function:




#### 4.13.3.11 bool HashIndex::lookup ( void * *i_data*, void * *info*, void *& *result* ) [virtual]

lookup hash index.

**Parameters**

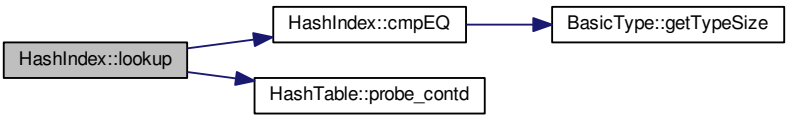
| | |
|---|---|
| *i_data* | buffer of column data |
| *info* | HashInfo pointer processed by set_ls |
| *result* | reference of record pointer |

**Return values**

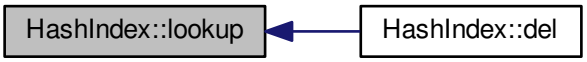
| | |
|---|---|
| *true* | found |
| *false* | not found |

Reimplemented from Index.

Here is the call graph for this function:



Here is the caller graph for this function:




#### 4.13.3.12 bool HashIndex::lookup ( void * *i_data[]*, void * *info*, void *& *result* ) [virtual]

lookup hash index.

**Parameters**

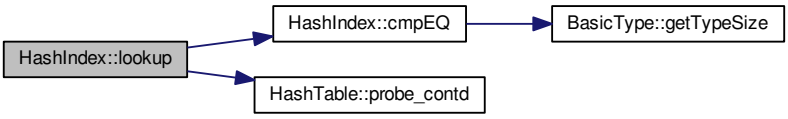
| | |
|---|---|
| *i_data* | pointers of column data |
| *info* | HashInfo pointer processed by set_ls |
| *result* | reference of record pointer |

**Return values**

| | |
|---|---|
| *true* | found |
| *false* | not found |

Reimplemented from Index.

Here is the call graph for this function:

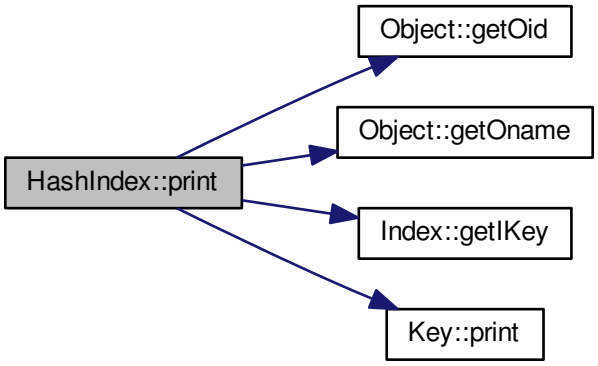



#### 4.13.3.13 void HashIndex::print ( void ) [private], [virtual]

print hash index information.

Reimplemented from Index.

Here is the call graph for this function:




#### 4.13.3.14 bool HashIndex::set_ls ( void * *i_data1*, void * *i_data2*, void * *info* ) [virtual]

setup for hash index lookup.

**Parameters**

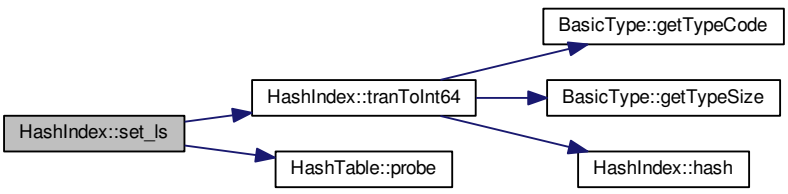
| | |
|---|---|
| *i_data1* | buffer of column data for lookup or scan(">=") |
| *i_data2* | set NULL |
| *info* | HashInfo pointer |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Index.

Here is the call graph for this function:



Here is the caller graph for this function:


#### 4.13.3.15 bool HashIndex::set_ls ( void * *i_data1[]*, void * *i_data2[]*, void * *info* ) [virtual]

setup for hash index lookup.

**Parameters**

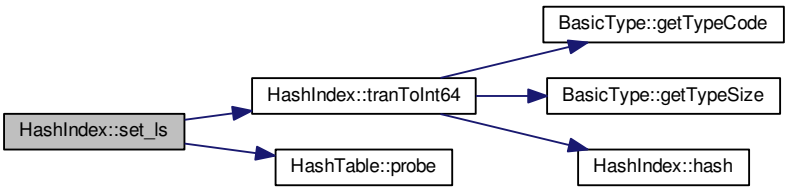
| | |
|---|---|
| *i_data1* | pointers of column data for lookup |
| *i_data2* | set NULL when call |
| *info* | HashInfo pointer |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Index.

Here is the call graph for this function:



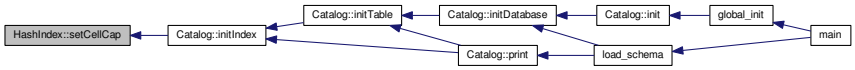

#### 4.13.3.16 void HashIndex::setCellCap ( int64_t *cell_capbits* ) [inline]

set hashtable cell capicity.

**Parameters**

| | |
|---|---|
| *cell_capbits* | the number of cells in hashtable is 2^cell_capbits |

Here is the caller graph for this function:




#### 4.13.3.17 bool HashIndex::shut ( void ) [virtual]

free memory of hash table and other strucure.

**Return values**

| | |
|---|---|
| *true* | success |

Reimplemented from Index.

#### 4.13.3.18 int64_t HashIndex::tranToInt64 ( void * *i_data* ) [private], [virtual]

assemble hash keys, INT and CHARN.

**Parameters**

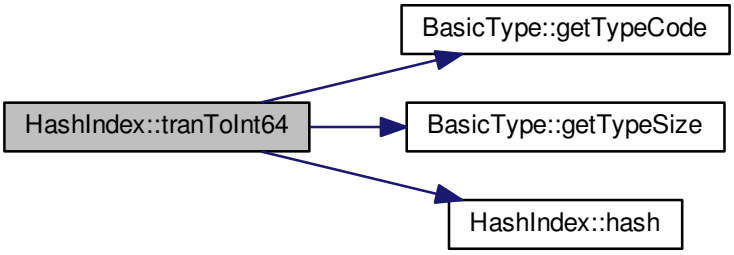
| | |
|---|---|
| *i_data* | buffer of column data |

**Return values**

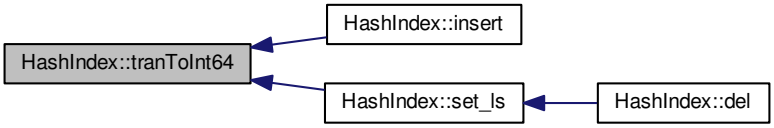
| | |
|---|---|
| *int64* | hash index code |

Reimplemented from Index.

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.13.3.19 int64_t HashIndex::tranToInt64 ( void * *i_data[ ]* ) [private], [virtual]

assemble hash keys, INT and CHARN.

**Parameters**

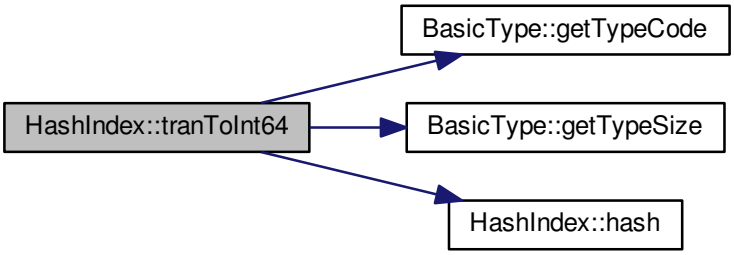
| | |
|---|---|
| *i_data* | pointers of column data |

**Return values**

| | |
|---|---|
| *int64* | hash index code |

Reimplemented from Index.

Here is the call graph for this function:




### 4.13.4 Member Data Documentation

#### 4.13.4.1 int64_t HashIndex::ih_cell_capbits [private]

as cellnum is power of 2, so the number of bits can use is log2(cellnum)

#### 4.13.4.2 int64_t HashIndex::ih_column_cap [private]

got from parent class, the number of columns in the key

#### 4.13.4.3 int64_t HashIndex::ih_column_num [private]

current number of added columns

#### 4.13.4.4 BasicType** HashIndex::ih_datatype [private]

each column data type

#### 4.13.4.5 int64_t* HashIndex::ih_hash_bits [private]

each column assigend bits when hashing

#### 4.13.4.6 HashTable* HashIndex::ih_hashtable [private]

main part hash table

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

- system/hashindex.h
- system/hashindex.cc

## 4.14 HashInfo Struct Reference

`#include <hashindex.h>`

### Public Attributes

- int64_t hash
- int last
- int ppos
- char * result [HASHINFO_CAPICITY]
- int rnum

### 4.14.1 Detailed Description

definition of HashInfo.

### 4.14.2 Member Data Documentation

#### 4.14.2.1 int64_t HashInfo::hash

hashcell position in HashTable

#### 4.14.2.2 int HashInfo::last

retval of HashTable lookup

#### 4.14.2.3 int HashInfo::ppos

pair with last, |last|

#### 4.14.2.4 char* HashInfo::result[HASHINFO_CAPICITY]

buffer for value of void *pointer

#### 4.14.2.5 int HashInfo::rnum

current result number

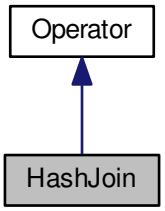
The documentation for this struct was generated from the following file:

- system/hashindex.h

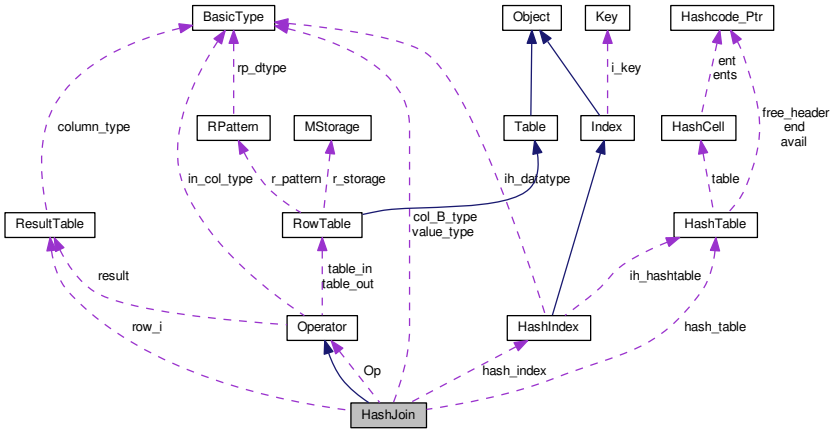
## 4.15 HashJoin Class Reference

`#include <executor.h>`

Inheritance diagram for HashJoin:




Collaboration diagram for HashJoin:




### Public Member Functions

- HashJoin (int operator_num, Operator **Op, int cond_num, Condition *condi)
- void alloValue (char *value, char *format_value)
- bool close ()
- void Fdeal (int operator_num, Operator **Op)
- bool get_Next (ResultTable *result)
- bool init ()
- bool is_End ()
- void MakeOrder ()
- bool ShutMem ()
- bool WriteRow (bool flag, ResultTable *result)

### Private Attributes

- int64_t col_A_oid = -1
- int64_t col_A_rank = -1
- int64_t col_B_oid = -1
- int64_t col_B_rank = -1
- int col_B_row = 0
- BasicType ** col_B_type
- int col_num [2] = {0, 0}
- int cond_num = 0
- HashIndex * hash_index = NULL
- HashTable * hash_table = NULL
- Operator * Op [2] = {NULL, NULL}
- int operator_num = 0
- ResultTable row_i [200000]
- BasicType * value_type

### Additional Inherited Members

## 4.15.1 Detailed Description

definition of hashjoin.

## 4.15.2 Constructor & Destructor Documentation

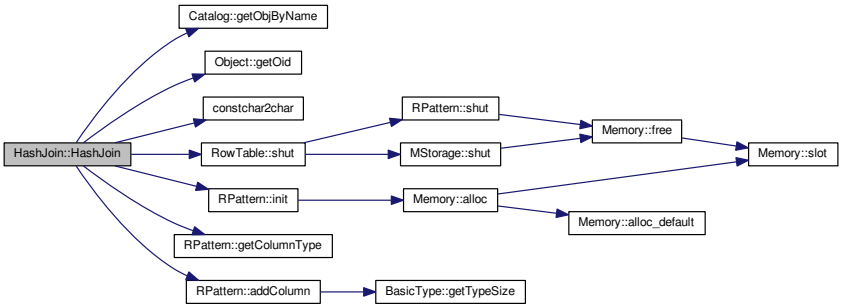
#### 4.15.2.1 HashJoin::HashJoin ( int *operator_num,* Operator ** *Op,* int *cond_num,* Condition * *condi* )

construction of HashJoin

**Parameters**

| | |
|---|---|
| *operator_num* | tables need to Join |
| *Op* | prior oprators |
| *cond_num* | condition number of join |
| *join* | condtions |

Here is the call graph for this function:




### 4.15.3 Member Function Documentation

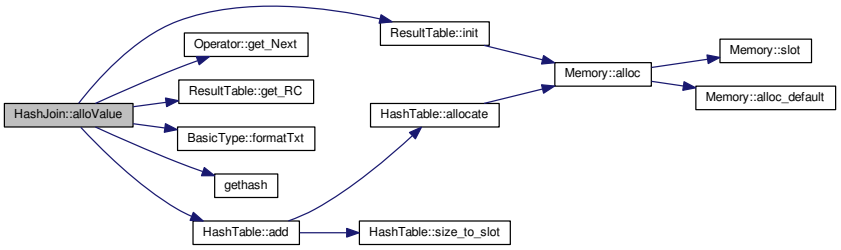
#### 4.15.3.1 void HashJoin::alloValue ( char * *value*, char * *format_value* ) [inline]

allocate value

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:




#### 4.15.3.2 bool HashJoin::close ( void ) [virtual]

close operator and release memory.

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

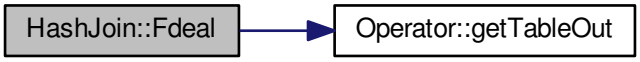
#### 4.15.3.3 void HashJoin::Fdeal ( int *operator_num,* Operator ** *Op* ) [inline]

first deal with the data

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:




#### 4.15.3.4 bool HashJoin::get_Next ( ResultTable * *result* ) [virtual]

get next record of hash Join

**Parameters**

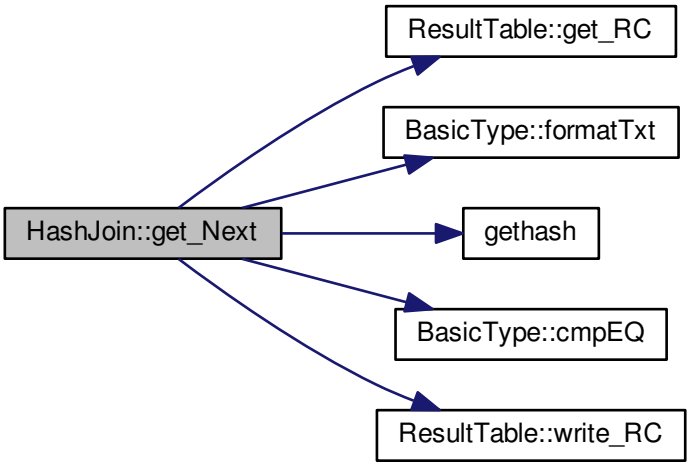
| | |
|---|---|
| *result* | buffer to store result |

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

Here is the call graph for this function:




#### 4.15.3.5 bool HashJoin::init ( void ) [virtual]

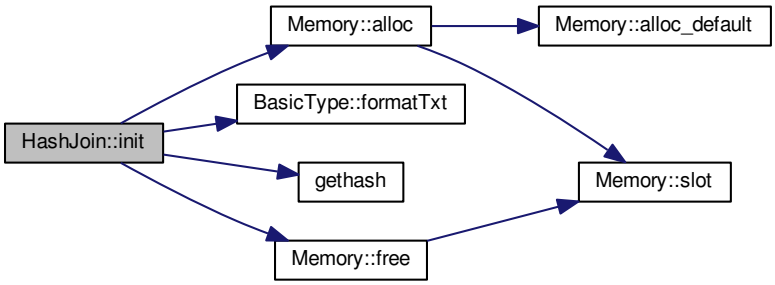
init hashjoin.

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

Here is the call graph for this function:



#### 4.15.3.6 bool HashJoin::is_End ( void ) [virtual]

judge whether is end

**Return values**

| | |
|---|---|
| *false* | not end |
| *true* | run end |

Implements Operator.

#### 4.15.3.7 void HashJoin::MakeOrder ( ) [inline]

make sure colA belongs to table_0

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

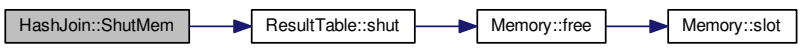
#### 4.15.3.8 bool HashJoin::ShutMem ( ) [inline]

shut memory

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:



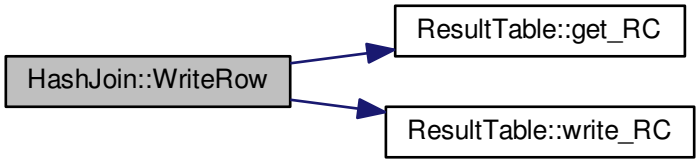

#### 4.15.3.9 bool HashJoin::WriteRow ( bool *flag*, ResultTable * *result* ) [inline]

write row

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:




### 4.15.4 Member Data Documentation

#### 4.15.4.1 int64_t HashJoin::col_A_oid = -1 [private]

column A object id

#### 4.15.4.2 int64_t HashJoin::col_A_rank = -1 [private]

column A rank

#### 4.15.4.3 int64_t HashJoin::col_B_oid = -1 [private]

column B object id

#### 4.15.4.4 int64_t HashJoin::col_B_rank = -1 [private]

column B rank

#### 4.15.4.5 int HashJoin::col_B_row = 0 [private]

the rownumber of table B

#### 4.15.4.6 BasicType** HashJoin::col_B_type [private]

column types of table B

#### 4.15.4.7 int HashJoin::col_num[2] = {0, 0} [private]

the clolumn numbers of two table

#### 4.15.4.8 int HashJoin::cond_num = 0 [private]

number of join conditions

#### 4.15.4.9 HashIndex* HashJoin::hash_index = NULL [private]

hash index of hash table

#### 4.15.4.10 HashTable* HashJoin::hash_table = NULL [private]

hash tale to store data

#### 4.15.4.11 Operator* HashJoin::Op[2] = {NULL, NULL} [private]

prior operator from tow join table.

#### 4.15.4.12 int HashJoin::operator_num = 0 [private]

number of join table

#### 4.15.4.13 ResultTable HashJoin::row_i[200000] [private]

result table to store temp result

#### 4.15.4.14 BasicType* HashJoin::value_type [private]

result type of result

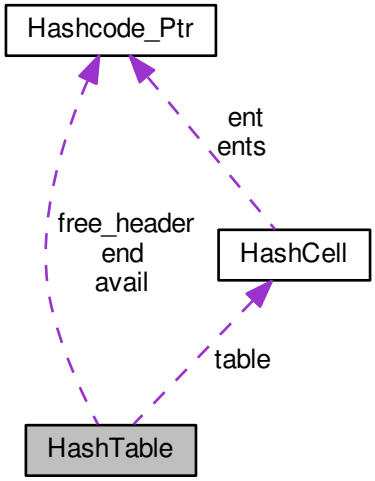
The documentation for this class was generated from the following files:

- system/executor.h
- system/executor.cc

## 4.16 HashTable Class Reference

`#include <hashtable.h>`

Collaboration diagram for HashTable:




### Public Member Functions

- HashTable (int estimatedNumDistinctKeys, double estimatedDupPerKey, int num_partitions)
- ~HashTable ()
- bool add (int64_t hashCode, char *tup)
- bool del (int64_t hashCode, char *tup)
- int probe (int64_t hashCode, char *match[], int capacity)
- int probe_contd (int64_t hashCode, int last, char *match[], int capacity)
- void show ()
- void utilization ()

### Public Attributes

- Hashcode_Ptr * avail
- char * begin
- Hashcode_Ptr * end
- double estimated_duplicates_per_key
- int estimated_num_distinct_keys
- Hashcode_Ptr * free_header [16]
- int initial_array_size
- int more_allocated
- HashCell * table
- int table_size

### Private Member Functions

- void * allocate (int size)
- void free (void *mem)
- int size_to_slot (int array_size)

### Private Attributes

- std::unordered_map< void *, int > pointer2size

## 4.16.1 Detailed Description

definition of class HashTable.

## 4.16.2 Constructor & Destructor Documentation

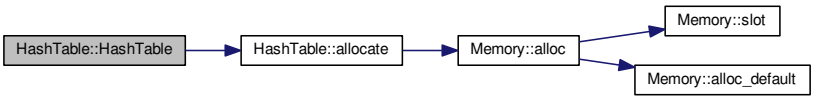
### 4.16.2.1 HashTable::HashTable ( int *estimatedNumDistinctKeys,* double *estimatedDupPerKey,* int *num_partitions* = 0 )

constructor.

**Parameters**

| | |
|---|---|
| *estimatedNumDistinctKeys* | estimated number of distinct keys,pre-knowledge for this HashTable usage |
| *estimatedDupPerKey* | estimated number of dupicate keys in average,pre-knowledge for this HashTable usage |
| *num_partitions* | leave it 0, unuseable |

Here is the call graph for this function:




### 4.16.2.2 HashTable::~HashTable ( )

destructor, free HashTable memory to g_memory.

Here is the call graph for this function:


### 4.16.3 Member Function Documentation

#### 4.16.3.1 bool HashTable::add ( int64_t *hashCode*, char * *tup* )

add an entry.

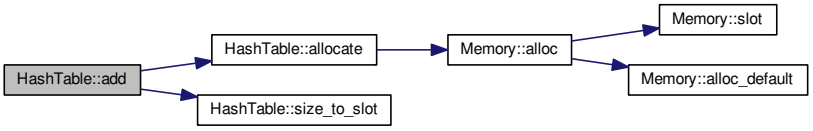
**Parameters**

| | |
|---|---|
| *hashCode* | hash code of specified data |
| *tup* | pointer of a record tuple |

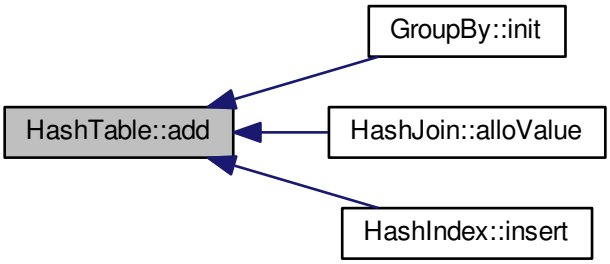
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:



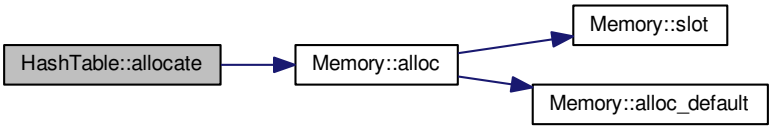
Here is the caller graph for this function:



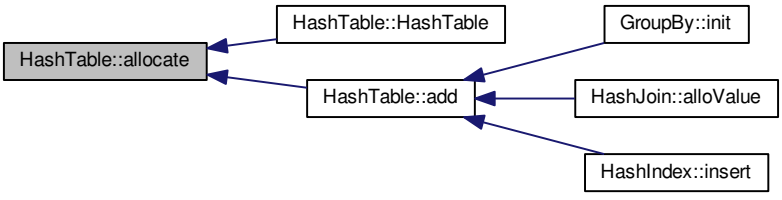

#### 4.16.3.2 void * HashTable::allocate ( int *size* ) [private]

mymemory alloc interface like malloc

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.16.3.3 bool HashTable::del ( int64_t *hashCode*, char * *tup* )

del an entry.

**Parameters**

| | |
|---|---|
| *hashCode* | hash code of specified data |
| *tup* | pointer of a record tuple |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

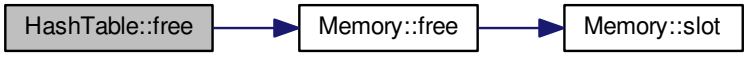
Here is the call graph for this function:


Here is the caller graph for this function:


#### 4.16.3.4 void HashTable::free ( void * *mem* ) [private]

mymemory free interface like free in stdlib

Here is the call graph for this function:




#### 4.16.3.5 int HashTable::probe ( int64_t *hashCode*, char * *match[]*, int *capacity* )

probe(lookup) entries with specified hashCode.

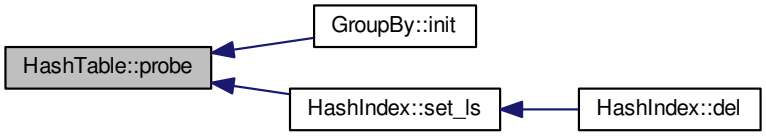
**Parameters**

| | |
|---|---|
| *hashCode* | the specified hashCode to find |
| *match* | the buffer to store the result matched |
| *capacity* | the maximum number of tuple pointers in this buffer |

**Return values**

| | |
|---|---|
| *<0* | means capacity has been reached, there could be more |
| *>=0* | means this probe has finished all searching work,retval is the number of result |

Here is the caller graph for this function:




#### 4.16.3.6 int HashTable::probe_contd ( int64_t *hashCode*, int *last*, char * *match*[ ], int *capacity* )

probe_contd(lookup) more entries with specified hashCode.

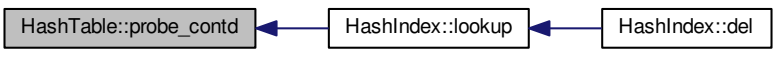
**Parameters**

| | |
|---|---|
| *hashCode* | the specified hashCode to find |
| *last* | inverse number of the retval returned by last call of probe or probe_contd function |
| *match* | the buffer to store the result matched |
| *capacity* | the maximum number of tuple pointers in this buffer |

**Return values**

| | |
|---|---|
| *<0* | means capacity has been reached, there could be more |
| *>=0* | means this probe has finished all searching work,retval is the number of result |

Here is the caller graph for this function:



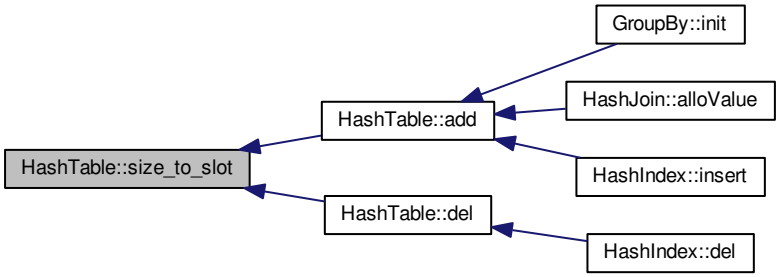
#### 4.16.3.7 void HashTable::show ( )

display data in this hash table, for debug use.

#### 4.16.3.8 int HashTable::size_to_slot ( int *array_size* ) [private]

find the offset in free_header arrray to get suitable free memory used in this HashTable

Here is the caller graph for this function:




#### 4.16.3.9 void HashTable::utilization ( )

display usage analysis of this hash table, for debug use.

### 4.16.4 Member Data Documentation

#### 4.16.4.1 Hashcode_Ptr* HashTable::avail

pointer of next available Hashcode_Ptr

#### 4.16.4.2 char* HashTable::begin

start pointer of HashCells

#### 4.16.4.3 Hashcode_Ptr* HashTable::end

the end pointer of Hashcode_Ptr in array

#### 4.16.4.4 double HashTable::estimated_duplicates_per_key

estimated number of dupicate keys in average,pre-knowledge for this HashTable usage

#### 4.16.4.5 int HashTable::estimated_num_distinct_keys

estimated number of distinct keys,pre-knowledge for this HashTable usage

#### 4.16.4.6 Hashcode_Ptr* HashTable::free_header[16]

free memory in the list with different number of Hashcode_Ptr,link-list

#### 4.16.4.7 int HashTable::initial_array_size

when hc_num of HashCell exceeds 1 at the fist time, number of Hashcode_Ptr allcated for HashCell

#### 4.16.4.8 int HashTable::more_allocated

analysis of more memory allocated from g_memory

#### 4.16.4.9 std::unordered_map<void*, int> HashTable::pointer2size [private]

unordered map, memory pointer to its size,an adapter for mymemory component

#### 4.16.4.10 HashCell* HashTable::table

pointer of an array of HashCell

#### 4.16.4.11 int HashTable::table_size

the number of HashCells in this table

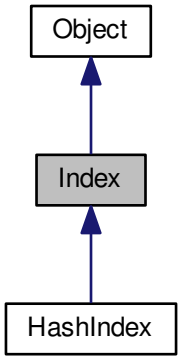
The documentation for this class was generated from the following files:

- system/hashtable.h
- system/hashtable.cc

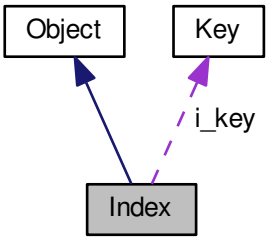
## 4.17 Index Class Reference

`#include <schema.h>`

Inheritance diagram for Index:




Collaboration diagram for Index:




### Public Member Functions

- Index (int64_t i_id, const char *i_name, IndexType i_type, Key &i_key)
- virtual ~Index (void)
- virtual bool del (void *i_data)
- virtual bool del (void *i_data[ ])
- virtual bool finish (void)
- Key & getIKey (void)
- IndexType getIType (void)
- virtual bool init (void)
- virtual bool insert (void *i_data, void *p_in)

- virtual bool insert (void *i_data[], void *p_in)
- virtual bool lookup (void *i_data, void *&result)
- virtual bool lookup (void *i_data[], void *&result)
- virtual bool lookup (void *i_data, void *info, void *&result)
- virtual bool lookup (void *i_data[], void *info, void *&result)
- virtual void print (void)
- virtual bool scan_1 (void *i_left, void *info)
- virtual bool scan_1 (void *i_left[], void *info)
- virtual bool scan_2 (void *i_right, void *info, void *&result)
- virtual bool scan_2 (void *i_right[], void *info, void *&result)
- virtual bool set_ls (void *i_data1, void *i_data2, void *info)
- virtual bool set_ls (void *i_data1[], void *i_data2[], void *info)
- virtual bool shut (void)
- virtual int64_t tranToInt64 (void *i_data)
- virtual int64_t tranToInt64 (void *i_data[])
- virtual bool update (void *i_data, void *p_in)
- virtual bool update (void *i_data[], void *p_in)

### Private Attributes

- Key i_key
- IndexType i_type

## 4.17.1 Detailed Description

definition of class Index

## 4.17.2 Constructor & Destructor Documentation

#### 4.17.2.1 Index::Index ( int64_t *i_id*, const char * *i_name*, IndexType *i_type*, Key & *i_key* ) [inline]

constructor.

**Parameters**

| | |
|---|---|
| *i_id* | index identifier |
| *i_name* | index name |
| *i_type* | index type |
| *i_key* | key of this index |

#### 4.17.2.2 virtual Index::~Index ( void ) [inline], [virtual]

destructor.

## 4.17.3 Member Function Documentation

#### 4.17.3.1 virtual bool Index::del ( void * *i_data* ) [inline], [virtual]

del an entry in Index.

**Parameters**

| | |
|---|---|
| *i_data* | char buffer to store data in pattern |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

Reimplemented in HashIndex.

#### 4.17.3.2 virtual bool Index::del ( void * *i_data[ ]* ) [inline], [virtual]

del an entry in BptreeIndex.

**Parameters**

| | |
|---|---|
| *i_data* | each element of the array stores a pointer to column key |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

Reimplemented in HashIndex.

#### 4.17.3.3 virtual bool Index::finish ( void ) [inline], [virtual]

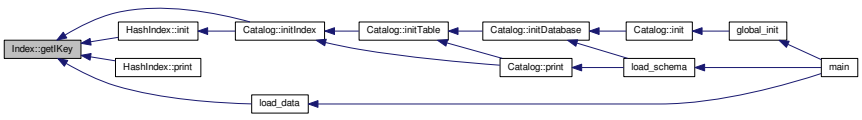
finish index, important interface for son class

Reimplemented in HashIndex.

#### 4.17.3.4 Key& Index::getIKey ( void ) [inline]

get index key

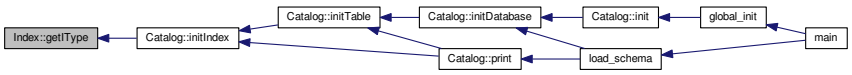
Here is the caller graph for this function:



#### 4.17.3.5 IndexType Index::getIType ( void ) [inline]

get index type

Here is the caller graph for this function:




#### 4.17.3.6 virtual bool Index::init ( void ) [inline], [virtual]

init index, important interface for son class

Reimplemented in HashIndex.

#### 4.17.3.7 virtual bool Index::insert ( void * *i_data*, void * *p_in* ) [inline], [virtual]

insert an entry to Index.

**Parameters**

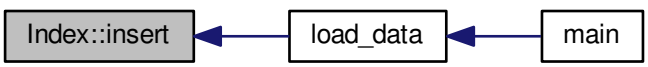
| | |
|---|---|
| *i_data* | char buffer to store data in pattern |
| *p_in* | pointer of a row to make index |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

Reimplemented in HashIndex.

Here is the caller graph for this function:



#### 4.17.3.8 virtual bool Index::insert ( void * *i_data[ ]*, void * *p_in* ) [inline], [virtual]

insert an entry to Index.

**Parameters**

| | |
|---|---|
| *i_data* | each element of the array stores a pointer to column key |
| *p_in* | pointer of a row to make index |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

Reimplemented in HashIndex.

#### 4.17.3.9 virtual bool Index::lookup ( void * *i_data,* void *& *result* ) [inline], [virtual]

lookup nonduplicate key in Index.

**Parameters**

| | |
|---|---|
| *i_data* | char buffer to store data in pattern |
| *result* | return the pointer to the indexed row |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

#### 4.17.3.10 virtual bool Index::lookup ( void * *i_data[],* void *& *result* ) [inline], [virtual]

lookup nonduplicate key in Index.

**Parameters**

| | |
|---|---|
| *i_data* | each element of the array stores a pointer to column key |
| *result* | return the pointer to the indexed row |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

#### 4.17.3.11 virtual bool Index::lookup ( void * *i_data,* void * *info,* void *& *result* ) [inline], [virtual]

lookup duplicate key, iterate through the Index.

**Parameters**

| | |
|---|---|
| *i_data* | char buffer to store data in pattern |
| *info* | pointer of a BptreeInfo |
| *result* | return the pointer to the indexed row |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

Reimplemented in HashIndex.

#### 4.17.3.12 virtual bool Index::lookup ( void * *i_data[]*, void * *info*, void *& *result* ) [inline], [virtual]

lookup duplicate key,iterate through the Index.

**Parameters**

| | |
|---|---|
| *i_data* | each element of the array stores a pointer to column key |
| *info* | pointer of an index info |
| *result* | return the pointer to the indexed row |

**Return values**

| | |
|---|---|
| *true* | has more values |
| *false* | no more values |

Reimplemented in HashIndex.

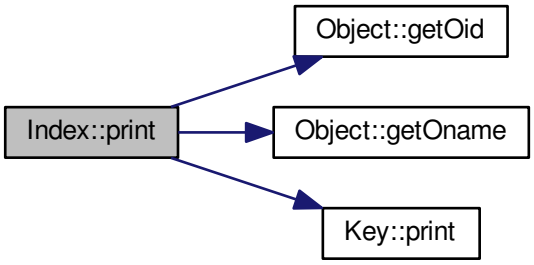
#### 4.17.3.13 virtual void Index::print ( void ) [inline], [virtual]

print index information

Reimplemented from Object.

Reimplemented in HashIndex.

Here is the call graph for this function:




#### 4.17.3.14 virtual bool Index::scan_1 ( void * *i_left*, void * *info* ) [inline], [virtual]

prepare for scan operation.

**Parameters**

| | |
|---|---|
| *i_left* | char buffer to store data in pattern, ">=" |
| *info* | pointer of an index info |

**Return values**

| | |
|---|---|
| *true* | has more values |
| *false* | no more values |

#### 4.17.3.15 virtual bool Index::scan_1 ( void * *i_left[ ]*, void * *info* ) [inline], [virtual]

pepare for scan operation.

**Parameters**

| | |
|---|---|
| *i_left* | each element of the array stores a pointer to column key, ">=" |
| *info* | pointer of an index info |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

#### 4.17.3.16 virtual bool Index::scan_2 ( void * *i_right,* void * *info,* void *& *result* ) [inline], [virtual]

iterate on calling to scan for values.

**Parameters**

| | |
|---|---|
| *i_right* | char buffer to store data in pattern, "<" |
| *info* | pointer of an index info |
| *result* | return the pointer to the indexed row |

**Return values**

| | |
|---|---|
| *true* | has more values |
| *false* | no more values |

#### 4.17.3.17 virtual bool Index::scan_2 ( void * *i_right[ ],* void * *info,* void *& *result* ) [inline], [virtual]

iterate on calling to scan for values.

**Parameters**

| | |
|---|---|
| *i_right* | each element of the array stores a pointer to column key, "<" |
| *info* | pointer of an index info |
| *result* | return the pointer to the indexed row |

**Return values**

| | |
|---|---|
| *true* | has more values |
| *false* | no more values |

#### 4.17.3.18 virtual bool Index::set_ls ( void * *i_data1,* void * *i_data2,* void * *info* ) [inline], [virtual]

prepare for lookup/scan operation.

**Parameters**

| | |
|---|---|
| *i_data1* | char buffer to store data in pattern |
| *i_data2* | char buffer to store data in pattern, for lookup, i_data2=NULL |
| *info* | pointer of an index info |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

Reimplemented in HashIndex.

#### 4.17.3.19 virtual bool Index::set_ls ( void * *i_data1[ ],* void * *i_data2[ ],* void * *info* ) [inline], [virtual]

prepare for lookup/scan operation.

**Parameters**

| | |
|---|---|
| *i_data1* | each element of the array stores a pointer to column key |
| *i_data2* | each element of the array stores a pointer to column key, for lookup, i_data2=NULL |
| *info* | pointer of an index info |

**Return values**

| | |
|---|---|
| *true* | operation success |
| *false* | operation failure |

Reimplemented in HashIndex.

#### 4.17.3.20 virtual bool Index::shut ( void ) [inline], [virtual]

shut dowm the index, free memory.

Reimplemented from Object.

Reimplemented in HashIndex.

#### 4.17.3.21 virtual int64_t Index::tranToInt64 ( void * *i_data* ) [inline], [virtual]

encode keys.

**Parameters**

| | |
|---|---|
| *i_data* | char buffer to store data in pattern |

**Return values**

| | |
|---|---|
| *int64_t* | index code of i_data |

Reimplemented in HashIndex.

#### 4.17.3.22 virtual int64_t Index::tranToInt64 ( void * *i_data[ ]* ) [inline], [virtual]

encode keys.

**Parameters**

| | |
|---|---|
| *i_data* | each element of the array stores a pointer to column key |

**Return values**

| | |
|---|---|
| *int64_t* | index code of i_data |

Reimplemented in HashIndex.

#### 4.17.3.23 virtual bool Index::update ( void * *i_data*, void * *p_in* ) [inline], [virtual]

#### 4.17.3.24 virtual bool Index::update ( void * *i_data[]*, void * *p_in* ) [inline], [virtual]

### 4.17.4 Member Data Documentation

#### 4.17.4.1 Key Index::i_key [private]

index keys

#### 4.17.4.2 IndexType Index::i_type [private]

index type

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

- system/schema.h

## 4.18 Key Class Reference

`#include <schema.h>`

### Public Member Functions

- Key (void)
- bool contain (int64_t col_id)
- std::vector< int64_t > & getKey (void)
- Key & operator= (const Key &p)
- void print (void)
- void set (std::vector< int64_t > &in_key)

### Private Attributes

- std::vector< int64_t > key

### 4.18.1 Detailed Description

definition of class Key.

### 4.18.2 Constructor & Destructor Documentation

#### 4.18.2.1 Key::Key ( void ) [inline]

constructor.

### 4.18.3 Member Function Documentation

#### 4.18.3.1 bool Key::contain ( int64_t *col_id* ) [inline]

check if the key contain a column identifier. column identifier.

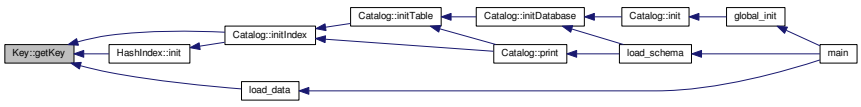
**Return values**

| | |
|---|---|
| *true* | contain |
| *false* | don't contain |

#### 4.18.3.2 std::vector< int64_t >& Key::getKey ( void ) [inline]

get key data.

Here is the caller graph for this function:



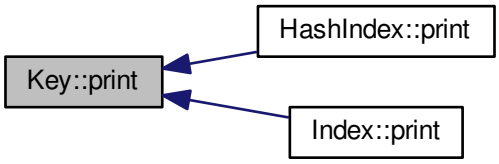

#### 4.18.3.3 Key& Key::operator= ( const Key & *p* ) [inline]

over write operator(=).

#### 4.18.3.4 void Key::print ( void ) [inline]

print key information.

Here is the caller graph for this function:



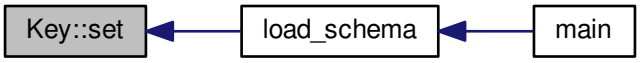

#### 4.18.3.5 void Key::set ( std::vector< int64_t > & *in_key* ) [inline]

set key.

**Parameters**

| | |
|---|---|
| *in_key* | keys(column identifiers) in vector |

Here is the caller graph for this function:




### 4.18.4 Member Data Documentation

#### 4.18.4.1 std::vector< int64_t > Key::key [private]

index iidentifier container

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

- system/schema.h

## 4.19 Memory Class Reference

`#include <mymemory.h>`

### Public Member Functions

- int64_t alloc (char ∗&p, int64_t size)
- int64_t free (char ∗p, int64_t size)
- int init (int64_t total, int64_t mins)
- int print (void)
- int shut (void)

### Private Member Functions

- int64_t alloc_default (char ∗&p, int64_t size)
- unsigned int slot (int64_t size)

### Private Attributes

- char ∗∗ m_array_list
- char ∗ m_curr
- char ∗ m_head
- int64_t m_mins
- char ∗ m_tail
- int64_t m_total

## 4.19.1 Member Function Documentation

#### 4.19.1.1 int64_t Memory::alloc ( char ∗& *p*, int64_t *size* )

alloc db memory for inside usage.

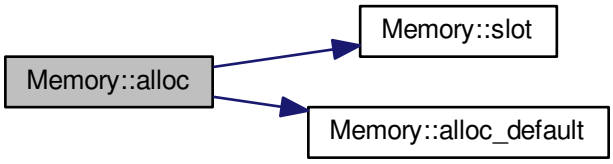
**Parameters**

| | |
|---|---|
| *p* | store the pointer result allocated from db memory |
| *size* | required size by caller, power of 2 |

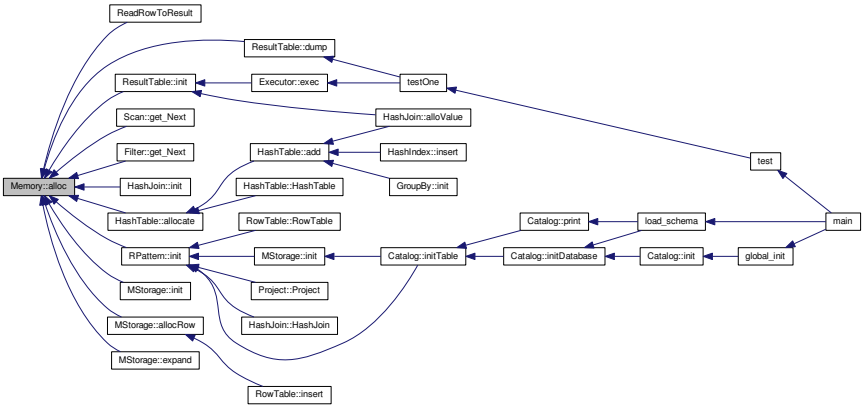
**Return values**

| | |
|---|---|
| *==size* | successfuly allocated from db memory |
| *<=0* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.19.1.2 int64_t Memory::alloc_default ( char *& *p*, int64_t *size* ) [private]

default alloc from db memory when free list has no free memory of this size

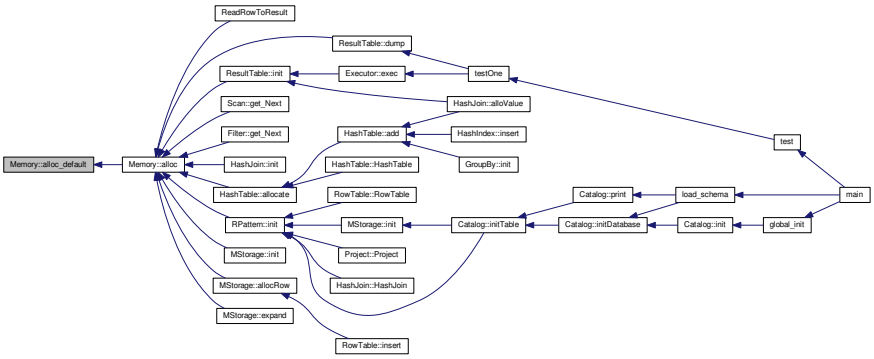
**Parameters**

| | |
|---|---|
| *size* | required size by caller, power of 2 |

**Return values**

| | |
|---|---|
| *==size* | successfuly allocated from db memory |
| *<=0* | failure |

Here is the caller graph for this function:




#### 4.19.1.3 int64_t Memory::free ( char * *p*, int64_t *size* )

free memory to db memory.

**Parameters**

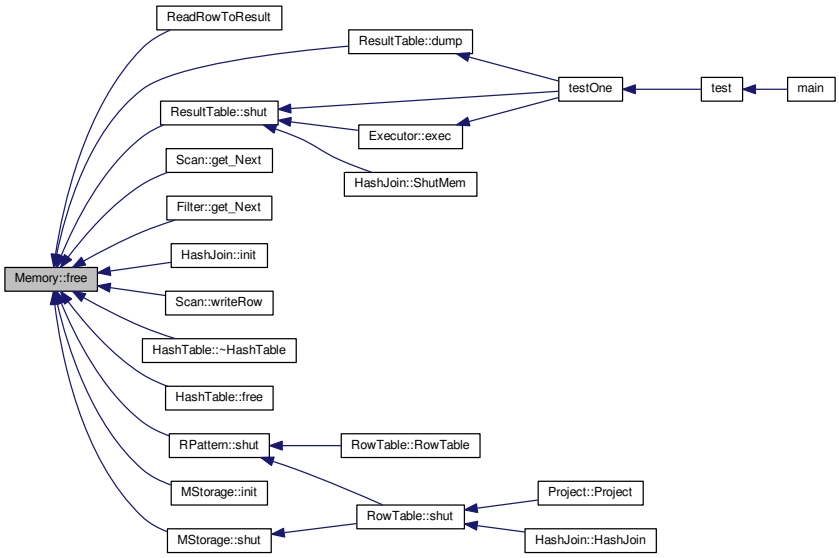
| | |
|---|---|
| *p* | the pointer of memory to free |
| *size* | provided by caller, power of 2 |

**Return values**

| | |
|---|---|
| *==size* | successfuly free to db memory |
| *<=0* | failure |

Here is the call graph for this function:

Here is the caller graph for this function:




#### 4.19.1.4 int Memory::init ( int64_t *total*, int64_t *mins* )

init db memory.

**Parameters**

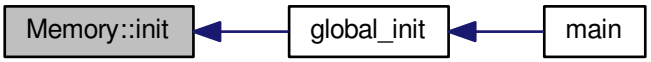
| | |
|---|---|
| *total* | total size allocated from operate system, usually large enough |
| *mins* | minimux size db object allocated from db memory |

**Return values**

| | |
|---|---|
| *==0* | success |
| *<0* | failure |

Here is the call graph for this function:

Here is the caller graph for this function:




#### 4.19.1.5 int Memory::print ( void )

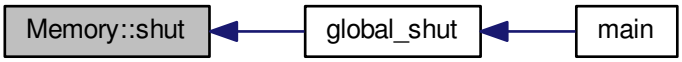
print memory usage.

Here is the call graph for this function:


#### 4.19.1.6 int Memory::shut ( void )

free db memory to operate system.

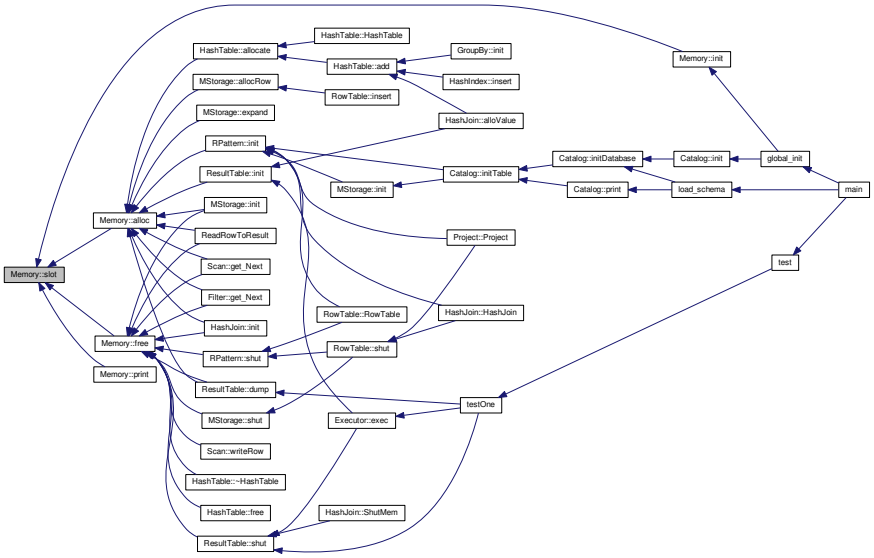
Here is the caller graph for this function:



#### 4.19.1.7 unsigned int Memory::slot ( int64_t *size* ) [private]

calculate position of free list.

Here is the caller graph for this function:




### 4.19.2 Member Data Documentation

#### 4.19.2.1 char** Memory::m_array_list [private]

free arrray list

#### 4.19.2.2 char* Memory::m_curr [private]

pointer of db memory already in use

#### 4.19.2.3 char* Memory::m_head [private]

db memory pointer, pointer of a large memory allocated from operate system

#### 4.19.2.4 int64_t Memory::m_mins [private]

minimux size to alloc, at least sizeof(void*), recommend 8

#### 4.19.2.5 char* Memory::m_tail [private]

end pointer of db memory allocated from operate system

#### 4.19.2.6 int64_t Memory::m_total [private]

total size of database system

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

- system/mymemory.h
- system/mymemory.cc

## 4.20 MStorage Class Reference

`#include <rowtable.h>`

### Public Member Functions

- int64_t allocRow (char ∗&pointer)
- int64_t getRecordNum (void)
- char ∗ getRow (int64_t record_rank)
- bool init (int64_t record_size)
- bool init (int64_t record_size, int64_t init_slot_cap, int64_t per_size)
- void shut (void)

### Private Member Functions

- bool expand (void)

### Private Attributes

- char ∗ ms_memory
- int64_t ms_memory_size
- int64_t ms_record_num
- int64_t ms_record_per_slot
- int64_t ms_record_size
- int64_t ms_slot_size
- int64_t ms_slots_cap
- int64_t ms_slots_num
- char ∗∗ ms_slots_point
- char pad [128-7 ∗sizeof(int64_t)-3 ∗sizeof(void ∗)]

### 4.20.1 Detailed Description

definition of MStorage, table storage manager.

### 4.20.2 Member Function Documentation

#### 4.20.2.1 int64_t MStorage::allocRow ( char ∗& *pointer* ) [inline]

alloc an empty row.

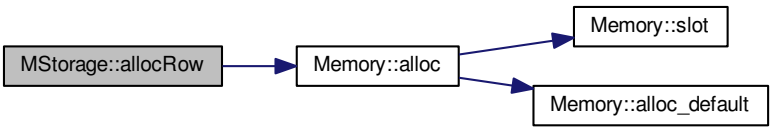
**Parameters**

| | |
|---|---|
| *pointer* | reference of pointer result |

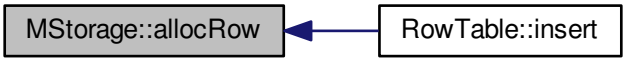
**Return values**

| | |
|---|---|
| *>=0* | row rank in all table |
| *<0* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.20.2.2 bool MStorage::expand ( void ) [inline],[private]

expand slots for more storage avaliable for this table.

**Return values**

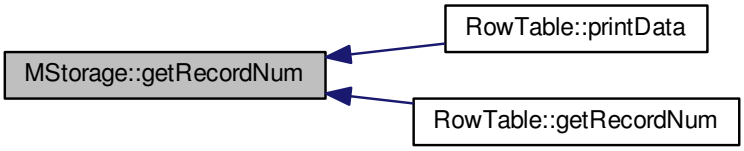
| | |
|---|---|
| *true* | success |
| *false* | lack memory |

Here is the call graph for this function:


#### 4.20.2.3 int64_t MStorage::getRecordNum ( void ) [inline]

get the last record rank till now.

Here is the caller graph for this function:




#### 4.20.2.4 char* MStorage::getRow ( int64_t *record_rank* ) [inline]

get the pointer of a row specified by record_rank.

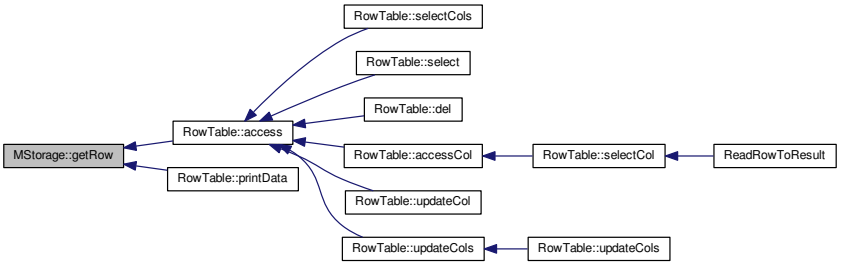
**Parameters**

| | |
|---|---|
| *record_rank* | the n th row in the table |

**Return values**

| | |
|---|---|
| *!=NULL* | valid |
| *==NULL* | param error |

Here is the caller graph for this function:




#### 4.20.2.5 bool MStorage::init ( int64_t *record_size* ) [inline]

mkae sizeof(MStorage)==128, managed by g_memory init, allocate memory and initial setting.

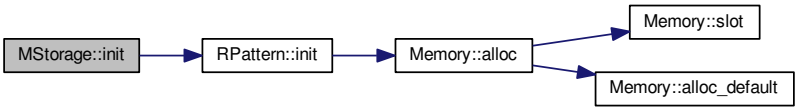
**Parameters**

| | |
|---|---|
| *record_size* | size of a row record |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:

#### 4.20.2.6 bool MStorage::init ( int64_t *record_size,* int64_t *init_slot_cap,* int64_t *per_size* ) [inline]

init, allocate memory and initial setting.

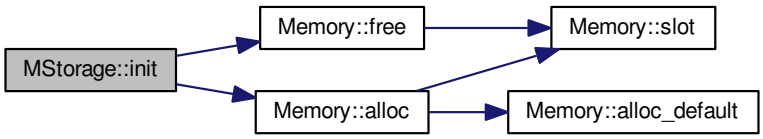
**Parameters**

| | |
|---|---|
| *record_size* | size of a row record |
| *init_slot_cap* | maximum slots initial set |
| *per_size* | slot size, power of 2 |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

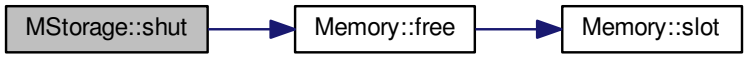
Here is the call graph for this function:



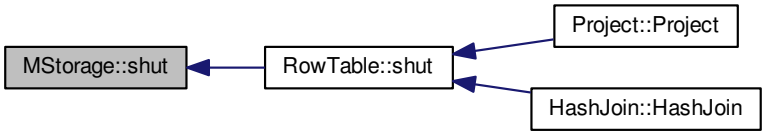

#### 4.20.2.7 void MStorage::shut ( void ) [inline]

shut down, free memory to g_memory.

Here is the call graph for this function:




Here is the caller graph for this function:



### 4.20.3 Member Data Documentation

#### 4.20.3.1 char* MStorage::ms_memory [private]

memory for slots

#### 4.20.3.2 int64_t MStorage::ms_memory_size [private]

memory size

#### 4.20.3.3 int64_t MStorage::ms_record_num [private]

current record used

#### 4.20.3.4 int64_t MStorage::ms_record_per_slot [private]

record number stored in a slot

#### 4.20.3.5 int64_t MStorage::ms_record_size [private]

size per record

#### 4.20.3.6 int64_t MStorage::ms_slot_size [private]

memory size per slot

#### 4.20.3.7 int64_t MStorage::ms_slots_cap [private]

maximum number of slots

#### 4.20.3.8 int64_t MStorage::ms_slots_num [private]

current slots used

#### 4.20.3.9 char** MStorage::ms_slots_point [private]

each of array stores a pointer to a slot

#### 4.20.3.10 char MStorage::pad[128-7 *sizeof(int64_t)-3 *sizeof(void *)] [private]

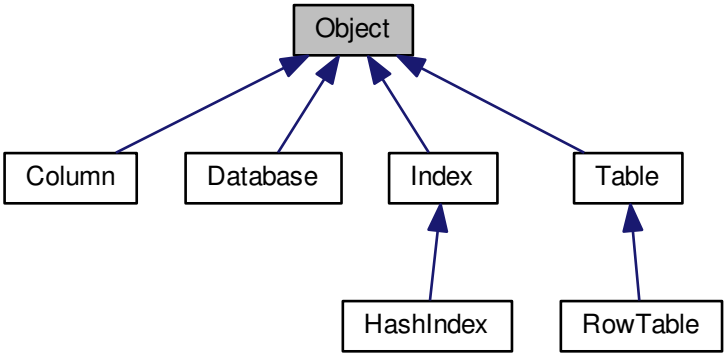
The documentation for this class was generated from the following file:

- system/rowtable.h

## 4.21 Object Class Reference

`#include <schema.h>`

Inheritance diagram for Object:




### Public Member Functions

- Object (int64_t o_id, ObjectType o_type, const char *o_name)
- bool changeName (char *o_name)
- int64_t getOid (void)
- char * getOname (void)
- ObjectType getOtype (void)
- virtual void print (void)
- virtual bool shut (void)

### Private Attributes

- int64_t o_id
- char o_name [OBJ_NAME_MAX]
- ObjectType o_type

### 4.21.1 Detailed Description

definition of Object, basic element in database.

### 4.21.2 Constructor & Destructor Documentation

#### 4.21.2.1 Object::Object ( int64_t *o_id,* ObjectType *o_type,* const char * *o_name* ) [inline]

constructor.

**Parameters**

| | |
|---|---|
| *o_id* | object identifier |
| *o_type* | object type |
| *o_name* | object name |

### 4.21.3 Member Function Documentation

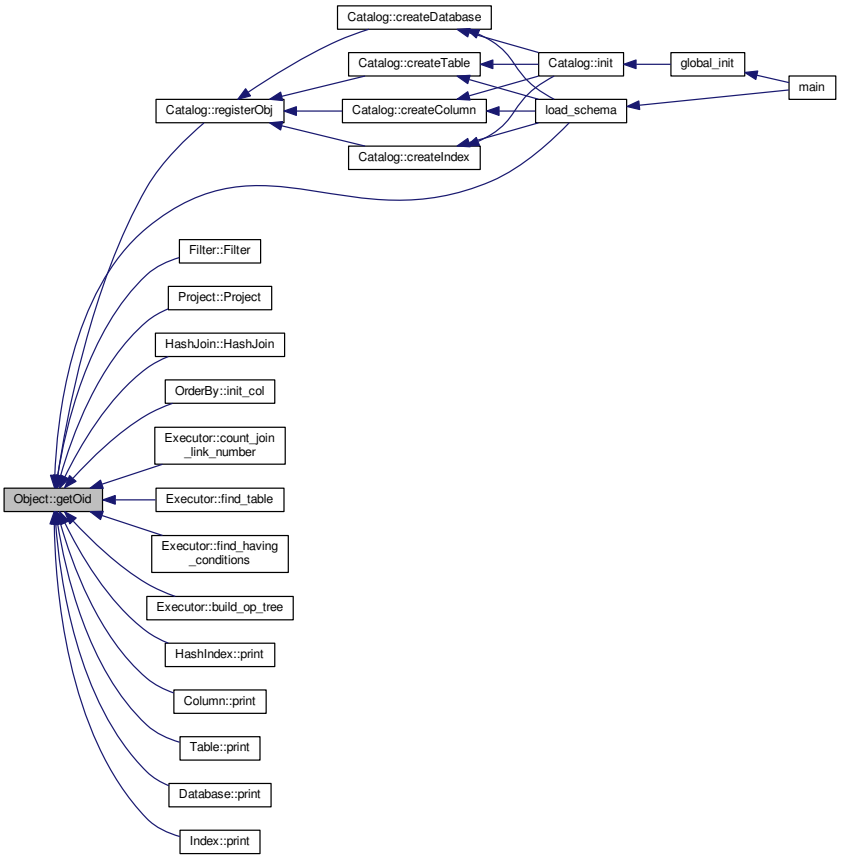
#### 4.21.3.1 bool Object::changeName ( char * *o_name* ) [inline]

change object name(not in use).

#### 4.21.3.2 int64_t Object::getOid ( void ) [inline]

get identifier of object.

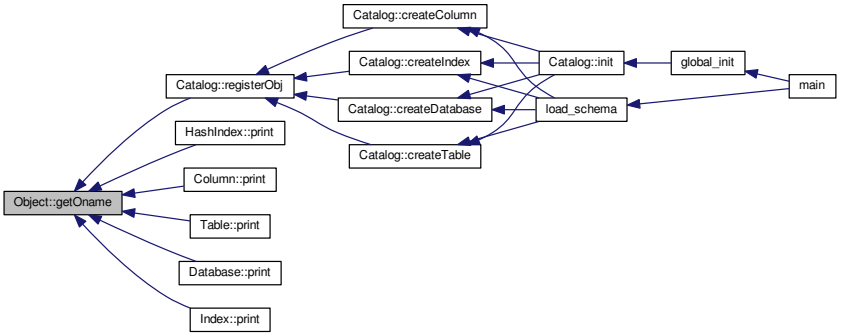
Here is the caller graph for this function:



#### 4.21.3.3 char* Object::getOname ( void ) [inline]

get object name.

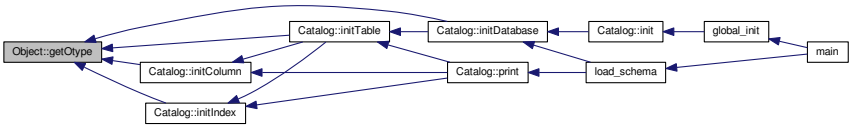
Here is the caller graph for this function:




#### 4.21.3.4 ObjectType Object::getOtype ( void ) [inline]

get object type.

Here is the caller graph for this function:

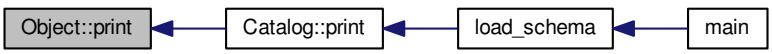



#### 4.21.3.5 virtual void Object::print ( void ) [inline], [virtual]

print the object infomation.

Reimplemented in Index, Database, Table, Column, and HashIndex.

Here is the caller graph for this function:



#### 4.21.3.6 virtual bool Object::shut ( void ) [inline], [virtual]

shut down the object.

Reimplemented in Index, Database, Table, RowTable, Column, and HashIndex.

### 4.21.4 Member Data Documentation

#### 4.21.4.1 int64_t Object::o_id [private]

object identifier

#### 4.21.4.2 char Object::o_name[OBJ_NAME_MAX] [private]

object name, max length OBJ_NAME_MAX

#### 4.21.4.3 ObjectType Object::o_type [private]

object type

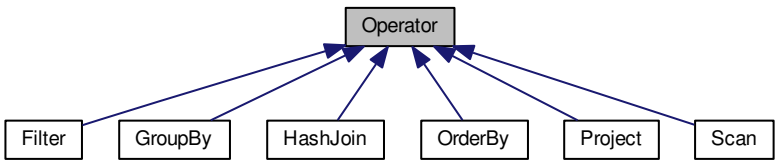
The documentation for this class was generated from the following file:

- system/schema.h

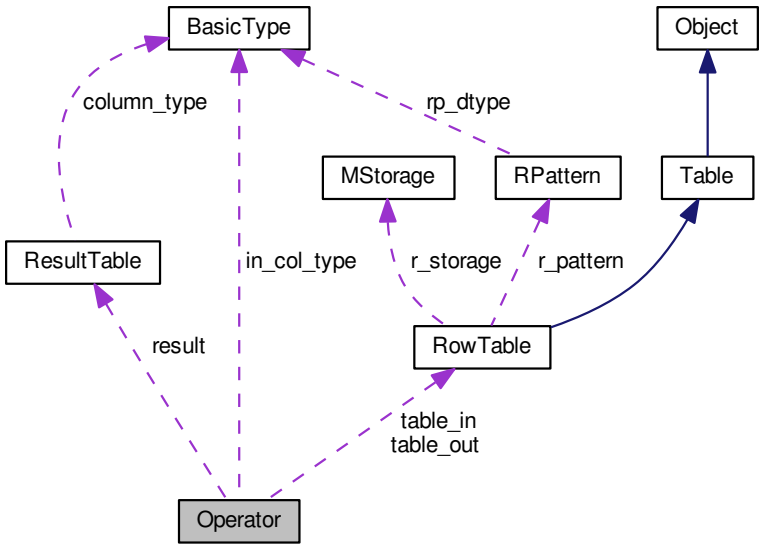
## 4.22 Operator Class Reference

```
#include <executor.h>
```

Inheritance diagram for Operator:



Collaboration diagram for Operator:




## Public Member Functions

- Operator (void)
- virtual bool close ()=0
- virtual bool get_Next (ResultTable *result)=0
- RowTable * getTableOut ()
- virtual bool init ()=0
- virtual bool is_End ()=0

## Public Attributes

- int64_t Ope_id = 0

## Protected Attributes

- BasicType ** in_col_type
- ResultTable result
- RowTable * table_in [4]
- RowTable * table_out
- int64_t table_out_col_num = 0

### 4.22.1 Detailed Description

definition of Operator.

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 Operator::Operator ( void ) [inline]

construction of Operator.

### 4.22.3 Member Function Documentation

#### 4.22.3.1 virtual bool Operator::close ( ) [pure virtual]

close the operator and release memory

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implemented in GroupBy, OrderBy, Project, HashJoin, Filter, and Scan.

#### 4.22.3.2 virtual bool Operator::get_Next ( ResultTable * *result* ) [pure virtual]

get next record and put it in resulttable

**Parameters**

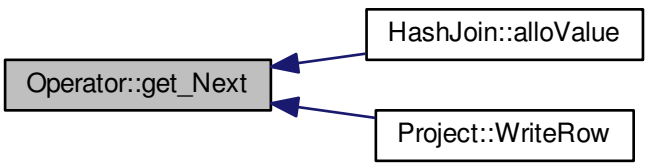
| | |
|---|---|
| *result* | buffer to store the record |

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implemented in GroupBy, OrderBy, Project, HashJoin, Filter, and Scan.

Here is the caller graph for this function:



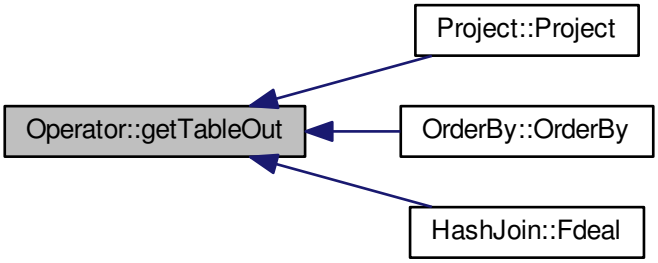
#### 4.22.3.3 RowTable* Operator::getTableOut ( ) [inline]

get the output result table of this Operator

**Return values**

| | |
|---|---|
| *table_out* | |

Here is the caller graph for this function:




#### 4.22.3.4 virtual bool Operator::init ( ) [pure virtual]

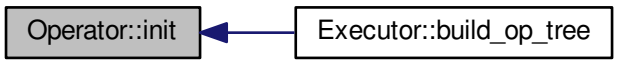
operator init

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implemented in GroupBy, OrderBy, Project, HashJoin, Filter, and Scan.

Here is the caller graph for this function:



#### 4.22.3.5 virtual bool Operator::is_End ( ) [pure virtual]

where this operator is end

**Return values**

| | |
|---|---|
| *false* | not end |
| *true* | run end |

Implemented in GroupBy, OrderBy, Project, HashJoin, Filter, and Scan.

### 4.22.4 Member Data Documentation

#### 4.22.4.1 BasicType** Operator::in_col_type [protected]

column types of input tables.

#### 4.22.4.2 int64_t Operator::Ope_id = 0

#### 4.22.4.3 ResultTable Operator::result [protected]

each operator got its own ResultTable(Buffer) except Scan Operator.

#### 4.22.4.4 RowTable* Operator::table_in[4] [protected]

tables input in each operation.

#### 4.22.4.5 RowTable* Operator::table_out [protected]

This is only for RPattern and getColumnRank We only need its skeleton: cols_name[], cols_id[] and col_num. record data inside is meaningless.

#### 4.22.4.6 int64_t Operator::table_out_col_num = 0 [protected]

the number of column in table_out.

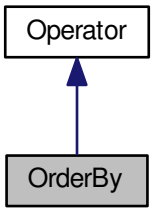
The documentation for this class was generated from the following file:

- system/executor.h

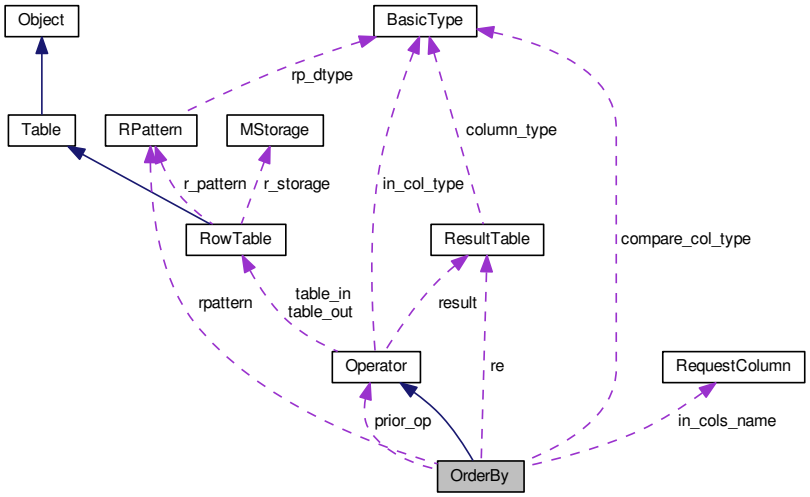
## 4.23 OrderBy Class Reference

`#include <executor.h>`

Inheritance diagram for OrderBy:




Collaboration diagram for OrderBy:




### Public Member Functions

- OrderBy (Operator ∗op_ptr, int64_t order_by_num, RequestColumn ∗cols_name)
  *construction of OrderBy*
- bool close ()
  *close operator and release memory.*
- int cmp (void ∗l, void ∗r)
  *compare tow buffers*
- bool get_Next (ResultTable ∗result)
  *get next record of operator*

- bool init ()

  *init of project*
- bool is_End ()

  *judge whether is end*
- void quick_sort (char *buffer, int64_t left, int64_t right)

  *quick sort*

### Private Member Functions

- bool check_not_end ()

  *check not end*
- bool check_stop_get_Next (ResultTable *result, char *buffer)
- int64_t * get_compare_col_offset ()

  *get ptr to offset of each compared columns*
- int64_t * get_compare_col_rank ()

  *get ptr to ranks of each compare condtions*
- BasicType ** get_compare_col_type_ptr ()

  *get ptr to column types of compare conditons*
- int64_t get_index ()

  *get index that has been ordered*
- int64_t get_number_of_columns ()

  *get number of columns*
- int64_t get_order_by_num ()

  *get number of order conditions*
- int64_t get_prior_col_rank (int64_t object_oid)

  *get prior column rank*
- Operator * get_prior_operator ()

  *get pointer to prior Operator*
- RowTable * get_prior_tableout ()

  *get prior tableout*
- int64_t get_record_size ()

  *get size of each record*
- ResultTable get_ResultTable ()

  *get temp result table to store result*
- int64_t get_row_length ()

  *get length of each row*
- RPattern get_rpattern ()

  *get rpattern*
- void init_col ()

  *init col*
- void init_col_type ()

  *init col type*
- void qsort_helper (char *buffer)
- void write_result_col (int row, char *buffer)

  *write result col*

### Private Attributes

- int64_t col_num = 0
- int64_t compare_col_offset [4]
- int64_t compare_col_rank [4]
- BasicType * compare_col_type [4]
- int64_t high
- RequestColumn * in_cols_name
- int64_t index = 0
- char * key
- int64_t low
- int64_t order_by_num
- Operator * prior_op
- ResultTable re
- int64_t record_size
- int64_t row_length
- RPattern rpattern

### Additional Inherited Members

## 4.23.1 Detailed Description

definiton of OrderBy

## 4.23.2 Constructor & Destructor Documentation

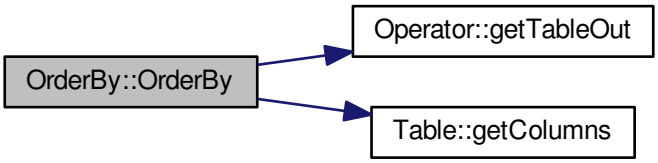
### 4.23.2.1 OrderBy::OrderBy ( Operator * *op_ptr,* int64_t *order_by_num,* RequestColumn * *cols_name* )

construction of OrderBy

**Parameters**

| | |
|---|---|
| *Op* | prior operators |
| *order_by_num* | number of condtions in order |
| *cols_name* | names of columns |

Here is the call graph for this function:



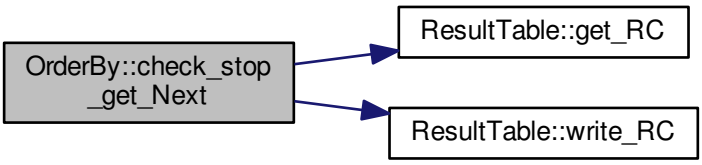
### 4.23.3 Member Function Documentation

#### 4.23.3.1 bool OrderBy::check_not_end ( ) [inline], [private]

check not end

#### 4.23.3.2 bool OrderBy::check_stop_get_Next ( ResultTable * *result*, char * *buffer* ) [inline], [private]

Here is the call graph for this function:




#### 4.23.3.3 bool OrderBy::close ( void ) [virtual]

close operator and release memory.

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

#### 4.23.3.4 int OrderBy::cmp ( void * *l*, void * *r* )

compare tow buffers

**Parameters**

| | |
|---|---|
| *l* | first buffer |
| *r* | second buffer |

**Return values**

| | |
|---|---|
| *2* | for > |
| *1* | for = |

**Return values**

| | |
|---|---|
| *0* | for < |

#### 4.23.3.5 int64_t* OrderBy::get_compare_col_offset ( ) [inline], [private]

get ptr to offset of each compared columns

**Return values**

| | |
|---|---|
| *ptr* | to offset of each compared columns |

#### 4.23.3.6 int64_t* OrderBy::get_compare_col_rank ( ) [inline], [private]

get ptr to ranks of each compare condtions

**Return values**

| | |
|---|---|
| *ptr* | to ranks of each compare condtions |

#### 4.23.3.7 BasicType** OrderBy::get_compare_col_type_ptr ( ) [inline], [private]

get ptr to column types of compare conditons

**Return values**

| | |
|---|---|
| *ptr* | to column types of compare conditons |

#### 4.23.3.8 int64_t OrderBy::get_index ( ) [inline], [private]

get index that has been ordered

**Return values**

| | |
|---|---|
| *index* | that has been ordered |

#### 4.23.3.9 bool OrderBy::get_Next ( ResultTable * *result* ) [virtual]

get next record of operator

**Parameters**

| | |
|---|---|
| *result* | buffer to store result |

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

#### 4.23.3.10 int64_t OrderBy::get_number_of_columns ( ) [inline], [private]

get number of columns

**Return values**

| | |
|---|---|
| *number* | of columns |

#### 4.23.3.11 int64_t OrderBy::get_order_by_num ( ) [inline], [private]

get number of order conditions

**Return values**

| | |
|---|---|
| *number* | of order conditions |

#### 4.23.3.12 int64_t OrderBy::get_prior_col_rank ( int64_t *object_oid* ) [inline], [private]

get prior column rank

**Return values**

| | |
|---|---|
| *prior* | column rank |

#### 4.23.3.13 Operator* OrderBy::get_prior_operator ( ) [inline], [private]

get pointer to prior Operator

**Return values**

| | |
|---|---|
| *pointer* | to prior Operator |

#### 4.23.3.14 RowTable* OrderBy::get_prior_tableout ( ) [inline], [private]

get prior tableout

**Return values**

| | |
|---|---|
| *prior* | tableout |

#### 4.23.3.15 int64_t OrderBy::get_record_size ( ) [inline], [private]

get size of each record

**Return values**

| | |
|---|---|
| *size* | of each record |

#### 4.23.3.16 ResultTable OrderBy::get_ResultTable ( ) [inline], [private]

get temp result table to store result

**Return values**

| | |
|---|---|
| *temp* | result table to store result |

#### 4.23.3.17 int64_t OrderBy::get_row_length ( ) [inline], [private]

get length of each row

**Return values**

| | |
|---|---|
| *length* | of each row |

#### 4.23.3.18 RPattern OrderBy::get_rpattern ( ) [inline], [private]

get rpattern

**Return values**

| | |
|---|---|
| *rpattern* | inside |

#### 4.23.3.19 bool OrderBy::init ( void ) [virtual]

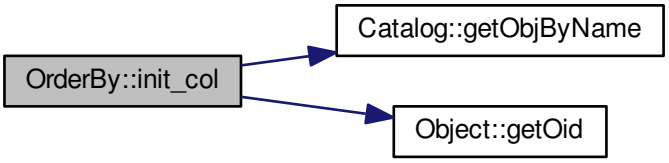
init of project

Implements Operator.

#### 4.23.3.20 void OrderBy::init_col ( ) [inline], [private]

init col

Here is the call graph for this function:




#### 4.23.3.21 void OrderBy::init_col_type ( ) [inline], [private]

init col type

#### 4.23.3.22 bool OrderBy::is_End ( void ) [virtual]

judge whether is end

**Return values**

| | |
|---|---|
| *false* | not end |
| *true* | run end |

Implements Operator.

#### 4.23.3.23 void OrderBy::qsort_helper ( char * *buffer* ) [inline], [private]

#### 4.23.3.24 void OrderBy::quick_sort ( char * *buffer*, int64_t *left*, int64_t *right* )

quick sort

**Parameters**

| | |
|---|---|
| *buffer* | the begin of orderby address |
| *left* | left (low)side of order sequence |
| *right* | right (high)side of order sequence |

#### 4.23.3.25 void OrderBy::write_result_col ( int *row,* char * *buffer* ) [inline], [private]

write result col

Here is the call graph for this function:


### 4.23.4 Member Data Documentation

#### 4.23.4.1 int64_t OrderBy::col_num = 0 [private]

number of columns

#### 4.23.4.2 int64_t OrderBy::compare_col_offset[4] [private]

offset of each compared columns

#### 4.23.4.3 int64_t OrderBy::compare_col_rank[4] [private]

ranks of each compare condtions

#### 4.23.4.4 BasicType* OrderBy::compare_col_type[4] [private]

column types of compare conditons

#### 4.23.4.5 int64_t OrderBy::high [private]

inside class use

#### 4.23.4.6 RequestColumn* OrderBy::in_cols_name [private]

inside class use

#### 4.23.4.7 int64_t OrderBy::index = 0 [private]

index that has been ordered

#### 4.23.4.8 char* OrderBy::key [private]

inside class use

#### 4.23.4.9 int64_t OrderBy::low [private]

inside class use

#### 4.23.4.10 int64_t OrderBy::order_by_num [private]

number of order conditions

#### 4.23.4.11 Operator* OrderBy::prior_op [private]

prior Operator

#### 4.23.4.12 ResultTable OrderBy::re [private]

temp result table to store result

#### 4.23.4.13 int64_t OrderBy::record_size [private]

size of each record

#### 4.23.4.14 int64_t OrderBy::row_length [private]

length of each row

#### 4.23.4.15 RPattern OrderBy::rpattern [private]

inside class use

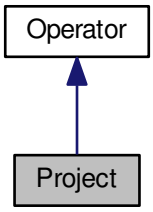
The documentation for this class was generated from the following files:

- system/executor.h
- system/executor.cc

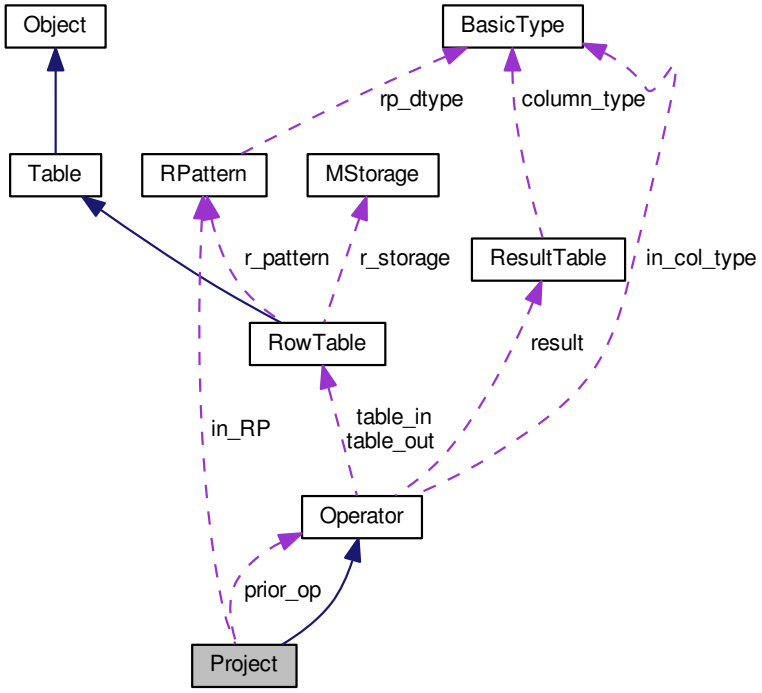
## 4.24 Project Class Reference

`#include <executor.h>`

Inheritance diagram for Project:




Collaboration diagram for Project:




### Public Member Functions

- Project (Operator *Op, int64_t col_tot, RequestColumn *cols_name)

*construction of project*

- bool close ()
- bool get_Next (ResultTable *result)

  *get next record of operator*
- bool init ()

  *init of project*
- bool is_End ()

  *judge whether is end*
- void ResInit (int in_colnum)

  *init result table*
- void ShutMem ()

  *shut memory*
- bool WriteRow (ResultTable *result)

  *Write a row.*

### Private Attributes

- int64_t col_rank [4]
- int64_t col_tot
- RPattern in_RP
- Operator * prior_op

### Additional Inherited Members

#### 4.24.1 Detailed Description

definition of project

#### 4.24.2 Constructor & Destructor Documentation

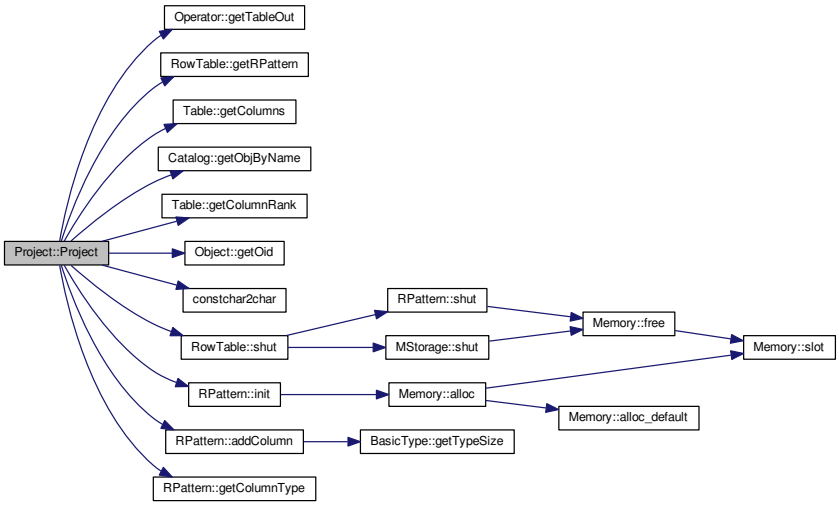
##### 4.24.2.1 Project::Project ( Operator * *Op*, int64_t *col_tot*, RequestColumn * *cols_name* )

construction of project

**Parameters**

| | |
|---|---|
| *Op* | prior operators |
| *col_tot* | total columns projected |
| *cols_name* | columns name of projected column |

Here is the call graph for this function:




## 4.24.3 Member Function Documentation

### 4.24.3.1 bool Project::close ( void ) [virtual]

close operator and release memory.

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

### 4.24.3.2 bool Project::get_Next ( ResultTable * *result* ) [virtual]

get next record of operator

**Parameters**

| | |
|---|---|
| *result* | buffer to store result |

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

#### 4.24.3.3 bool Project::init ( void ) [virtual]

init of project

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

#### 4.24.3.4 bool Project::is_End ( void ) [virtual]

judge whether is end

**Return values**

| | |
|---|---|
| *false* | not end |
| *true* | run end |

Implements Operator.

#### 4.24.3.5 void Project::ResInit ( int *in_colnum* ) [inline]

init result table

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:


#### 4.24.3.6 void Project::ShutMem ( ) [inline]

shut memory

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

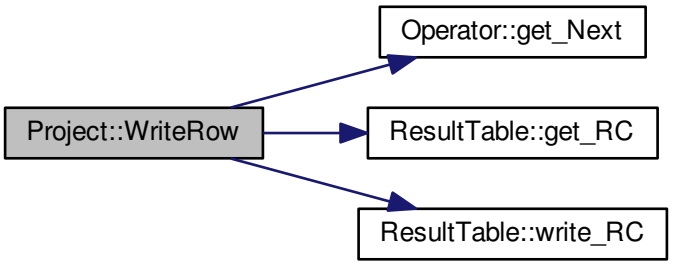
#### 4.24.3.7 bool Project::WriteRow ( ResultTable * *result* ) [inline]

Write a row.

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:




### 4.24.4 Member Data Documentation

#### 4.24.4.1 int64_t Project::col_rank[4] [private]

rank sets of columns projected

#### 4.24.4.2 int64_t Project::col_tot [private]

number of columns being projected.

#### 4.24.4.3 RPattern Project::in_RP [private]

inside class use

#### 4.24.4.4 Operator* Project::prior_op [private]

prior operators

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

- system/executor.h
- system/executor.cc

## 4.25 RequestColumn Struct Reference

`#include <executor.h>`

### Public Attributes

- AggrerateMethod aggrerate_method
- char name [128]

### 4.25.1 Detailed Description

definition of request column.

### 4.25.2 Member Data Documentation

#### 4.25.2.1 AggrerateMethod RequestColumn::aggrerate_method

#### 4.25.2.2 char RequestColumn::name[128]

name of column

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

- system/executor.h

## 4.26 RequestTable Struct Reference

`#include <executor.h>`

### Public Attributes

- char name [128]

### 4.26.1 Detailed Description

definition of request table.

### 4.26.2 Member Data Documentation

#### 4.26.2.1 char RequestTable::name[128]

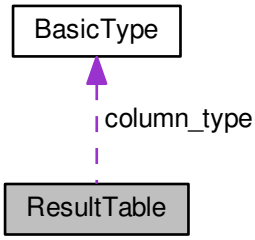
The documentation for this struct was generated from the following file:

- system/executor.h

## 4.27 ResultTable Class Reference

`#include <executor.h>`

Collaboration diagram for ResultTable:




### Public Member Functions

- int dump (FILE *fp)
- char * get_RC (int row, int column)
- int init (BasicType *col_types[ ], int col_num, int64_t capicity=1024)
- int print (void)
- int shut (void)
- int write_RC (int row, int column, void *data)

### Public Attributes

- char * buffer
- int64_t buffer_size
- int column_number
- BasicType ** column_type
- int * offset
- int offset_size
- int row_capicity
- int row_length
- int row_number

### 4.27.1 Detailed Description

definition of result table.

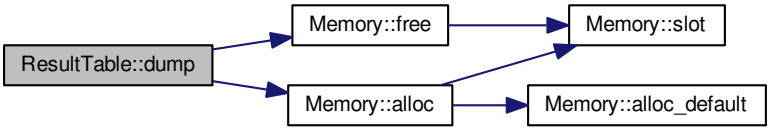
### 4.27.2 Member Function Documentation

#### 4.27.2.1 int ResultTable::dump ( FILE * *fp* )

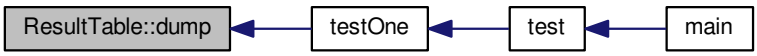
write to file with FILE *fp

dump result to fp

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.27.2.2 char * ResultTable::get_RC ( int *row,* int *column* )

calculate the char pointer of data spcified by row and column id you should set up column_type,then call init function

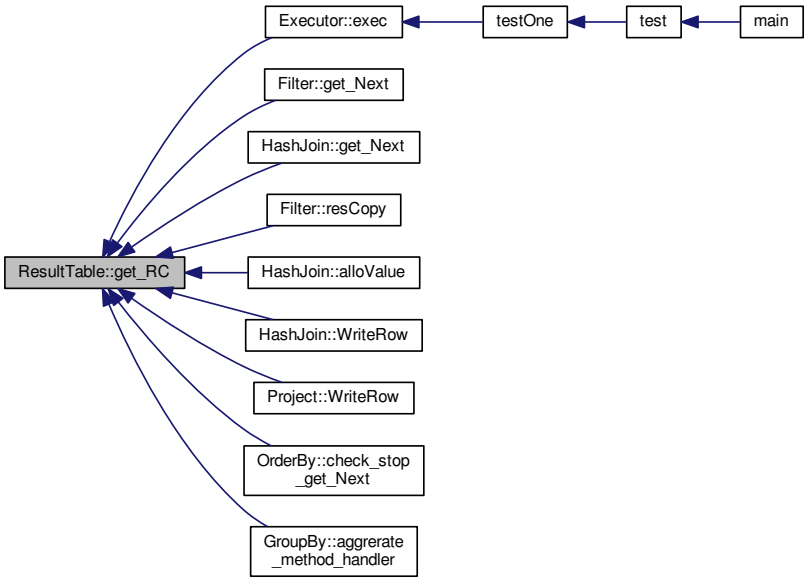
**Parameters**

| | |
|---|---|
| *row* | row id in result table |
| *column* | column id in result table |

**Return values**

| | |
|---|---|
| *!=NULL* | pointer of a column |
| *==NULL* | error |

Here is the caller graph for this function:




#### 4.27.2.3 int ResultTable::init ( BasicType * *col_types[ ]*, int *col_num*, int64_t *capicity* = 1024 )

init alloc memory and set initial value array of column type pointers number of columns in this ResultTable

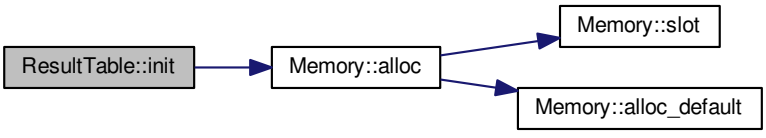
**Parameters**

| | |
|---|---|
| *capicity* | buffer_size, power of 2 |

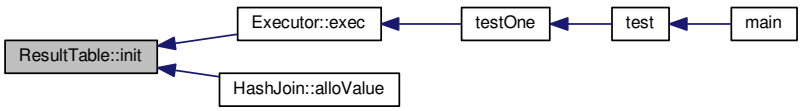
**Return values**

| | |
|---|---|
| *>0* | success |
| *<=0* | failure |

Here is the call graph for this function:



Here is the caller graph for this function:




#### 4.27.2.4 int ResultTable::print ( void )

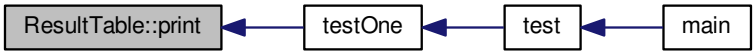
print result table, split by '', output a line per row

**Return values**

| | |
|---|---|
| *the* | number of rows printed |

print

Here is the caller graph for this function:

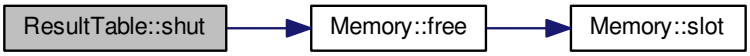



#### 4.27.2.5 int ResultTable::shut ( void )

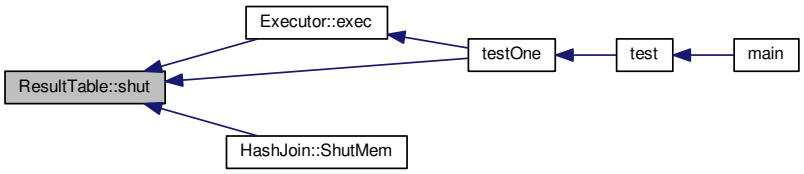
free memory of this result table to g_memory

shut memory

Here is the call graph for this function:



Here is the caller graph for this function:




#### 4.27.2.6 int ResultTable::write_RC ( int *row,* int *column,* void * *data* )

write data to position row,column

**Parameters**

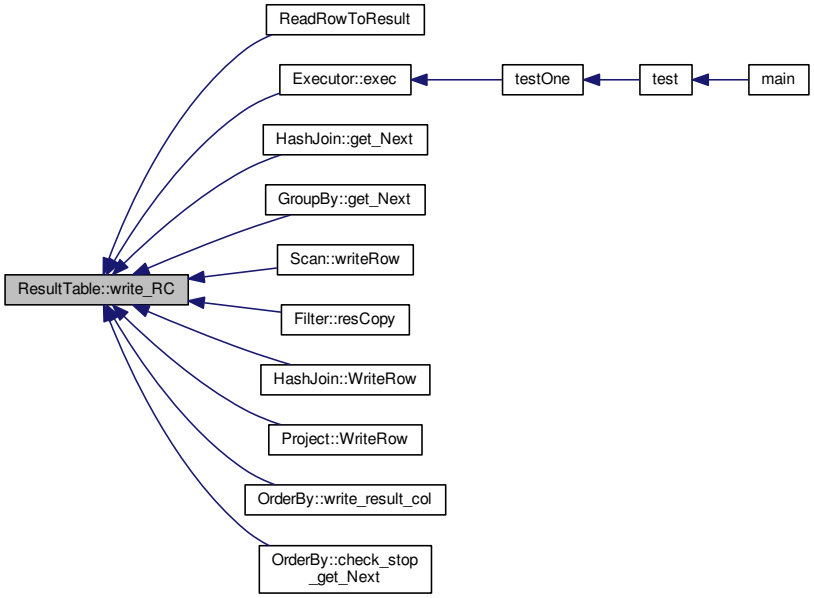
| | |
|---|---|
| *row* | row id in result table |
| *column* | column id in result table data pointer of a column |

**Return values**

| | |
|---|---|
| *!=NULL* | pointer of a column |
| *==NULL* | error |

write rc with data

Here is the caller graph for this function:




### 4.27.3 Member Data Documentation

#### 4.27.3.1 char* ResultTable::buffer

pointer of buffer alloced from g_memory

#### 4.27.3.2 int64_t ResultTable::buffer_size

size of buffer, power of 2

#### 4.27.3.3 int ResultTable::column_number

columns number that a result row consist of

#### 4.27.3.4 BasicType** ResultTable::column_type

each column data type

#### 4.27.3.5 int* ResultTable::offset

#### 4.27.3.6 int ResultTable::offset_size

#### 4.27.3.7 int ResultTable::row_capicity

maximum capicity of rows according to buffer size and length of row MAXIMUN OF ROW

#### 4.27.3.8 int ResultTable::row_length

length per result row

#### 4.27.3.9 int ResultTable::row_number

current usage of rows CURRENT NUMBER OF ROW

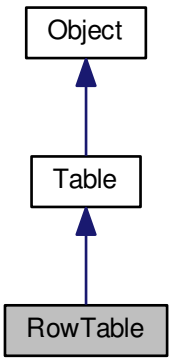
The documentation for this class was generated from the following files:

- system/executor.h
- system/executor.cc

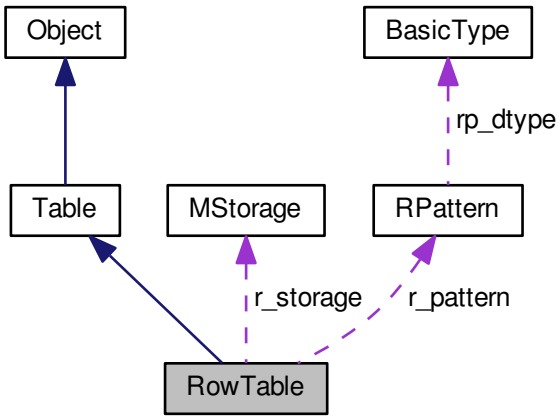
## 4.28 RowTable Class Reference

```
#include <rowtable.h>
```

Inheritance diagram for RowTable:



Collaboration diagram for RowTable:




## Public Member Functions

- RowTable (int64_t r_id, const char *r_name)
- bool del (int64_t record_rank)
- bool del (char *row_pointer)
- bool finish (void)
- MStorage & getMStorage (void)
- int64_t getRecordNum (void)
- void * getRecordPtr (int64_t row_rank)
- RPattern & getRPattern (void)
- bool init (void)
- bool insert (char *source)
- bool insert (char *columns[])
- bool loadData (const char *filename)
- bool printData (void)
- bool select (int64_t record_rank, char *dest)
- bool select (char *row_pointer, char *dest)
- bool selectCol (int64_t record_rank, int64_t column_rank, char *dest)
- bool selectCol (char *row_pointer, int64_t column_rank, char *dest)
- bool selectCols (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *dest)
- bool selectCols (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *dest)
- bool shut (void)
- bool updateCol (char *row_pointer, int64_t column_rank, char *source)
- bool updateCol (int64_t record_rank, int64_t column_rank, char *source)
- bool updateCols (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *source)
- bool updateCols (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *source)
- bool updateCols (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *source[])
- bool updateCols (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *source[])

### Private Member Functions

- bool access (int64_t record_rank, char ∗&pointer)
- bool accessCol (int64_t record_rank, int64_t column_rank, char ∗&pointer)
- bool invalid (char ∗record_ptr)
- bool isValid (char ∗record_ptr)

### Private Attributes

- RPattern r_pattern
- MStorage r_storage

## 4.28.1 Detailed Description

definition of class RowTable.

## 4.28.2 Constructor & Destructor Documentation

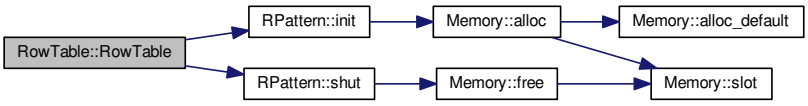
#### 4.28.2.1 RowTable::RowTable ( int64_t *r_id,* const char ∗ *r_name* ) [inline]

constructor.

**Parameters**

| | |
|---|---|
| *r_id* | table ideitifer |
| *r_name* | table name |

Here is the call graph for this function:




## 4.28.3 Member Function Documentation

#### 4.28.3.1 bool RowTable::access ( int64_t *record_rank,* char ∗& *pointer* ) [private]

get a row record pointer.

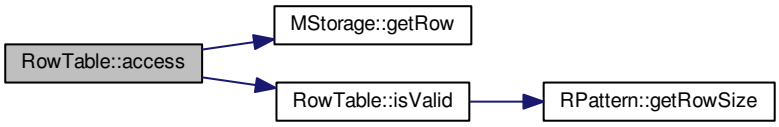
**Parameters**

| | |
|---|---|
| *record_rank* | the n th record in the table |
| *pointer* | result pointer to return |

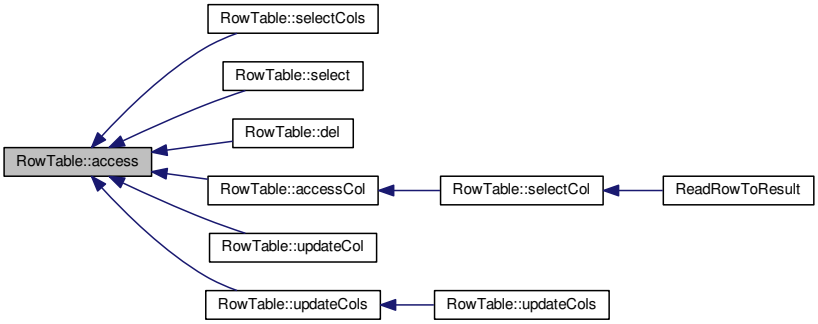
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.28.3.2 bool RowTable::accessCol ( int64_t *record_rank*, int64_t *column_rank*, char *&* *pointer* ) [private]

get a column of record pointer.

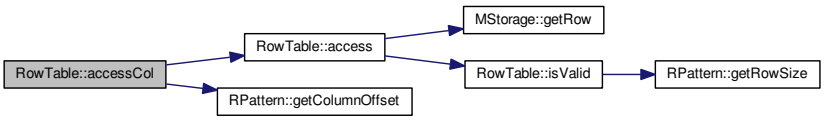
**Parameters**

| | |
|---|---|
| *record_rank* | the n th record in the table |
| *column_rank* | the n th cilumn in the pattern |
| *pointer* | result pointer to return |

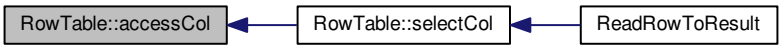
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.28.3.3 bool RowTable::del ( int64_t *record_rank* ) [virtual]

del a row.

**Parameters**

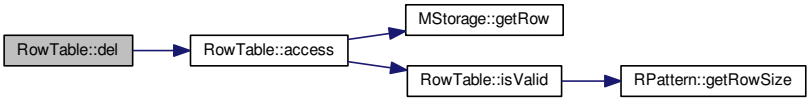
| | |
|---|---|
| *row_rank* | the n th record of the table |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.4 bool RowTable::del ( char * *row_pointer* ) [virtual]

del a row(not in use).

**Parameters**

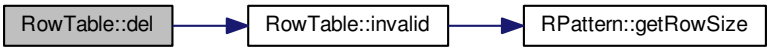
| | |
|---|---|
| *row_pointer* | the pointer of a row |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.5 bool RowTable::finish ( void ) [virtual]

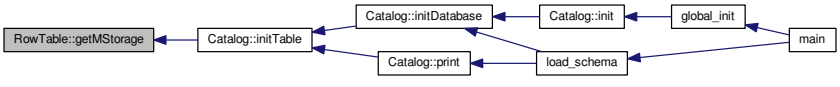
finish, leave it empty.

Reimplemented from Table.

#### 4.28.3.6 MStorage& RowTable::getMStorage ( void ) [inline]

get storage of table.

Here is the caller graph for this function:



#### 4.28.3.7 int64_t RowTable::getRecordNum ( void ) [inline],[virtual]

get the last record rank.

Reimplemented from Table.

Here is the call graph for this function:


#### 4.28.3.8 void* RowTable::getRecordPtr ( int64_t *row_rank* ) [inline],[virtual]

get row record pointer.

**Parameters**

| | |
|---|---|
| *row_rank* | the n th record in thetable |

**Return values**

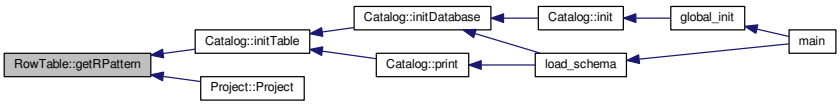
| | |
|---|---|
| *!=NULL* | success |
| *==NULL* | failure |

Reimplemented from Table.

#### 4.28.3.9 RPattern& RowTable::getRPattern ( void ) [inline]

get pattern of table.

Here is the caller graph for this function:




#### 4.28.3.10 bool RowTable::init ( void ) [virtual]

init, leave it empty.

Reimplemented from Table.

#### 4.28.3.11 bool RowTable::insert ( char * *source* ) [virtual]

insert a row.

**Parameters**

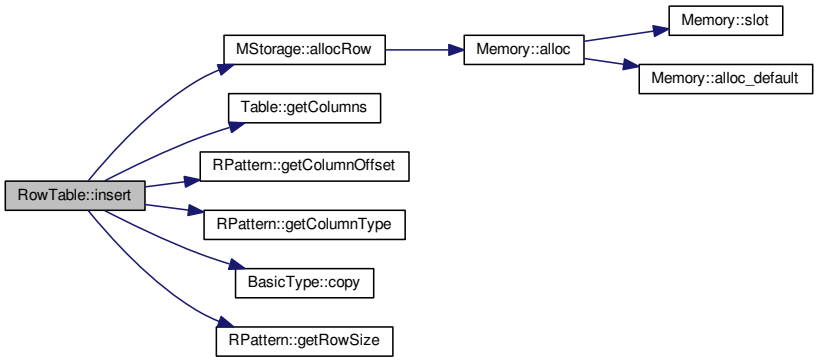
| | |
|---|---|
| *source* | buffer of a row in pattern |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.12 bool RowTable::insert ( char * *columns[ ]* ) [virtual]

insert a row.

**Parameters**

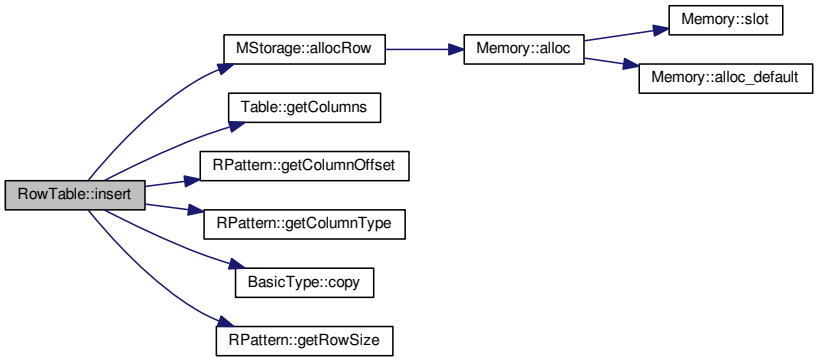
| | |
|---|---|
| *columns* | each element of the array pointed to a column data |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.13 bool RowTable::invalid ( char * *record_ptr* ) [inline],[private]

make the record invalid when del the record

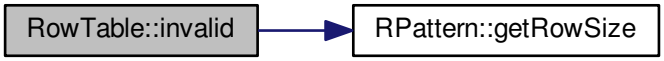
**Parameters**

| | |
|---|---|
| *record_ptr* | the pointer of a record |

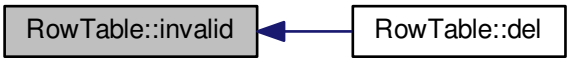
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:



Here is the caller graph for this function:




#### 4.28.3.14 bool RowTable::isValid ( char * *record_ptr* ) [inline], [private]

get result,whether the record is valid or not

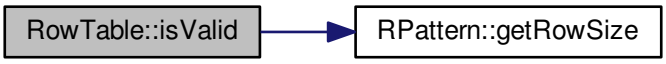
**Parameters**

| | |
|---|---|
| *record_ptr* | the pointer of a record |

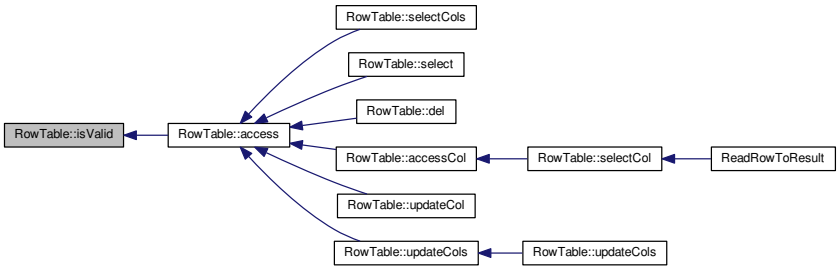
**Return values**

| | |
|---|---|
| *true* | valid |
| *false* | invalid |

Here is the call graph for this function:




Here is the caller graph for this function:



#### 4.28.3.15 bool RowTable::loadData ( const char * *filename* ) [virtual]

load data of the table(not in use).

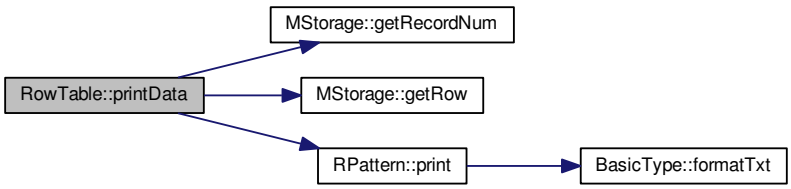
Reimplemented from Table.

#### 4.28.3.16 bool RowTable::printData ( void ) [virtual]

print table data, for debug.

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.17 bool RowTable::select ( int64_t *record_rank*, char * *dest* ) [virtual]

select all columns' data.

**Parameters**

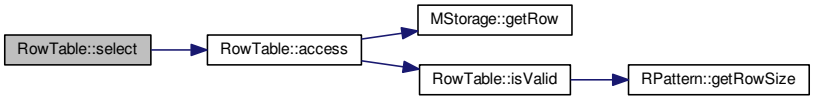
| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.18 bool RowTable::select ( char * *row_pointer,* char * *dest* ) [virtual]

select all columns' data.

**Parameters**

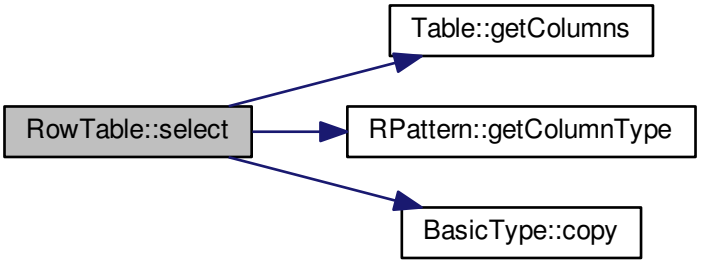
| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.19 bool RowTable::selectCol ( int64_t *record_rank,* int64_t *column_rank,* char * *dest* ) [virtual]

select one column data.

**Parameters**

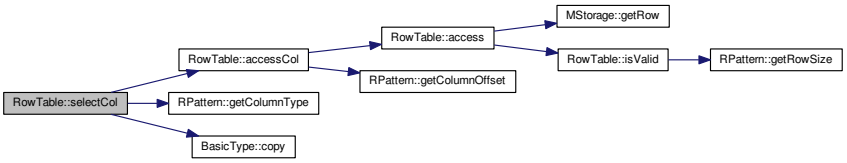
| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_rank* | the n th column in table pattern |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




Here is the caller graph for this function:


#### 4.28.3.20 bool RowTable::selectCol ( char * *row_pointer*, int64_t *column_rank*, char * *dest* ) [virtual]

select one column data by pointer of a row.

**Parameters**

| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_rank* | the n th column in table pattern |
| *dest* | buffer to store result |

**Return values**

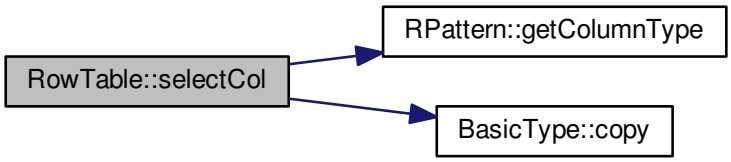
| | |
|---|---|
| *true* | success |

**Return values**

| | |
|---|---|
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.21 bool RowTable::selectCols ( int64_t *record_rank,* int64_t *column_total,* int64_t * *column_ranks,* char * *dest* ) [virtual]

select several column data.

**Parameters**

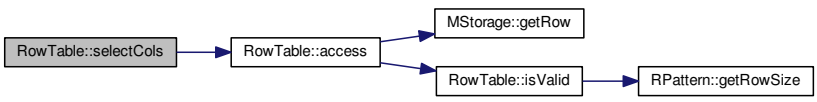
| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:



#### 4.28.3.22 bool RowTable::selectCols ( char * *row_pointer,* int64_t *column_total,* int64_t * *column_ranks,* char * *dest* ) [virtual]

select several column data.

**Parameters**

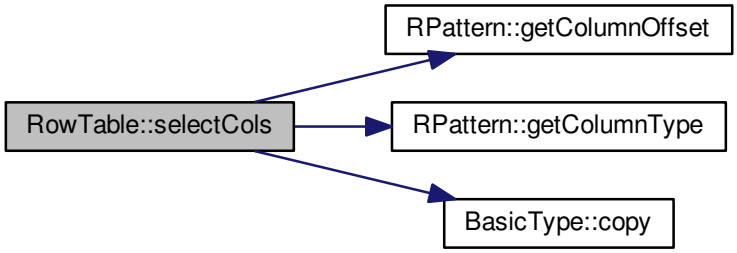
| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:

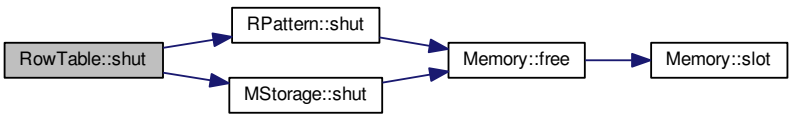



#### 4.28.3.23 bool RowTable::shut ( void ) [virtual]

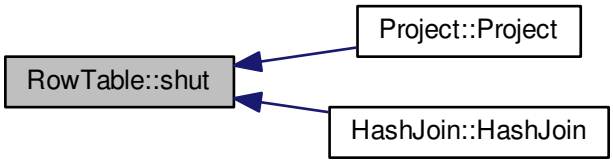
shut down r_pattern and r_storage, free their memory.

Reimplemented from Table.

Here is the call graph for this function:



Here is the caller graph for this function:




#### 4.28.3.24 bool RowTable::updateCol ( char * *row_pointer,* int64_t *column_rank,* char * *source* ) [virtual]

update a column data.

**Parameters**

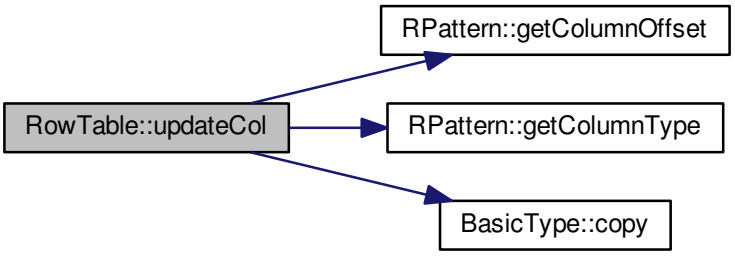
| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_rank* | the n th column in table pattern |
| *source* | buffer to store data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:



Here is the caller graph for this function:


#### 4.28.3.25 bool RowTable::updateCol ( int64_t *record_rank*, int64_t *column_rank*, char * *source* ) [virtual]

update a column data.

**Parameters**

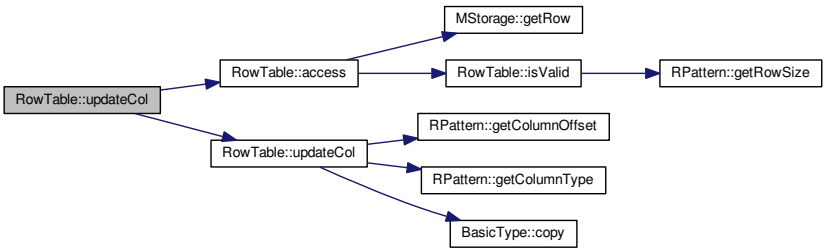
| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_rank* | the n th column in table pattern |
| *source* | buffer to store data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.26 bool RowTable::updateCols ( int64_t *record_rank*, int64_t *column_total*, int64_t * *column_ranks*, char * *source* ) [virtual]

update several column data.

**Parameters**

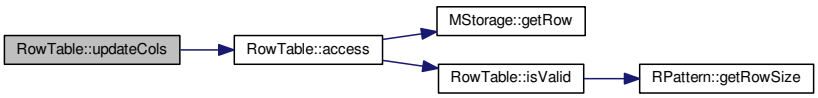
| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *source* | buffer to store data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




Here is the caller graph for this function:


#### 4.28.3.27 bool RowTable::updateCols ( char * *row_pointer,* int64_t *column_total,* int64_t * *column_ranks,* char * *source* ) [virtual]

update several column data.

**Parameters**

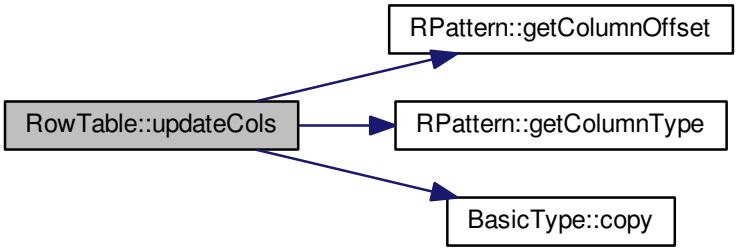
| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *source* | buffer to store data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




#### 4.28.3.28 bool RowTable::updateCols ( int64_t *record_rank*, int64_t *column_total*, int64_t * *column_ranks*, char * *source[ ]* ) [virtual]

update several column data.

**Parameters**

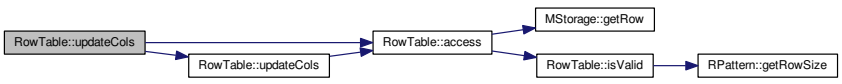
| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *source* | array of columns' pointers, each points a column data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:



#### 4.28.3.29 bool RowTable::updateCols ( char * *row_pointer,* int64_t *column_total,* int64_t * *column_ranks,* char * *source[ ]* ) [virtual]

update several column data.

**Parameters**

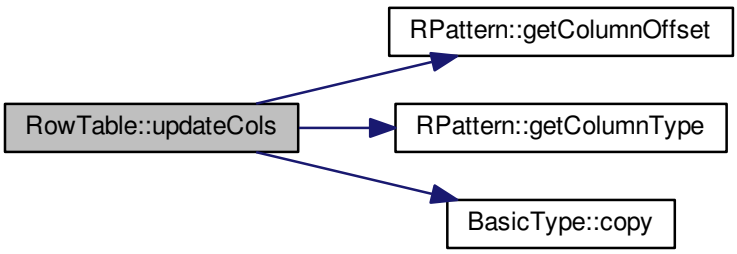
| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *source* | array of columns' pointers, each points a column data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented from Table.

Here is the call graph for this function:




### 4.28.4 Member Data Documentation

#### 4.28.4.1 RPattern RowTable::r_pattern [private]

pattern of row

#### 4.28.4.2 MStorage RowTable::r_storage [private]

storage of table

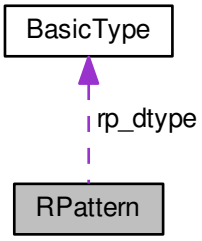
The documentation for this class was generated from the following files:

- system/rowtable.h
- system/rowtable.cc

## 4.29 RPattern Class Reference

`#include <rowtable.h>`

Collaboration diagram for RPattern:




### Public Member Functions

- bool addColumn (BasicType ∗col_type)
- int64_t getColumnOffset (int64_t col_rank)
- BasicType ∗ getColumnType (int64_t col_rank)
- int64_t getRowSize (void)
- bool init (int64_t col_num)
- int64_t print (char ∗r_ptr)
- void reset (void)
- void shut (void)

### Private Attributes

- char par [128-3 ∗sizeof(void ∗)-4 ∗sizeof(int64_t)]
- int64_t rp_colnum
- int64_t rp_current
- BasicType ∗∗ rp_dtype
- int64_t rp_mem_sz
- char ∗ rp_memory
- int64_t ∗ rp_offset
- int64_t rp_row_sz

### 4.29.1 Detailed Description

definition of class RPattern, describe row struture.

### 4.29.2 Member Function Documentation

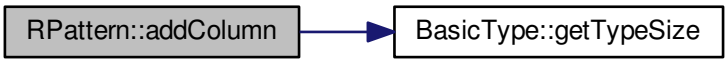
#### 4.29.2.1 bool RPattern::addColumn ( BasicType ∗ *col_type* ) [inline]

add column infomation.

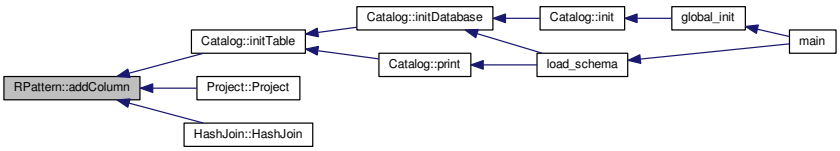
**Parameters**

| | |
|---|---|
| *col_type* | data type of the column |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.29.2.2 int64_t RPattern::getColumnOffset ( int64_t *col_rank* ) [inline]

get offset of column in a row record.

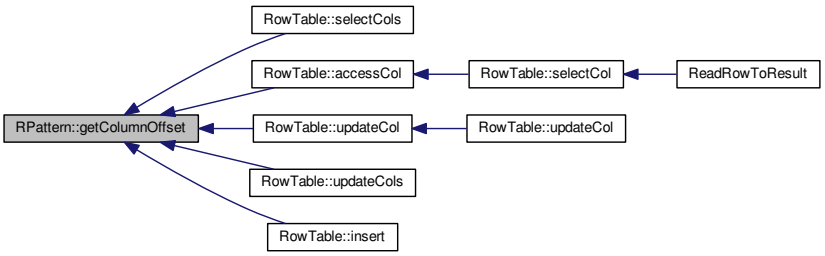
**Parameters**

| | |
|---|---|
| *col_rank* | the n th column in the table |

**Return values**

| | |
|---|---|
| *>=0* | valid offset |
| *==-1* | input error |

Here is the caller graph for this function:




#### 4.29.2.3 BasicType* RPattern::getColumnType ( int64_t *col_rank* ) [inline]

get data type of column.

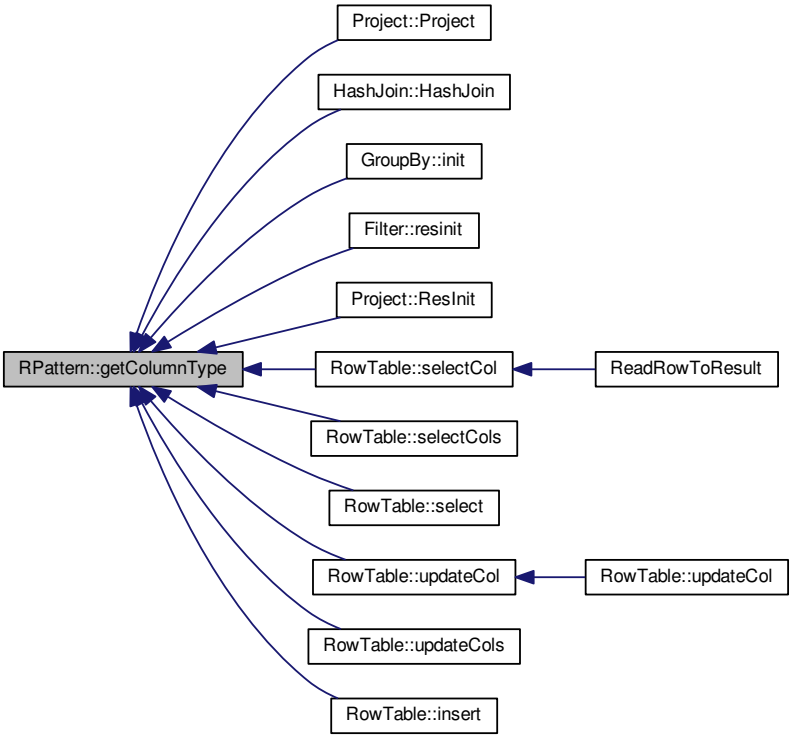
**Parameters**

| | |
|---|---|
| *col_rank* | the n th column in the table |

**Return values**

| | |
|---|---|
| *!=* | NULL valid pointer |
| *==* | NULL input error |

Here is the caller graph for this function:



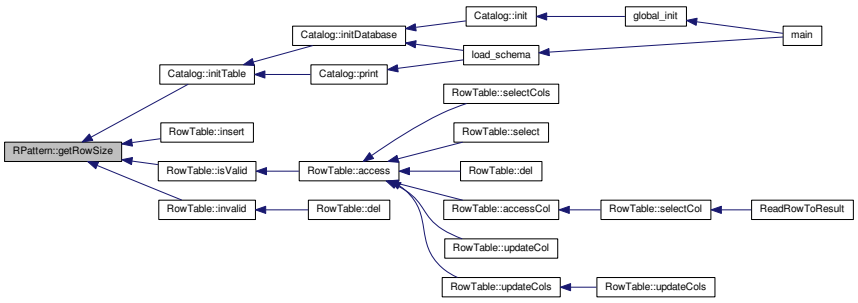

#### 4.29.2.4 int64_t RPattern::getRowSize ( void ) [inline]

get size of a row record.

**Return values**

| | |
|---|---|
| *the* | size of a row record |

Here is the caller graph for this function:



#### 4.29.2.5 bool RPattern::init ( int64_t *col_num* ) [inline]

init, alloc memory and initial setting.

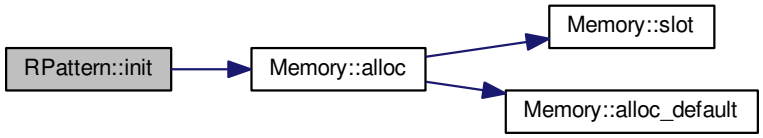
**Parameters**

| | |
|---|---|
| *col_num* | number of columns, from class Table |

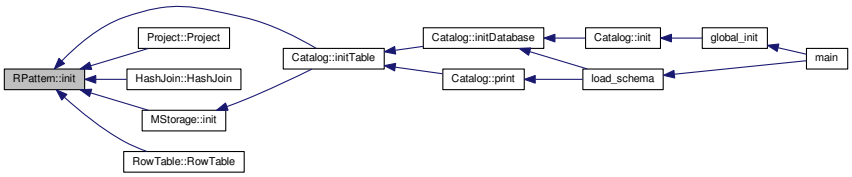
**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Here is the call graph for this function:




Here is the caller graph for this function:




#### 4.29.2.6 int64_t RPattern::print ( char * *r_ptr* ) [inline]

print a row following this pattern.

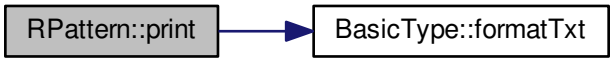
**Parameters**

| | |
|---|---|
| *r_ptr* | pointer of a row |

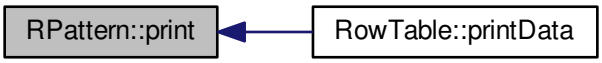
**Return values**

| | |
|---|---|
| *==rp_row_size* | success |
| *!=rp_row_size* | error |

Here is the call graph for this function:




Here is the caller graph for this function:



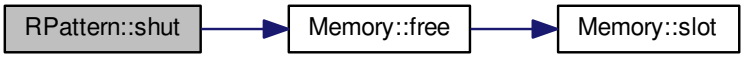

#### 4.29.2.7 void RPattern::reset ( void ) [inline]

reset if addcolumn error happen.

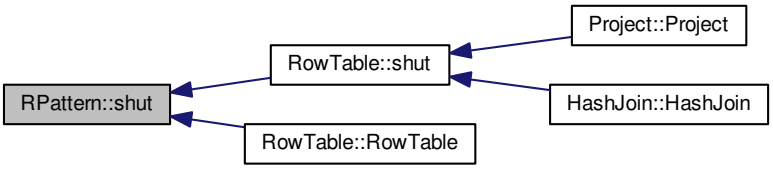
#### 4.29.2.8 void RPattern::shut ( void ) [inline]

shut down, free memory allocated from g_memory.

Here is the call graph for this function:



Here is the caller graph for this function:




### 4.29.3 Member Data Documentation

#### 4.29.3.1 char RPattern::par[128-3 ∗sizeof(void ∗)-4 ∗sizeof(int64_t)] [private]

make the sizeof RPattern 128, managed by g_memory

#### 4.29.3.2 int64_t RPattern::rp_colnum [private]

total columns

#### 4.29.3.3 int64_t RPattern::rp_current [private]

currrent columns already set offset and datatype

#### 4.29.3.4 BasicType∗∗ RPattern::rp_dtype [private]

array of pointers to each column's data type

#### 4.29.3.5 int64_t RPattern::rp_mem_sz [private]

memory size

#### 4.29.3.6 char∗ RPattern::rp_memory [private]

memory to store rp_offfset and rp_datatype

#### 4.29.3.7 int64_t∗ RPattern::rp_offset [private]

offset of column in a row

#### 4.29.3.8 int64_t RPattern::rp_row_sz [private]

size of a row record

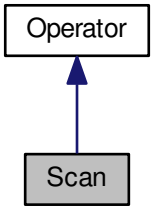
The documentation for this class was generated from the following file:

- system/rowtable.h

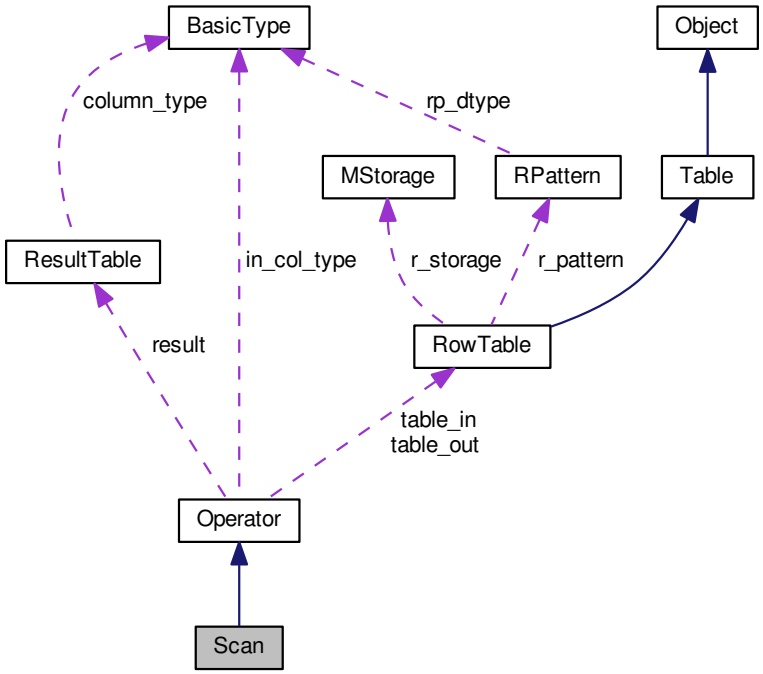
## 4.30 Scan Class Reference

`#include <executor.h>`

Inheritance diagram for Scan:




Collaboration diagram for Scan:



### Public Member Functions

- Scan (char *tablename)
- bool close ()
- bool equalornot ()
- bool get_Next (ResultTable *result)
- bool init ()
- bool is_End ()
- bool writeRow (char *buffer, ResultTable *result)

### Private Attributes

- int64_t col_num
- int64_t current_row

### Additional Inherited Members

## 4.30.1 Detailed Description

definition of Scan operator.

## 4.30.2 Constructor & Destructor Documentation

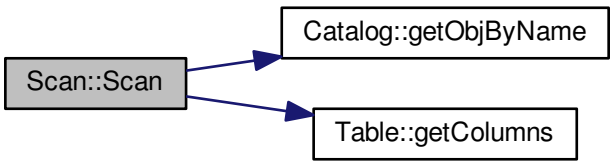
### 4.30.2.1 Scan::Scan ( char * *tablename* )

Scan rowtable from table_in

**Parameters**

| | |
|---|---|
| *tablename* | the table_in that this function will scan |

Here is the call graph for this function:




## 4.30.3 Member Function Documentation

#### 4.30.3.1 bool Scan::close ( void ) [virtual]

close scan and release memory

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

#### 4.30.3.2 bool Scan::equalornot ( ) [inline]

#### 4.30.3.3 bool Scan::get_Next ( ResultTable * *result* ) [virtual]

get next record of scan

**Parameters**

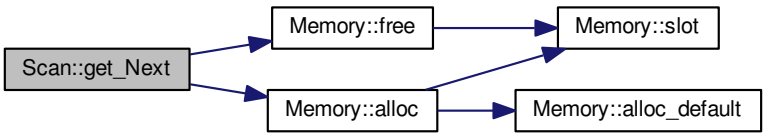
| | |
|---|---|
| *result* | the buffer to store result of scan |

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

Here is the call graph for this function:




#### 4.30.3.4 bool Scan::init ( void ) [virtual]

init scan operator

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Implements Operator.

#### 4.30.3.5 bool Scan::is_End ( void ) [virtual]

judge where scan is end

**Return values**

| | |
|---|---|
| *false* | not end |
| *true* | run end |

Implements Operator.

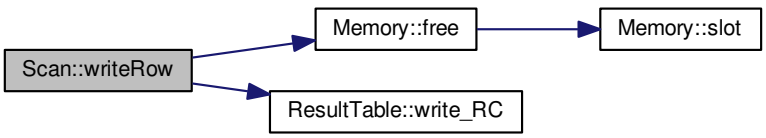
#### 4.30.3.6 bool Scan::writeRow ( char * *buffer*, ResultTable * *result* ) [inline]

write a row in resulttable

**Return values**

| | |
|---|---|
| *false* | for failure |
| *true* | for success |

Here is the call graph for this function:




### 4.30.4 Member Data Documentation

#### 4.30.4.1 int64_t Scan::col_num [private]

number of columns

#### 4.30.4.2 int64_t Scan::current_row [private]

row number has been scaned.

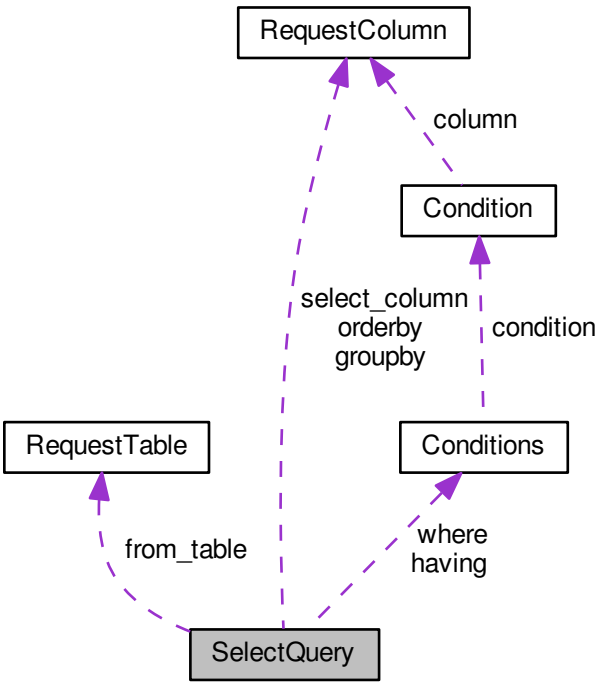
The documentation for this class was generated from the following files:

- system/executor.h
- system/executor.cc

## 4.31 SelectQuery Class Reference

`#include <executor.h>`

Collaboration diagram for SelectQuery:




### Public Attributes

- int64_t database_id
- int from_number
- RequestTable from_table [4]
- RequestColumn groupby [4]
- int groupby_number
- Conditions having
- RequestColumn orderby [4]
- int orderby_number
- RequestColumn select_column [4]
- int select_number
- Conditions where

### 4.31.1 Detailed Description

definition of selectquery.

### 4.31.2 Member Data Documentation

#### 4.31.2.1 int64_t SelectQuery::database_id

database to execute

#### 4.31.2.2 int SelectQuery::from_number

number of tables to select from

#### 4.31.2.3 RequestTable SelectQuery::from_table[4]

tables to select from, maximum 4

#### 4.31.2.4 RequestColumn SelectQuery::groupby[4]

columns to groupby

#### 4.31.2.5 int SelectQuery::groupby_number

number of columns to groupby

#### 4.31.2.6 Conditions SelectQuery::having

groupby conditions

#### 4.31.2.7 RequestColumn SelectQuery::orderby[4]

columns to orderby

#### 4.31.2.8 int SelectQuery::orderby_number

number of columns to orderby

#### 4.31.2.9 RequestColumn SelectQuery::select_column[4]

columns to select, maximum 4

#### 4.31.2.10 int SelectQuery::select_number

number of column to select

#### 4.31.2.11 Conditions SelectQuery::where

where meets conditions, maximum 4 & conditions

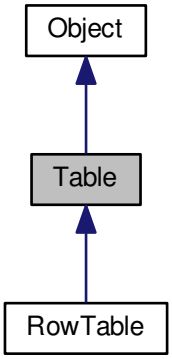
The documentation for this class was generated from the following file:

- system/executor.h

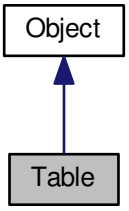
## 4.32 Table Class Reference

`#include <schema.h>`

Inheritance diagram for Table:




Collaboration diagram for Table:



### Public Member Functions

- Table (int64_t t_id, const char *t_name, TableType t_type)
- virtual ~Table (void)
- virtual bool addColumn (int64_t column_id)
- virtual bool addIndex (int64_t index_id)
- virtual bool del (int64_t record_rank)
- virtual bool del (char *row_pointer)
- virtual bool del (char *columns[])
- virtual bool finish (void)
- int64_t getColumnRank (int64_t c_id)
- std::vector< int64_t > & getColumns (void)
- int64_t getIndexRank (int64_t i_id)
- std::vector< int64_t > & getIndexs (void)
- int64_t getRank (std::vector< int64_t > &vec, int64_t id)
- virtual int64_t getRecordNum (void)
- virtual void * getRecordPtr (int64_t row_rank)
- TableType getTtype (void)
- virtual bool init (void)
- virtual bool insert (char *source)
- virtual bool insert (char *columns[])
- virtual bool loadData (const char *filename)
- virtual void print (void)
- virtual bool printData (void)
- virtual bool select (int64_t record_rank, char *dest)
- virtual bool select (char *row_pointer, char *dest)
- virtual bool selectCol (int64_t record_rank, int64_t column_rank, char *dest)
- virtual bool selectCol (char *row_pointer, int64_t column_rank, char *dest)
- virtual bool selectCols (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *dest)
- virtual bool selectCols (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *dest)
- virtual bool shut (void)
- virtual bool updateCol (int64_t record_rank, int64_t column_rank, char *source)
- virtual bool updateCol (char *row_pointer, int64_t column_rank, char *source)
- virtual bool updateCols (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *source)
- virtual bool updateCols (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *source)
- virtual bool updateCols (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *source[])
- virtual bool updateCols (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *source[])

### Private Attributes

- std::vector< int64_t > t_columns
- std::vector< int64_t > t_index
- TableType t_type

## 4.32.1 Detailed Description

definition of class Table.

## 4.32.2 Constructor & Destructor Documentation

#### 4.32.2.1 virtual Table::~Table ( void ) [inline], [virtual]

destructor.

#### 4.32.2.2 Table::Table ( int64_t *t_id,* const char * *t_name,* TableType *t_type* ) [inline]

constructor.

**Parameters**

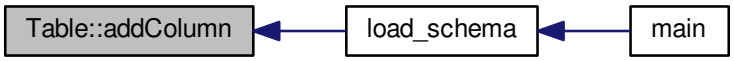
| | |
|---|---|
| *t_id* | table identifier |
| *t_name* | table name |
| *t_type* | table type |

### 4.32.3 Member Function Documentation

#### 4.32.3.1 virtual bool Table::addColumn ( int64_t *column_id* ) [inline], [virtual]

add column identifier to this table.

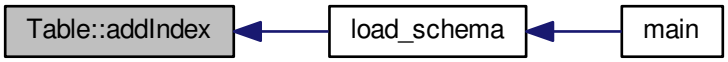
Here is the caller graph for this function:




#### 4.32.3.2 virtual bool Table::addIndex ( int64_t *index_id* ) [inline], [virtual]

add index identifier to this table.

Here is the caller graph for this function:




#### 4.32.3.3 virtual bool Table::del ( int64_t *record_rank* ) [inline], [virtual]

del a row.

**Parameters**

| | |
|---|---|
| *row_rank* | the n th record of the table |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.4 virtual bool Table::del ( char * *row_pointer* ) [inline], [virtual]

del a row(not in use).

**Parameters**

| | |
|---|---|
| *row_pointer* | the pointer of a row |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.5 virtual bool Table::del ( char * *columns[ ]* ) [inline], [virtual]

del a row(not in use).

**Parameters**

| | |
|---|---|
| *columns* | array of the pointers in a row |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

#### 4.32.3.6 virtual bool Table::finish ( void ) [inline], [virtual]

finish, important interface for son class

Reimplemented in RowTable.

#### 4.32.3.7 int64_t Table::getColumnRank ( int64_t *c_id* ) [inline]

get column rank in this table

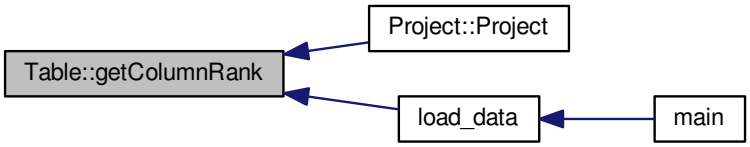
**Parameters**

| | |
|---|---|
| *c↩ _id* | column identifier |

**Return values**

| | |
|---|---|
| *>=* | 0 valid rank |
| *<0* | invalid, not exist |

Here is the caller graph for this function:



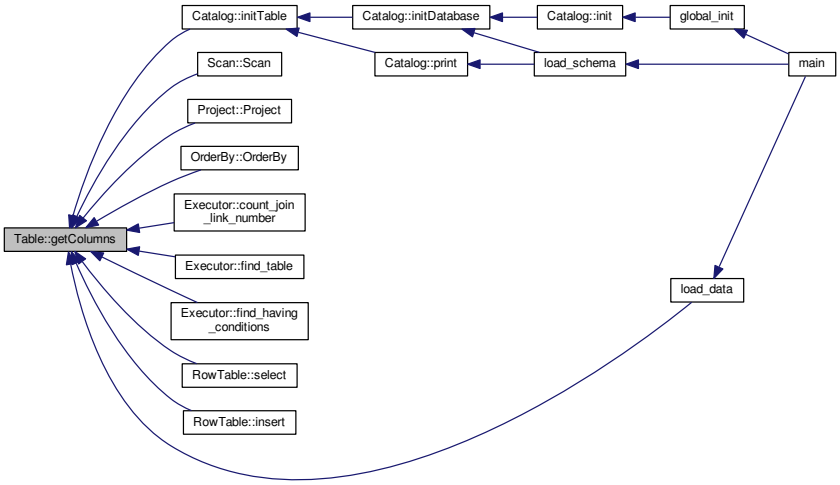

#### 4.32.3.8 std::vector< int64_t >& Table::getColumns ( void ) [inline]

get column identifier vector.

**Return values**

| | |
|---|---|
| *vector* | of column identifiers |

Here is the caller graph for this function:




#### 4.32.3.9 int64_t Table::getIndexRank ( int64_t *i_id* ) [inline]

get index rank in this table

**Parameters**

| | |
|---|---|
| *i_id* | column identifier |

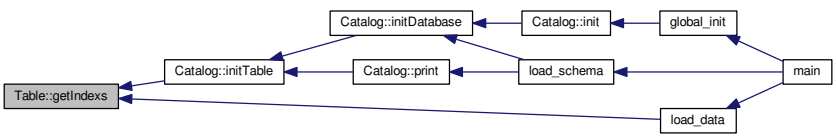
**Return values**

| | |
|---|---|
| *>=* | 0 valid rank |
| *<0* | invalid, not exist |

#### 4.32.3.10 std::vector< int64_t >& Table::getIndexs ( void ) [inline]

get index identifier vector.

Here is the caller graph for this function:



#### 4.32.3.11 int64_t Table::getRank ( std::vector< int64_t > & *vec,* int64_t *id* ) [inline]

get rank in a vector

**Parameters**

| | |
|---|---|
| *vec* | vector to search in |
| *id* | object identifier |

**Return values**

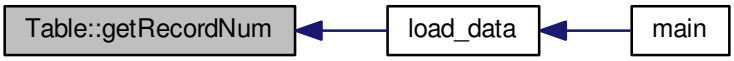
| | |
|---|---|
| >= | 0 valid rank |
| <0 | invalid, not exist |

#### 4.32.3.12 virtual int64_t Table::getRecordNum ( void ) [inline], [virtual]

get record number.

Reimplemented in RowTable.

Here is the caller graph for this function:

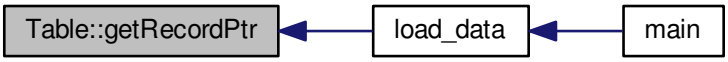



#### 4.32.3.13 virtual void* Table::getRecordPtr ( int64_t *row_rank* ) [inline], [virtual]

get record pointer.

Reimplemented in RowTable.

Here is the caller graph for this function:



#### 4.32.3.14 TableType Table::getTtype ( void ) [inline]

get table type.

Here is the caller graph for this function:


#### 4.32.3.15 virtual bool Table::init ( void ) [inline], [virtual]

init, important interface for son class

Reimplemented in RowTable.

#### 4.32.3.16 virtual bool Table::insert ( char * *source* ) [inline], [virtual]

insert a row.

**Parameters**

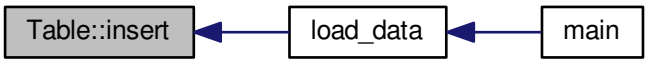
| | |
|---|---|
| *source* | buffer of a row in pattern |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

Here is the caller graph for this function:




#### 4.32.3.17 virtual bool Table::insert ( char * *columns[]* ) [inline], [virtual]

insert a row.

**Parameters**

| | |
|---|---|
| *columns* | each element of the array pointed to a column data |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.18 virtual bool Table::loadData ( const char * *filename* ) [inline], [virtual]

load data(not in use).

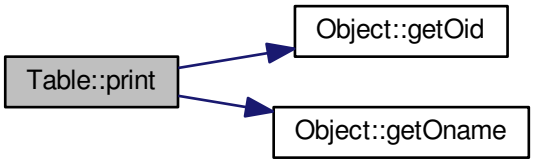
Reimplemented in RowTable.

#### 4.32.3.19 virtual void Table::print ( void ) [inline], [virtual]

print table information.

Reimplemented from Object.

Here is the call graph for this function:

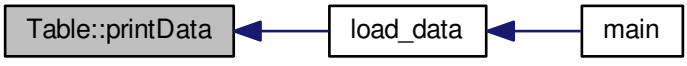



#### 4.32.3.20 virtual bool Table::printData ( void ) [inline], [virtual]

print data in table.

Reimplemented in RowTable.

Here is the caller graph for this function:



#### 4.32.3.21 virtual bool Table::select ( int64_t *record_rank*, char * *dest* ) [inline], [virtual]

select all columns' data.

**Parameters**

| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.22 virtual bool Table::select ( char * *row_pointer*, char * *dest* ) [inline], [virtual]

select all columns' data.

**Parameters**

| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.23 virtual bool Table::selectCol ( int64_t *record_rank*, int64_t *column_rank*, char * *dest* ) [inline], [virtual]

select one column data.

**Parameters**

| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_rank* | the n th column in table pattern |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.24 virtual bool Table::selectCol ( char * *row_pointer,* int64_t *column_rank,* char * *dest* ) [inline],[virtual]

select one column data by pointer of a row.

**Parameters**

| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_rank* | the n th column in table pattern |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.25 virtual bool Table::selectCols ( int64_t *record_rank,* int64_t *column_total,* int64_t * *column_ranks,* char * *dest* ) [inline],[virtual]

select several column data.

**Parameters**

| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.26 virtual bool Table::selectCols ( char * *row_pointer,* int64_t *column_total,* int64_t * *column_ranks,* char * *dest* ) [inline],[virtual]

select several column data.

**Parameters**

| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *dest* | buffer to store result |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.27 virtual bool Table::shut ( void ) [inline], [virtual]

shut, important interface for son class

Reimplemented from Object.

Reimplemented in RowTable.

#### 4.32.3.28 virtual bool Table::updateCol ( int64_t *record_rank*, int64_t *column_rank*, char * *source* ) [inline], [virtual]

update one column data.

**Parameters**

| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_rank* | the n th column in table pattern |
| *source* | buffer to store data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.29 virtual bool Table::updateCol ( char * *row_pointer*, int64_t *column_rank*, char * *source* ) [inline], [virtual]

update one column data.

**Parameters**

| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_rank* | the n th column in table pattern |
| *source* | buffer to store data to change for |

**Return values**

| | |
|---|---|
| *true* | success |

**Return values**

| | |
|---|---|
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.30 virtual bool Table::updateCols ( int64_t *record_rank,* int64_t *column_total,* int64_t * *column_ranks,* char * *source* ) [inline], [virtual]

Reimplemented in RowTable.

#### 4.32.3.31 virtual bool Table::updateCols ( char * *row_pointer,* int64_t *column_total,* int64_t * *column_ranks,* char * *source* ) [inline], [virtual]

update several column data.

**Parameters**

| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *source* | buffer to store data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.32 virtual bool Table::updateCols ( int64_t *record_rank,* int64_t *column_total,* int64_t * *column_ranks,* char * *source[ ]* ) [inline], [virtual]

update several column data.

**Parameters**

| | |
|---|---|
| *record_rank* | the n th row in the table storage |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *source* | array of columns' pointers, each points a column data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

#### 4.32.3.33 virtual bool Table::updateCols ( char * *row_pointer,* int64_t *column_total,* int64_t * *column_ranks,* char * *source[ ]* ) [inline], [virtual]

update several column data.

**Parameters**

| | |
|---|---|
| *row_pointer* | the pointer of a row |
| *column_total* | total number of columns to select |
| *column_ranks* | array of column_rank, column_rank is the n th column in table pattern |
| *source* | array of columns' pointers, each points a column data to change for |

**Return values**

| | |
|---|---|
| *true* | success |
| *false* | failure |

Reimplemented in RowTable.

### 4.32.4 Member Data Documentation

#### 4.32.4.1 std::vector< int64_t > Table::t_columns [private]

vector of columns' identifier

#### 4.32.4.2 std::vector< int64_t > Table::t_index [private]

vector of index's identifier

#### 4.32.4.3 TableType Table::t_type [private]

table type

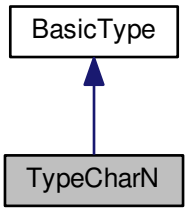
The documentation for this class was generated from the following file:

- system/schema.h

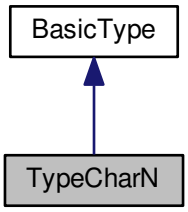
## 4.33 TypeCharN Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeCharN:




Collaboration diagram for TypeCharN:




### Public Member Functions

- TypeCharN (int64_t typesize)
- TypeCharN (TypeCode typecode=CHARN_TC, int64_t typesize=32)
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

#### 4.33.1 Detailed Description

definition of class TypeCharN,please refer to BasicType,it's same.

#### 4.33.2 Constructor & Destructor Documentation

##### 4.33.2.1 TypeCharN::TypeCharN ( int64_t *typesize* ) [inline]

constructor.

##### 4.33.2.2 TypeCharN::TypeCharN ( TypeCode *typecode* = CHARN_TC, int64_t *typesize* = 32 ) [inline]

#### 4.33.3 Member Function Documentation

##### 4.33.3.1 bool TypeCharN::cmpEQ ( void * *data1*, void * *data2* ) [inline], [virtual]

equal to.

Reimplemented from BasicType.

##### 4.33.3.2 bool TypeCharN::cmpGE ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than or equal to

Reimplemented from BasicType.

##### 4.33.3.3 bool TypeCharN::cmpGT ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than.

Reimplemented from BasicType.

##### 4.33.3.4 bool TypeCharN::cmpLE ( void * *data1*, void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

##### 4.33.3.5 bool TypeCharN::cmpLT ( void * *data1*, void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.33.3.6 int TypeCharN::copy ( void * *dest,* void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.33.3.7 int TypeCharN::formatBin ( void * *dest,* void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.33.3.8 int TypeCharN::formatTxt ( void * *dest,* void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

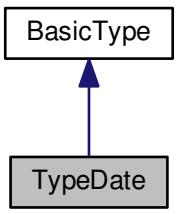
The documentation for this class was generated from the following file:

- system/datatype.h

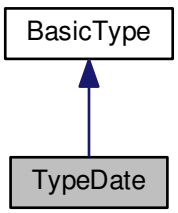
## 4.34 TypeDate Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeDate:




Collaboration diagram for TypeDate:



### Public Member Functions

- TypeDate (TypeCode typecode=DATE_TC, int64_t typesize=sizeof(time_t))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

## 4.34.1 Detailed Description

definition of class TypeDate,please refer to BasicType,it's same.

## 4.34.2 Constructor & Destructor Documentation

#### 4.34.2.1 TypeDate::TypeDate ( TypeCode *typecode* = DATE_TC, int64_t *typesize* = `sizeof(time_t)` ) `[inline]`

constructor.

## 4.34.3 Member Function Documentation

#### 4.34.3.1 bool TypeDate::cmpEQ ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

equal to.

Reimplemented from BasicType.

#### 4.34.3.2 bool TypeDate::cmpGE ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than or equal to

Reimplemented from BasicType.

#### 4.34.3.3 bool TypeDate::cmpGT ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than.

Reimplemented from BasicType.

#### 4.34.3.4 bool TypeDate::cmpLE ( void * *data1*, void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

#### 4.34.3.5 bool TypeDate::cmpLT ( void * *data1*, void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.34.3.6 int TypeDate::copy ( void * *dest*, void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.34.3.7 int TypeDate::formatBin ( void * *dest*, void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.34.3.8 int TypeDate::formatTxt ( void * *dest*, void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

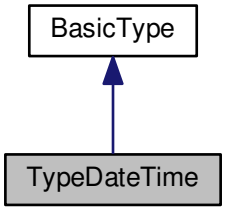
The documentation for this class was generated from the following file:

- system/datatype.h

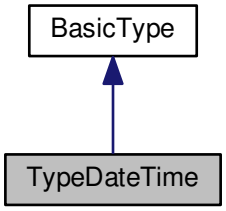
## 4.35 TypeDateTime Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeDateTime:




Collaboration diagram for TypeDateTime:




### Public Member Functions

- TypeDateTime (TypeCode typecode=DATETIME_TC, int64_t typesize=sizeof(time_t))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

### 4.35.1 Detailed Description

definition of class TypeDateTime,please refer to BasicType,it's same.

### 4.35.2 Constructor & Destructor Documentation

#### 4.35.2.1 TypeDateTime::TypeDateTime ( TypeCode *typecode* = DATETIME_TC, int64_t *typesize* = sizeof(time_t) ) [inline]

constructor.

### 4.35.3 Member Function Documentation

#### 4.35.3.1 bool TypeDateTime::cmpEQ ( void * *data1*, void * *data2* ) [inline], [virtual]

equal to.

Reimplemented from BasicType.

#### 4.35.3.2 bool TypeDateTime::cmpGE ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than or equal to

Reimplemented from BasicType.

#### 4.35.3.3 bool TypeDateTime::cmpGT ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than.

Reimplemented from BasicType.

#### 4.35.3.4 bool TypeDateTime::cmpLE ( void * *data1*, void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

#### 4.35.3.5 bool TypeDateTime::cmpLT ( void * *data1*, void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.35.3.6 int TypeDateTime::copy ( void * *dest*, void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.35.3.7 int TypeDateTime::formatBin ( void * *dest*, void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.35.3.8 int TypeDateTime::formatTxt ( void * *dest*, void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

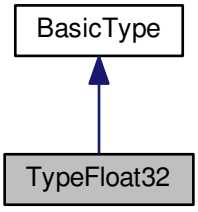
The documentation for this class was generated from the following file:

- system/datatype.h

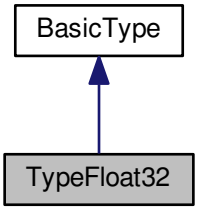
## 4.36 TypeFloat32 Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeFloat32:




Collaboration diagram for TypeFloat32:



### Public Member Functions

- TypeFloat32 (TypeCode typecode=FLOAT32_TC, int64_t typesize=sizeof(float))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

## 4.36.1 Detailed Description

definition of class TypeFloat32,please refer to BasicType,it's same.

## 4.36.2 Constructor & Destructor Documentation

#### 4.36.2.1 TypeFloat32::TypeFloat32 ( TypeCode *typecode* = FLOAT32_TC, int64_t *typesize* = `sizeof(float)` ) `[inline]`

constructor.

## 4.36.3 Member Function Documentation

#### 4.36.3.1 bool TypeFloat32::cmpEQ ( void * *data1*, void * *data2* ) `[inline],[virtual]`

equal to.

Reimplemented from BasicType.

#### 4.36.3.2 bool TypeFloat32::cmpGE ( void * *data1*, void * *data2* ) `[inline],[virtual]`

greater than or equal to

Reimplemented from BasicType.

#### 4.36.3.3 bool TypeFloat32::cmpGT ( void * *data1*, void * *data2* ) `[inline],[virtual]`

greater than.

Reimplemented from BasicType.

#### 4.36.3.4 bool TypeFloat32::cmpLE ( void * *data1*, void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

#### 4.36.3.5 bool TypeFloat32::cmpLT ( void * *data1*, void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.36.3.6 int TypeFloat32::copy ( void * *dest*, void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.36.3.7 int TypeFloat32::formatBin ( void * *dest*, void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.36.3.8 int TypeFloat32::formatTxt ( void * *dest*, void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

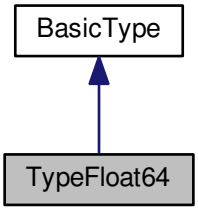
The documentation for this class was generated from the following file:

- system/datatype.h

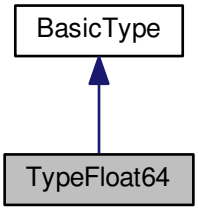
## 4.37 TypeFloat64 Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeFloat64:




Collaboration diagram for TypeFloat64:




### Public Member Functions

- TypeFloat64 (TypeCode typecode=FLOAT64_TC, int64_t typesize=sizeof(double))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

### 4.37.1 Detailed Description

definition of class TypeFloat64,please refer to BasicType,it's same.

### 4.37.2 Constructor & Destructor Documentation

#### 4.37.2.1 TypeFloat64::TypeFloat64 ( TypeCode *typecode* = FLOAT64_TC, int64_t *typesize* = `sizeof(double)` ) `[inline]`

constructor.

### 4.37.3 Member Function Documentation

#### 4.37.3.1 bool TypeFloat64::cmpEQ ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

equal to.

Reimplemented from BasicType.

#### 4.37.3.2 bool TypeFloat64::cmpGE ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than or equal to

Reimplemented from BasicType.

#### 4.37.3.3 bool TypeFloat64::cmpGT ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than.

Reimplemented from BasicType.

#### 4.37.3.4 bool TypeFloat64::cmpLE ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

less than or equal to.

Reimplemented from BasicType.

#### 4.37.3.5 bool TypeFloat64::cmpLT ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

less than.

Reimplemented from BasicType.

#### 4.37.3.6 int TypeFloat64::copy ( void * *dest*, void * *data* ) `[inline]`, `[virtual]`

copy from data to dest.

Reimplemented from BasicType.

#### 4.37.3.7 int TypeFloat64::formatBin ( void * *dest*, void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.37.3.8 int TypeFloat64::formatTxt ( void * *dest*, void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

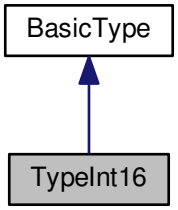
The documentation for this class was generated from the following file:

- system/datatype.h

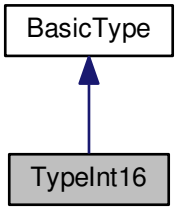
## 4.38 TypeInt16 Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeInt16:




Collaboration diagram for TypeInt16:



### Public Member Functions

- TypeInt16 (TypeCode typecode=INT16_TC, int64_t typesize=sizeof(int16_t))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

## 4.38.1 Detailed Description

definition of class TypeInt16,please refer to BasicType,it's same.

## 4.38.2 Constructor & Destructor Documentation

#### 4.38.2.1 TypeInt16::TypeInt16 ( TypeCode *typecode* = INT16_TC, int64_t *typesize* = `sizeof(int16_t)` ) `[inline]`

constructor.

## 4.38.3 Member Function Documentation

#### 4.38.3.1 bool TypeInt16::cmpEQ ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

equal to.

Reimplemented from BasicType.

#### 4.38.3.2 bool TypeInt16::cmpGE ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than or equal to

Reimplemented from BasicType.

#### 4.38.3.3 bool TypeInt16::cmpGT ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than.

Reimplemented from BasicType.

#### 4.38.3.4 bool TypeInt16::cmpLE ( void * *data1,* void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

#### 4.38.3.5 bool TypeInt16::cmpLT ( void * *data1,* void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.38.3.6 int TypeInt16::copy ( void * *dest,* void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.38.3.7 int TypeInt16::formatBin ( void * *dest,* void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.38.3.8 int TypeInt16::formatTxt ( void * *dest,* void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

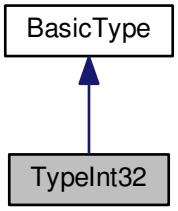
The documentation for this class was generated from the following file:

- system/datatype.h

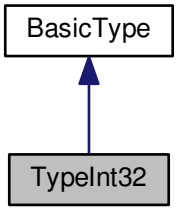
## 4.39 TypeInt32 Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeInt32:




Collaboration diagram for TypeInt32:




### Public Member Functions

- TypeInt32 (TypeCode typecode=INT32_TC, int64_t typesize=sizeof(int32_t))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

### 4.39.1 Detailed Description

definition of class TypeInt32,please refer to BasicType,it's same.

### 4.39.2 Constructor & Destructor Documentation

#### 4.39.2.1 TypeInt32::TypeInt32 ( TypeCode *typecode* = INT32_TC, int64_t *typesize* = sizeof(int32_t) ) [inline]

constructor.

### 4.39.3 Member Function Documentation

#### 4.39.3.1 bool TypeInt32::cmpEQ ( void * *data1*, void * *data2* ) [inline], [virtual]

equal to.

Reimplemented from BasicType.

#### 4.39.3.2 bool TypeInt32::cmpGE ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than or equal to

Reimplemented from BasicType.

#### 4.39.3.3 bool TypeInt32::cmpGT ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than.

Reimplemented from BasicType.

#### 4.39.3.4 bool TypeInt32::cmpLE ( void * *data1*, void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

#### 4.39.3.5 bool TypeInt32::cmpLT ( void * *data1*, void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.39.3.6 int TypeInt32::copy ( void * *dest*, void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.39.3.7 int TypeInt32::formatBin ( void * *dest*, void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.39.3.8 int TypeInt32::formatTxt ( void * *dest*, void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

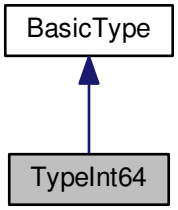
The documentation for this class was generated from the following file:

- system/datatype.h

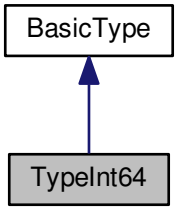
## 4.40 TypeInt64 Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeInt64:




Collaboration diagram for TypeInt64:



### Public Member Functions

- TypeInt64 (TypeCode typecode=INT64_TC, int64_t typesize=sizeof(int64_t))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

## 4.40.1 Detailed Description

definition of class TypeInt64,please refer to BasicType,it's same.

## 4.40.2 Constructor & Destructor Documentation

#### 4.40.2.1 TypeInt64::TypeInt64 ( TypeCode *typecode* = INT64_TC, int64_t *typesize* = `sizeof(int64_t)` ) `[inline]`

constructor.

## 4.40.3 Member Function Documentation

#### 4.40.3.1 bool TypeInt64::cmpEQ ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

equal to.

Reimplemented from BasicType.

#### 4.40.3.2 bool TypeInt64::cmpGE ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than or equal to

Reimplemented from BasicType.

#### 4.40.3.3 bool TypeInt64::cmpGT ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than.

Reimplemented from BasicType.

#### 4.40.3.4 bool TypeInt64::cmpLE ( void * *data1,* void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

#### 4.40.3.5 bool TypeInt64::cmpLT ( void * *data1,* void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.40.3.6 int TypeInt64::copy ( void * *dest,* void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.40.3.7 int TypeInt64::formatBin ( void * *dest,* void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.40.3.8 int TypeInt64::formatTxt ( void * *dest,* void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

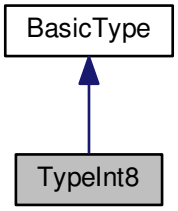
The documentation for this class was generated from the following file:

- system/datatype.h

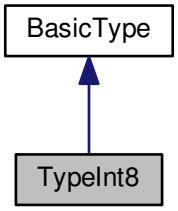
## 4.41 TypeInt8 Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeInt8:




Collaboration diagram for TypeInt8:




### Public Member Functions

- TypeInt8 (TypeCode typecode=INT8_TC, int64_t typesize=sizeof(int8_t))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

### 4.41.1 Detailed Description

definition of class TypeInt8,please refer to BasicType,it's same.

### 4.41.2 Constructor & Destructor Documentation

#### 4.41.2.1 TypeInt8::TypeInt8 ( TypeCode *typecode* = INT8_TC, int64_t *typesize* = sizeof(int8_t) ) [inline]

constructor.

### 4.41.3 Member Function Documentation

#### 4.41.3.1 bool TypeInt8::cmpEQ ( void * *data1*, void * *data2* ) [inline], [virtual]

equal to.

Reimplemented from BasicType.

#### 4.41.3.2 bool TypeInt8::cmpGE ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than or equal to

Reimplemented from BasicType.

#### 4.41.3.3 bool TypeInt8::cmpGT ( void * *data1*, void * *data2* ) [inline], [virtual]

greater than.

Reimplemented from BasicType.

#### 4.41.3.4 bool TypeInt8::cmpLE ( void * *data1*, void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

#### 4.41.3.5 bool TypeInt8::cmpLT ( void * *data1*, void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.41.3.6 int TypeInt8::copy ( void * *dest*, void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.41.3.7 int TypeInt8::formatBin ( void * *dest*, void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.41.3.8 int TypeInt8::formatTxt ( void * *dest*, void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

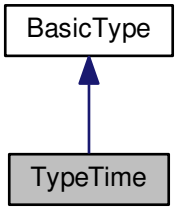
The documentation for this class was generated from the following file:

- system/datatype.h

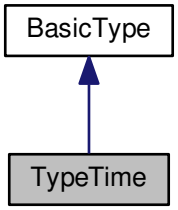
## 4.42 TypeTime Class Reference

`#include <datatype.h>`

Inheritance diagram for TypeTime:




Collaboration diagram for TypeTime:



### Public Member Functions

- TypeTime (TypeCode typecode=TIME_TC, int64_t typesize=sizeof(time_t))
- bool cmpEQ (void *data1, void *data2)
- bool cmpGE (void *data1, void *data2)
- bool cmpGT (void *data1, void *data2)
- bool cmpLE (void *data1, void *data2)
- bool cmpLT (void *data1, void *data2)
- int copy (void *dest, void *data)
- int formatBin (void *dest, void *data)
- int formatTxt (void *dest, void *data)

### Additional Inherited Members

## 4.42.1 Detailed Description

definition of class TypeTime,please refer to BasicType,it's same.

## 4.42.2 Constructor & Destructor Documentation

#### 4.42.2.1 TypeTime::TypeTime ( TypeCode *typecode* = TIME_TC, int64_t *typesize* = `sizeof(time_t)` ) `[inline]`

constructor.

## 4.42.3 Member Function Documentation

#### 4.42.3.1 bool TypeTime::cmpEQ ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

equal to.

Reimplemented from BasicType.

#### 4.42.3.2 bool TypeTime::cmpGE ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than or equal to

Reimplemented from BasicType.

#### 4.42.3.3 bool TypeTime::cmpGT ( void * *data1*, void * *data2* ) `[inline]`, `[virtual]`

greater than.

Reimplemented from BasicType.

#### 4.42.3.4 bool TypeTime::cmpLE ( void * *data1,* void * *data2* ) [inline], [virtual]

less than or equal to.

Reimplemented from BasicType.

#### 4.42.3.5 bool TypeTime::cmpLT ( void * *data1,* void * *data2* ) [inline], [virtual]

less than.

Reimplemented from BasicType.

#### 4.42.3.6 int TypeTime::copy ( void * *dest,* void * *data* ) [inline], [virtual]

copy from data to dest.

Reimplemented from BasicType.

#### 4.42.3.7 int TypeTime::formatBin ( void * *dest,* void * *data* ) [inline], [virtual]

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.42.3.8 int TypeTime::formatTxt ( void * *dest,* void * *data* ) [inline], [virtual]

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

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

- system/datatype.h

# Chapter 5

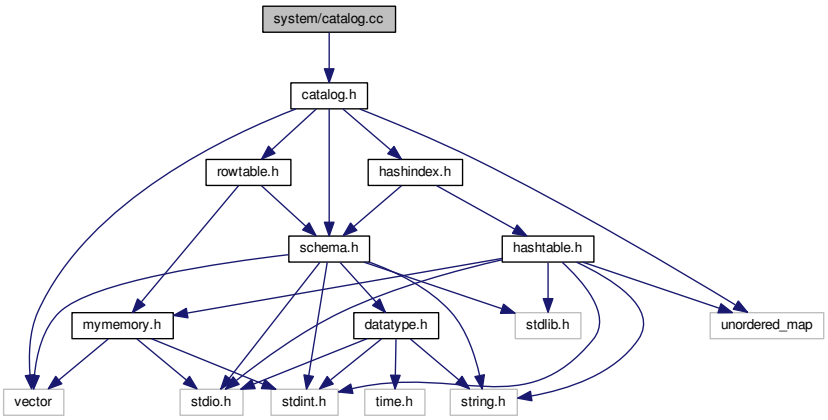
# File Documentation

## 5.1 debug/Makefile.c File Reference

## 5.2 system/catalog.cc File Reference

```
#include "catalog.h"
```

Include dependency graph for catalog.cc:




### Variables

- Catalog g_catalog

### 5.2.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

### 5.2.2 DESCRIPTION

this file provides an element container of database, which can be described as the stem of a tree with this Catalog, the only one in global system, you can access all elements of system

basic usage:

(1) you use g_catalog.createXXX[Database,Table,Column,Index] to create objects. (2) you can get the pointer of [Database,Table,Column,Index] by call g_catalog.getObjById/Name (3) in [Database,Table,Column,Index], you can define the relation of different objects by adding operation in those object (4) when all relation defined in database, you must call initDatabase to actually getting the database prepared to be used. (5) shut will get everything shutup can free shutDatabase will only shutup the selected database (6) for more tips, you may learn from debug_catalog.cc

### 5.2.3 Variable Documentation

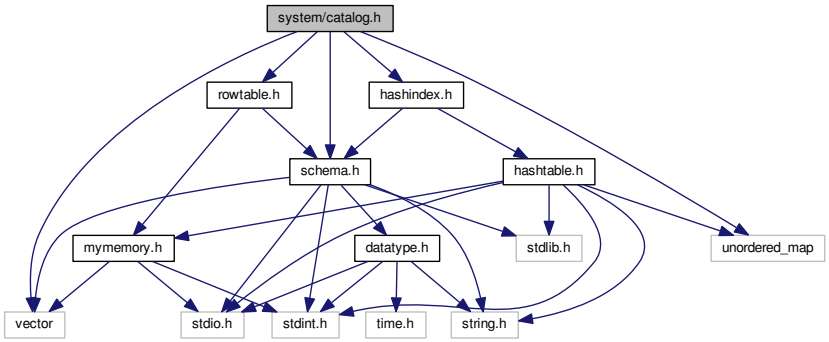
#### 5.2.3.1 Catalog g_catalog

## 5.3 system/catalog.d File Reference

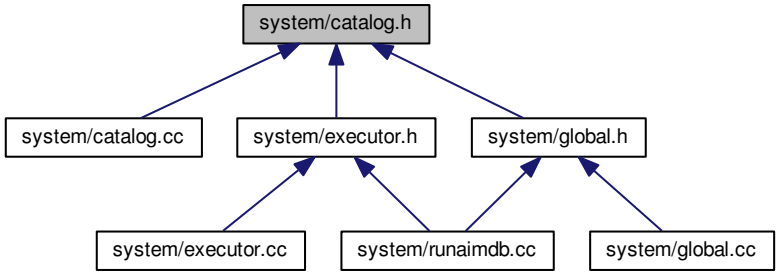
## 5.4 system/catalog.h File Reference

```
#include <vector>
#include <unordered_map>
#include "schema.h"
#include "rowtable.h"
#include "hashindex.h"
```

Include dependency graph for catalog.h:



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




## Classes

- class Catalog

## Variables

- Catalog g_catalog

### 5.4.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

### 5.4.2 DESCRIPTION

this file provides an element container of database, which can be described as the stem of a tree with this Catalog, the only one in global system, you can access all elements of system

basic usage:

(1) you use g_catalog.createXXX[Database,Table,Column,Index] to create objects. (2) you can get the pointer of [Database,Table,Column,Index] by call g_catalog.getObjById/Name (3) in [Database,Table,Column,Index], you can define the relation of different objects by adding operation in those object (4) when all relation defined in database, you must call initDatabase to actually getting the database prepared to be used. (5) shut will get everything shutup can free shutDatabase will only shutup the selected database (6) for more tips, you may learn from debug_catalog.cc

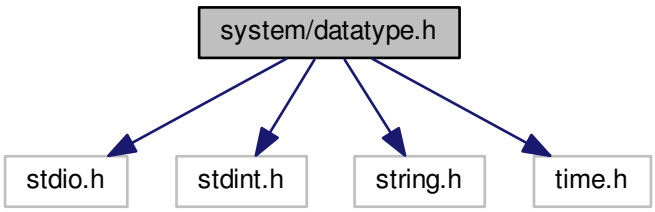
### 5.4.3 Variable Documentation

#### 5.4.3.1 Catalog g_catalog

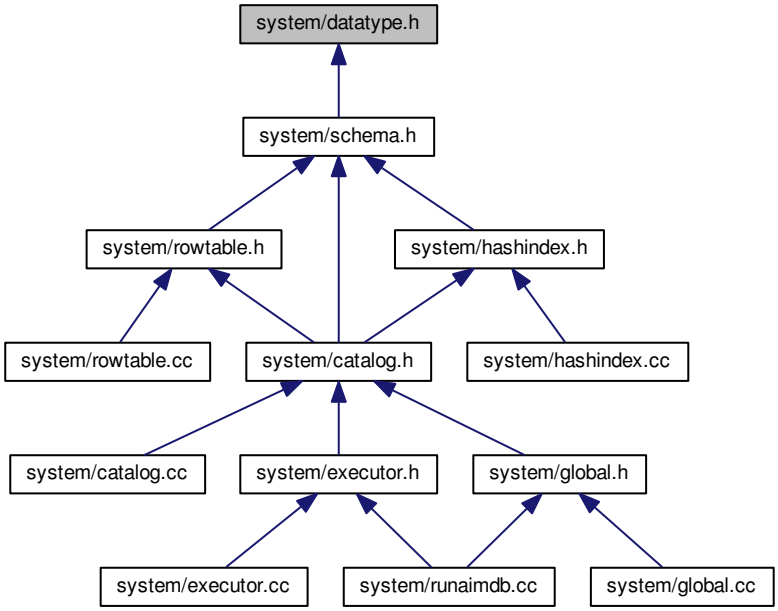
## 5.5 system/datatype.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <time.h>
```

Include dependency graph for datatype.h:




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



### Classes

- class BasicType
- class TypeCharN
- class TypeDate
- class TypeDateTime
- class TypeFloat32
- class TypeFloat64
- class TypeInt16
- class TypeInt32
- class TypeInt64
- class TypeInt8
- class TypeTime

### Enumerations

- enum TypeCode {
  INVID_TC = 0, INT8_TC, INT16_TC, INT32_TC,
  INT64_TC, FLOAT32_TC, FLOAT64_TC, CHARN_TC,
  DATE_TC, TIME_TC, DATETIME_TC, MAXTYPE_TC }

## 5.5.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

## 5.5.2 DESCRIPTION

all datatype supported by this system

## 5.5.3 Enumeration Type Documentation

### 5.5.3.1 enum TypeCode

data type code.

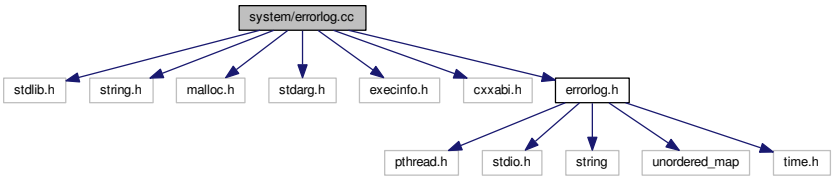
**Enumerator**

***INVID_TC***
***INT8_TC*** int8
***INT16_TC*** int16
***INT32_TC*** int32
***INT64_TC*** int64
***FLOAT32_TC*** float32
***FLOAT64_TC*** float64
***CHARN_TC*** charn
***DATE_TC*** days from 1970-01-01 till current DATE
***TIME_TC*** seconds from 00:00:00 till current TIME
***DATETIME_TC*** seconds from 1970-01-01 00:00:00 till current DATETIME
***MAXTYPE_TC***

## 5.6 system/errorlog.cc File Reference

```
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <stdarg.h>
#include <execinfo.h>
#include <cxxabi.h>
#include "errorlog.h"
```

Include dependency graph for errorlog.cc:




### Macros

- #define EL_TOTAL_FILES (sizeof(EL_src_file_name)/sizeof(char∗))

### Variables

- const char ∗ EL_src_file_name [ ]
- ErrorLog _Thread_local ∗ thread_el = NULL

### 5.6.1 Detailed Description

**Author**

Shimin Chen chensm@ict.ac.cn

**Version**

0.1

### 5.6.2 Description

This file provides the error handling and logging utility.

### 5.6.3 Macro Definition Documentation

#### 5.6.3.1 #define EL_TOTAL_FILES (sizeof(EL_src_file_name)/sizeof(char*))

### 5.6.4 Variable Documentation

#### 5.6.4.1 const char* EL_src_file_name[ ]

**Initial value:**

```
= {
     "ErrorLog.h",
     "ErrorLog.cc",


     "schema.h",
     "schema.cc",
     "rowtable.h",
     "rowtable.cc",
     "cursor.h",
     "cursor.cc",
     "hashindex.h",
     "hashindex.cc",
     "storeprocedure.h",
     "storeprocedure.cc",
     "tpcserver.h",
     "tpcserver.cc",
    "debug_Error.cc",

     NULL
}
```

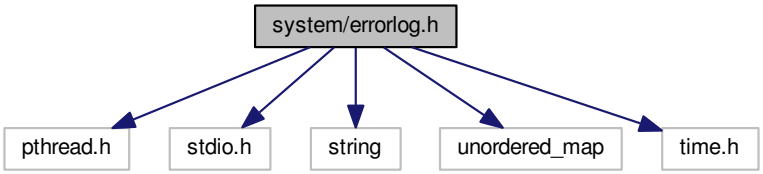
#### 5.6.4.2 ErrorLog _Thread_local* thread_el = NULL

## 5.7 system/errorlog.d File Reference

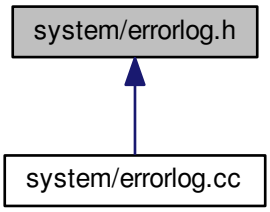
## 5.8 system/errorlog.h File Reference

```
#include <pthread.h>
#include <stdio.h>
#include <string>
#include <unordered_map>
#include <time.h>
```

Include dependency graph for errorlog.h:



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




## Classes

- class ErrorLog

  *an array of source file names*

## Macros

- #define _Thread_local __thread
- #define EL_ASSERT(t)
- #define EL_BAD_FILEID (99999999)
- #define EL_DEBUG 1
- #define EL_ERRCODE() (thread_el->getErrorCode())
- #define EL_ERRMSG() (thread_el->getErrorMsg())
- #define EL_ERROR 4
- #define EL_ERROR_CODE(fileid, lineno) ((fileid)*100000 + (lineno))
- #define EL_GET_FILEID(errcode) ((errcode)/100000)
- #define EL_GET_FILENAME(errcode) (ErrorLog::id2Name(EL_GET_FILEID(errcode)))
- #define EL_GET_LINENO(errcode) ((errcode)%100000)
- #define EL_INFO 2
- #define EL_LEVEL_COMPILE EL_INFO
- #define EL_LOG_DEBUG(...)
- #define EL_LOG_ERROR(...) thread_el->log(EL_ERROR,__FILE__,__LINE__,__VA_ARGS__)
- #define EL_LOG_INFO(...)
- #define EL_LOG_SERIOUS(...) thread_el->log(EL_SERIOUS,__FILE__,__LINE__,__VA_ARGS__)
- #define EL_LOG_WARN(...)
- #define EL_OK (0)
- #define EL_RESET() (thread_el->reset())
- #define EL_SERIOUS 5
- #define EL_WARN 3

## Variables

- const char * EL_src_file_name [ ]
- _Thread_local ErrorLog * thread_el

### 5.8.1 Detailed Description

**Author**

Shimin Chen chensm@ict.ac.cn

**Version**

0.1

### 5.8.2 Description

This file provides the error handling and logging utility.

1. error code The error code is a decimal number with 8 digits:

```
FFFLLLLL
```

The higher 3 digits show the source file ID, while the lower 5 digits indicate the line number in the source file, where the error occurs.

The following macros are useful:

```
EL_GET_FILEID(err_code)
EL_GET_LINENO(err_code)
EL_GET_FILENAME(err_code)
```

1. initialize

(1) main() must call ErrorLog::init(int level, const char *logfile);

(2) then every thread must new an ErrorLog instance:

```
thread_el= new ErrorLog("thread_name");
```

(3) put source file names into EL_src_file_name[] in ErrorLog.cc

1. normal use

   EL_LOG_DEBUG(format, args, ...); EL_LOG_INFO(format, args, ...); EL_LOG_WARN(format, args, ...); E↩ L_LOG_ERROR(format, args, ...); EL_LOG_SERIOUS(format, args, ...);

   The message will be written into the specified log file as follows

   [thread][level][file:lineno] message a call stack trace will be shown for error and serious messages.

   Moreover, an assertion can be written as:

   EL_ASSERT(expression);

   The expression is a test that evaluates to a True or False value. If the value is False, then a debug assertion message will be generated and written to the log.

2. In a worker thread:

   (1) reset and clear the error messages

   EL_RESET();

   (2) then follow the above to log messages

   (3) finally, obtain error code and message as follows

   int err_code= EL_ERRCODE(); const char *err_msg= EL_ERRMSG();

   (4) optionally, flush the log file

   ErrorLog::flushLog();

3. Before exiting, call the following globally once

   ErrorLog::closeLog();

### 5.8.3 Macro Definition Documentation

#### 5.8.3.1 #define _Thread_local __thread

#### 5.8.3.2 #define EL_ASSERT( *t* )

#### 5.8.3.3 #define EL_BAD_FILEID (99999999)

#### 5.8.3.4 #define EL_DEBUG 1

#### 5.8.3.5 #define EL_ERRCODE( ) (thread_el->getErrorCode())

#### 5.8.3.6 #define EL_ERRMSG( ) (thread_el->getErrorMsg())

#### 5.8.3.7 #define EL_ERROR 4

#### 5.8.3.8 #define EL_ERROR_CODE( *fileid, lineno* ) ((fileid)*100000 + (lineno))

#### 5.8.3.9 #define EL_GET_FILEID( *errcode* ) ((errcode)/100000)

#### 5.8.3.10 #define EL_GET_FILENAME( *errcode* ) (ErrorLog::id2Name(EL_GET_FILEID(errcode)))

#### 5.8.3.11 #define EL_GET_LINENO( *errcode* ) ((errcode)%100000)

#### 5.8.3.12 #define EL_INFO 2

#### 5.8.3.13 #define EL_LEVEL_COMPILE EL_INFO

#### 5.8.3.14 #define EL_LOG_DEBUG( ... )

#### 5.8.3.15 #define EL_LOG_ERROR( ... ) thread_el->log(EL_ERROR,__FILE__,__LINE__,__VA_ARGS__)

#### 5.8.3.16 #define EL_LOG_INFO( ... )

**Value:**

```
do{ if (EL_INFO>=ErrorLog::el_level) \
        thread_el->log(EL_INFO,__FILE__,__LINE__,__VA_ARGS__);}while(0)
```

#### 5.8.3.17 #define EL_LOG_SERIOUS( ... ) thread_el->log(EL_SERIOUS,__FILE__,__LINE__,__VA_ARGS__)

#### 5.8.3.18 #define EL_LOG_WARN( ... )

**Value:**

```
do{ if (EL_WARN>=ErrorLog::el_level) \
        thread_el->log(EL_WARN,__FILE__,__LINE__,__VA_ARGS__);}while(0)
```

#### 5.8.3.19 #define EL_OK (0)

#### 5.8.3.20 #define EL_RESET( ) (thread_el->reset())

#### 5.8.3.21 #define EL_SERIOUS 5

#### 5.8.3.22 #define EL_WARN 3

### 5.8.4 Variable Documentation

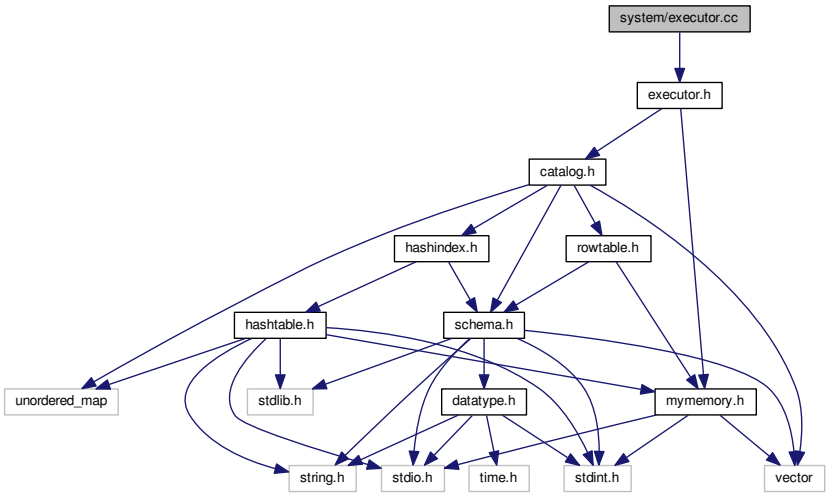
#### 5.8.4.1 const char* EL_src_file_name[ ]

#### 5.8.4.2 _Thread_local ErrorLog* thread_el

## 5.9 system/executor.cc File Reference

```
#include "executor.h"
```

Include dependency graph for executor.cc:




### Functions

- char * constchar2char (const char *str)
- uint32_t gethash (char *key, BasicType *type)
- string int2str (int64_t x)
- bool ReadRowToResult (RowTable *src, int64_t row_rank, int64_t col_num, ResultTable *dest)
- int64_t round2 (int64_t size)

### Variables

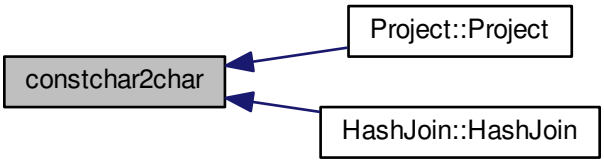
- char ∗∗ hashjoin_cmpSrcB_tables = new char∗ [2]
- char ∗ hashjoin_format_value = new char[128]
- int64_t project_tabout_id = 456
- int64_t tabout_id_hashjoin = 777

## 5.9.1 Function Documentation

#### 5.9.1.1 char∗ constchar2char ( const char ∗ *str* )

transfer const char to char

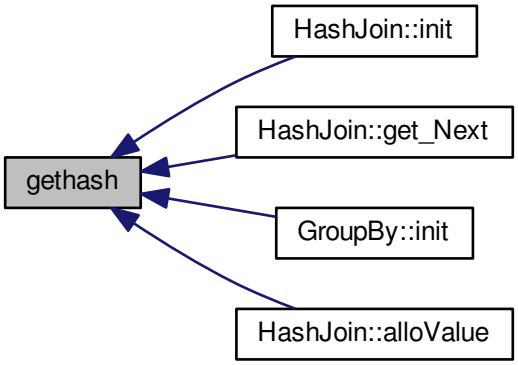
Here is the caller graph for this function:




#### 5.9.1.2 uint32_t gethash ( char ∗ *key*, BasicType ∗ *type* )

get hash number

Here is the caller graph for this function:



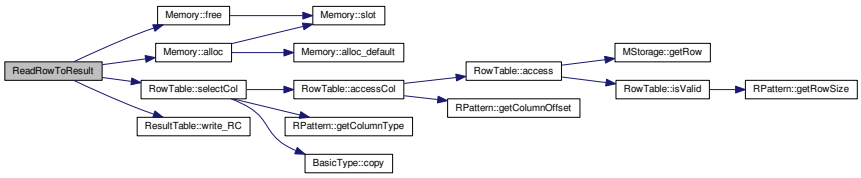
#### 5.9.1.3 string int2str ( int64_t *x* )

transfer int to string format

#### 5.9.1.4 bool ReadRowToResult ( RowTable * *src,* int64_t *row_rank,* int64_t *col_num,* ResultTable * *dest* )

read row to result table

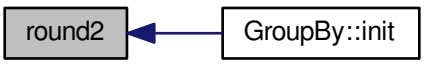
Here is the call graph for this function:




#### 5.9.1.5 int64_t round2 ( int64_t *size* )

get a number around size,powdered by 2

Here is the caller graph for this function:




### 5.9.2 Variable Documentation

#### 5.9.2.1 char** hashjoin_cmpSrcB_tables = new char* [2]

#### 5.9.2.2 char* hashjoin_format_value = new char[128]

#### 5.9.2.3 int64_t project_tabout_id = 456

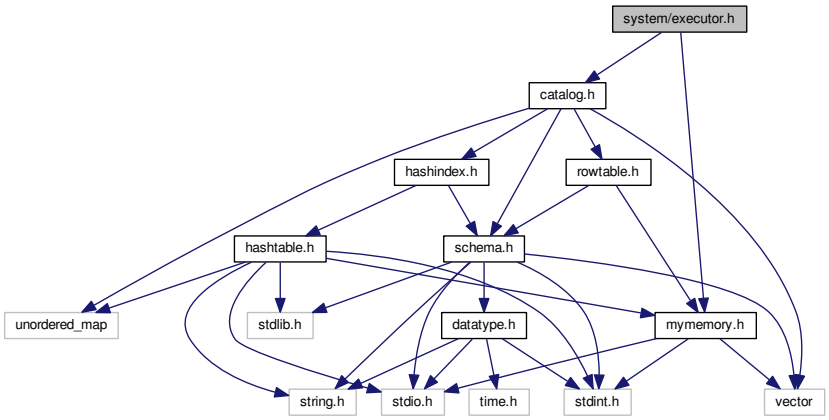
#### 5.9.2.4 int64_t tabout_id_hashjoin = 777

## 5.10 system/executor.d File Reference

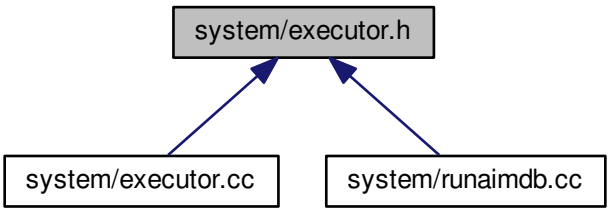
## 5.11 system/executor.h File Reference

```
#include "catalog.h"
#include "mymemory.h"
```

Include dependency graph for executor.h:




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




### Classes

- struct Condition
- struct Conditions
- class Executor
- class Filter
- class GroupBy
- class HashJoin
- class Operator
- class OrderBy
- class Project
- struct RequestColumn
- struct RequestTable
- class ResultTable
- class Scan
- class SelectQuery

### Enumerations

- enum AggrerateMethod {
  NONE_AM = 0, COUNT, SUM, AVG,
  MAX, MIN, MAX_AM }
- enum CompareMethod {
  NONE_CM = 0, LT, LE, EQ,
  NE, GT, GE, LINK,
  MAX_CM }

### Functions

- uint32_t gethash (char *key, BasicType *type)

## 5.11.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

## 5.11.2 DESCRIPTION

definition of executor

## 5.11.3 Enumeration Type Documentation

### 5.11.3.1 enum AggrerateMethod

aggrerate method.

**Enumerator**

***NONE_AM*** none

***COUNT*** count of rows

***SUM*** sum of data

***AVG*** average of data

***MAX*** maximum of data

***MIN*** minimum of data

***MAX_AM***

#### 5.11.3.2 enum CompareMethod

compare method.

**Enumerator**

***NONE_CM***

***LT*** less than

***LE*** less than or equal to

***EQ*** equal to

***NE*** not equal than

***GT*** greater than

***GE*** greater than or equal to

***LINK*** join

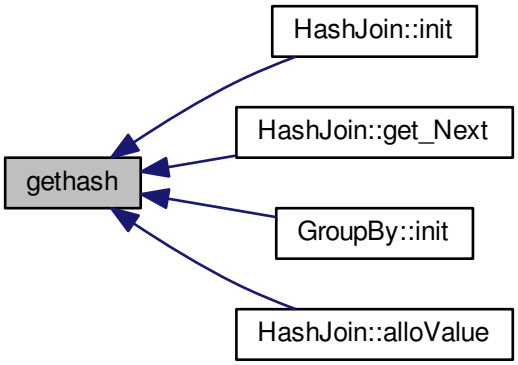
***MAX_CM***

### 5.11.4 Function Documentation

#### 5.11.4.1 uint32_t gethash ( char * *key*, BasicType * *type* )

get hash number

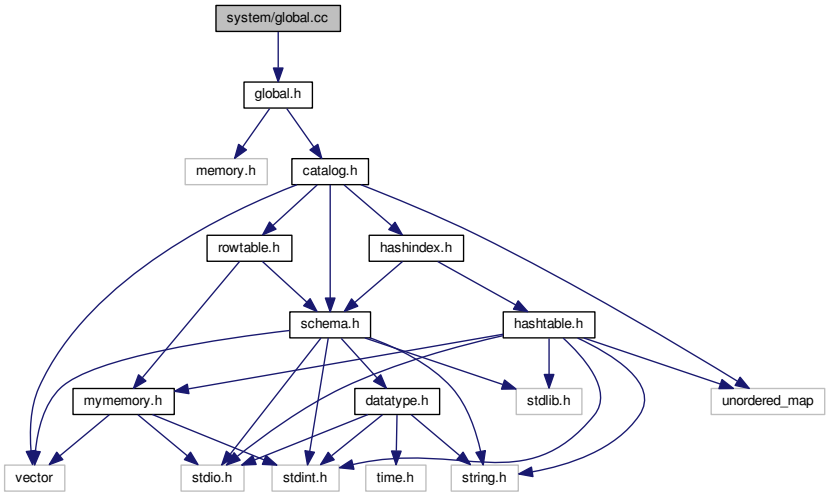
Here is the caller graph for this function:



## 5.12 system/global.cc File Reference

```
#include "global.h"
```

Include dependency graph for global.cc:




### Functions

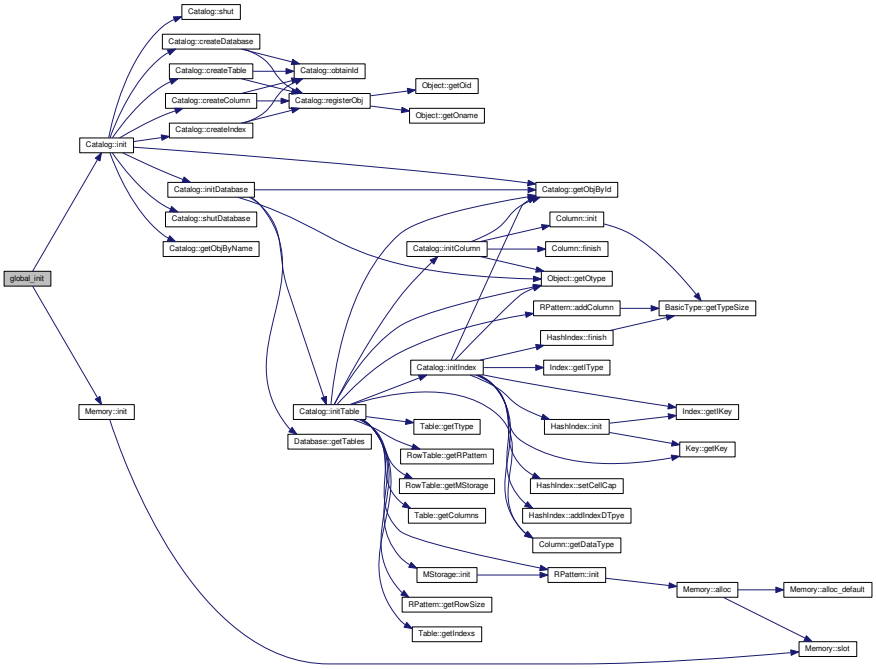
- int global_init ()
- int global_shut ()

### 5.12.1 Function Documentation

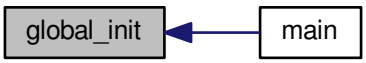
#### 5.12.1.1 int global_init ( )

init memory and catalog.

Here is the call graph for this function:



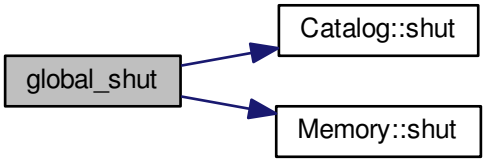

Here is the caller graph for this function:



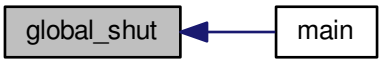

#### 5.12.1.2 int global_shut ( )

shut down catalog and memory.

Here is the call graph for this function:



Here is the caller graph for this function:

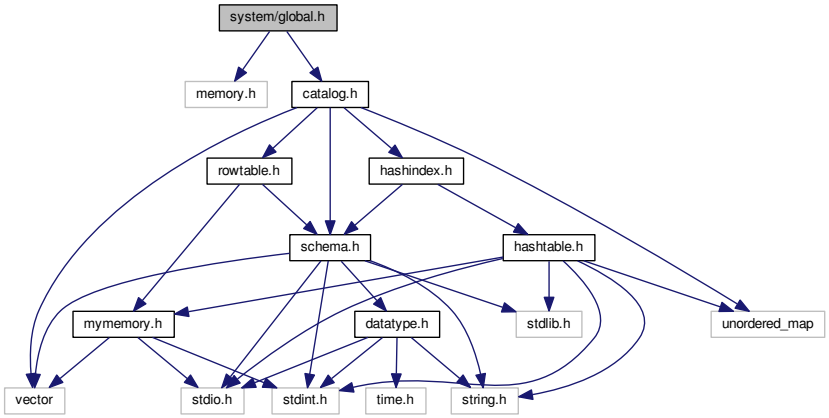



## 5.13 system/global.d File Reference

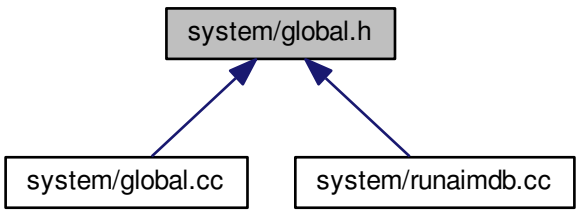
## 5.14 system/global.h File Reference

```
#include "memory.h"
#include "catalog.h"
```

Include dependency graph for global.h:




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



## Macros

- #define BNODE_POINTERS_NUM (16)
- #define GLOBAL_MEMORY_MINIMUM (1L<< 3)
- #define GLOBAL_MEMORY_SIZE (1L<<30)

## Functions

- int global_init ()
- int global_shut ()

## Variables

- Catalog g_catalog
- Memory g_memory

### 5.14.1 Macro Definition Documentation

#### 5.14.1.1 #define BNODE_POINTERS_NUM (16)

#### 5.14.1.2 #define GLOBAL_MEMORY_MINIMUM (1L<< 3)

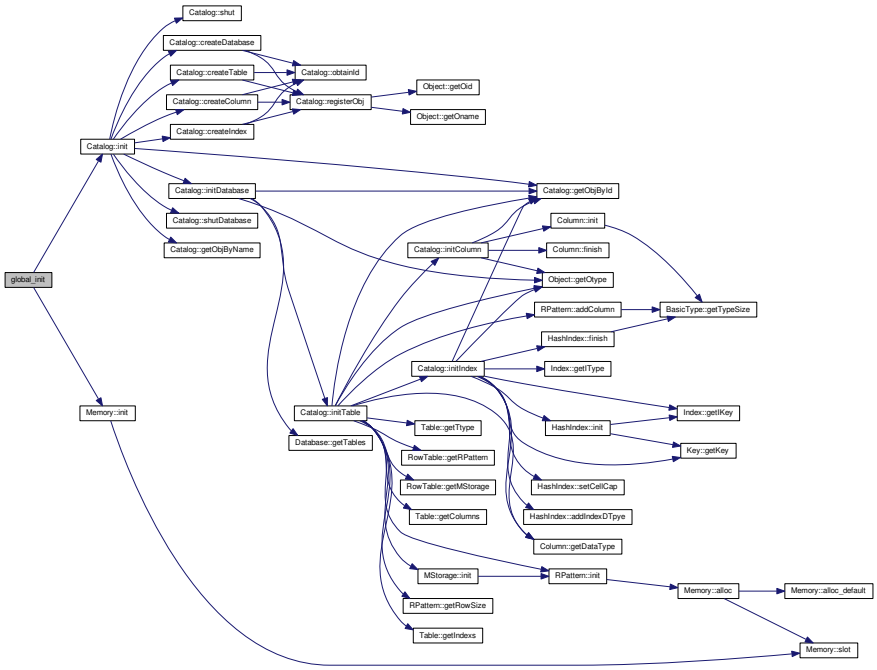
#### 5.14.1.3 #define GLOBAL_MEMORY_SIZE (1L<<30)

### 5.14.2 Function Documentation

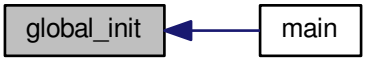
#### 5.14.2.1 int global_init ( )

init memory and catalog.

Here is the call graph for this function:



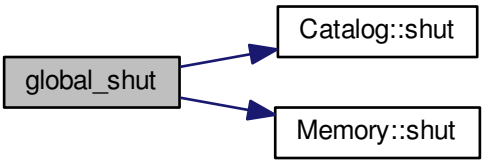
Here is the caller graph for this function:



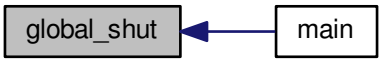

#### 5.14.2.2 int global_shut ( )

shut down catalog and memory.

Here is the call graph for this function:




Here is the caller graph for this function:




### 5.14.3 Variable Documentation

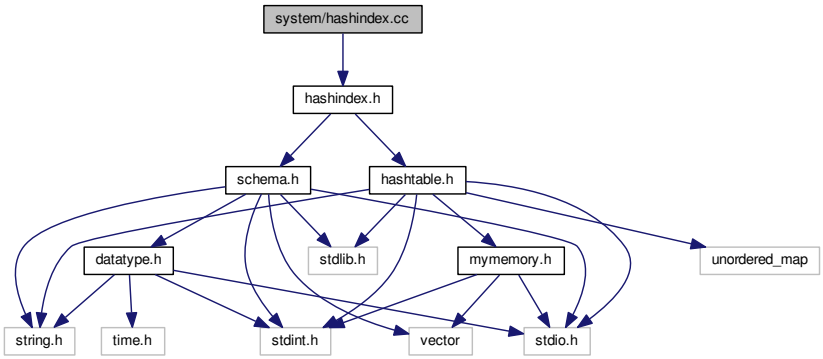
#### 5.14.3.1 Catalog g_catalog

#### 5.14.3.2 Memory g_memory

## 5.15 system/hashindex.cc File Reference

```
#include "hashindex.h"
```

Include dependency graph for hashindex.cc:




### 5.15.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

### 5.15.2 DESCRIPTION

hash index, here we support all type data. and you can use multi-keys, and I will guarantee it's right the value accessed at Element is the pointer of related recored. this implementation permits duplicated key with different value to be inserted, but not element with same key and value del operation will only del the first one which meet requirement, if you want delete all, you can call it many times till it returns false

usage:

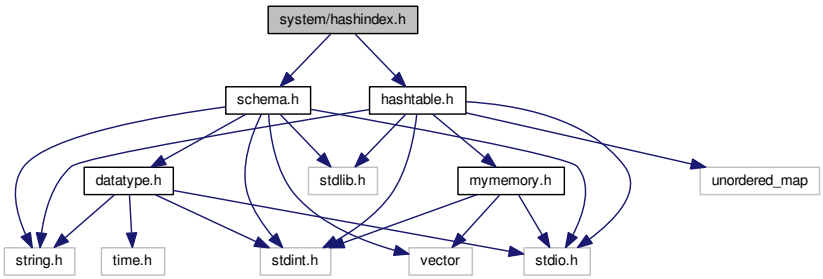
for each insert,del,look,scan, this file provides 2 same name method to handle 2 type data format you can use (1) for lookup, you should all use set_ls to set HashInfo with proper value, the first param is valid, leave the second to be NULL, HashInfo can help you iterately get values you need. (2) call lookup to get the value iterately.

## 5.16 system/hashindex.d File Reference

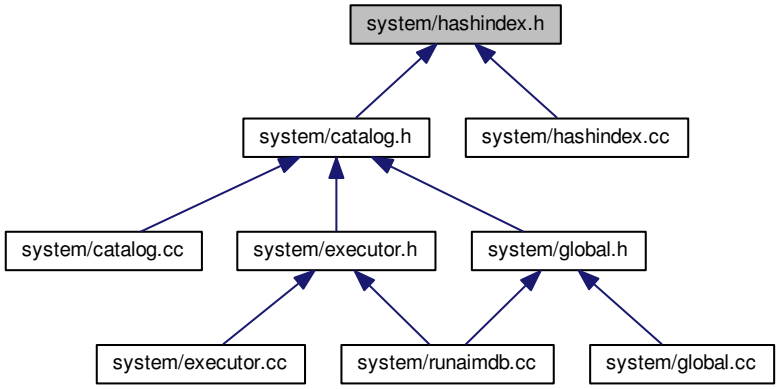
## 5.17 system/hashindex.h File Reference

```
#include "schema.h"
#include "hashtable.h"
```

Include dependency graph for hashindex.h:




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




### Classes

- class HashIndex
- struct HashInfo

### Macros

- #define HASHINFO_CAPICITY (8)

### 5.17.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

### 5.17.2 DESCRIPTION

hash index, here we support all type data. and you can use multi-keys, and I will guarantee it's right the value accessed at Element is the pointer of related recored. this implementation permits duplicated key with different value to be inserted, but not element with same key and value del operation will only del the first one which meet requirement, if you want delete all, you can call it many times till it returns false

usage:

for each insert,del,look,scan, this file provides 2 same name method to handle 2 type data format you can use (1) for lookup, you should all use set_ls to set HashInfo with proper value, the first param is valid, leave the second to be NULL, HashInfo can help you iterately get values you need. (2) call lookup to get the value iterately.

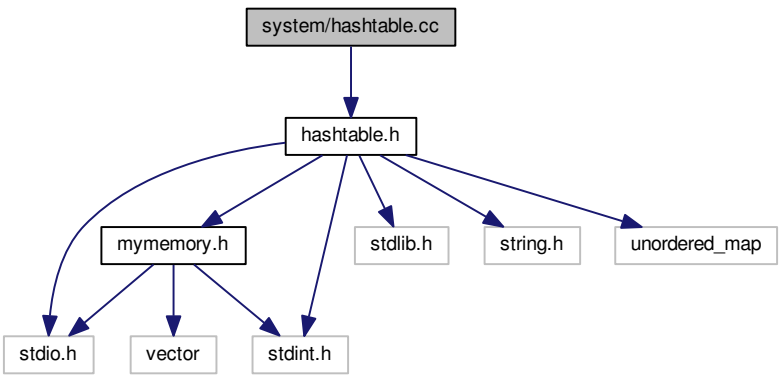
### 5.17.3 Macro Definition Documentation

#### 5.17.3.1 #define HASHINFO_CAPICITY (8)

## 5.18 system/hashtable.cc File Reference

```
#include "hashtable.h"
```

Include dependency graph for hashtable.cc:



### Macros

- #define ESTIMATE_ERROR (1024)

### 5.18.1 Macro Definition Documentation

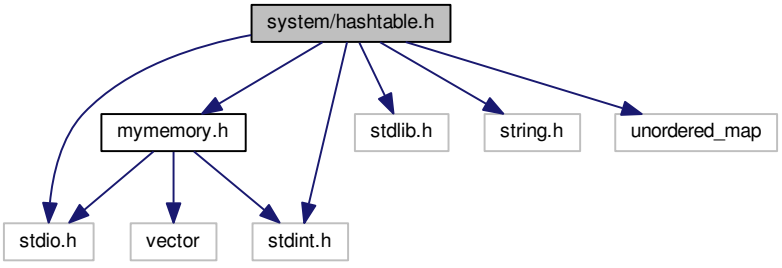
#### 5.18.1.1 #define ESTIMATE_ERROR (1024)

## 5.19 system/hashtable.d File Reference

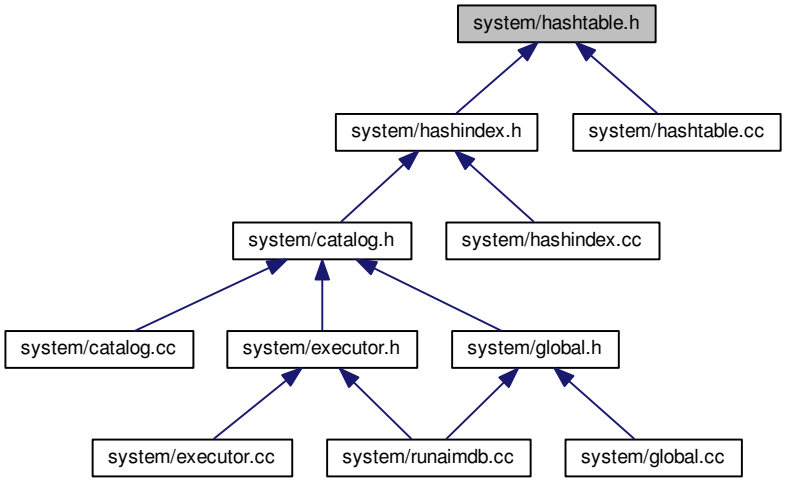
## 5.20 system/hashtable.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <unordered_map>
#include "mymemory.h"
```

Include dependency graph for hashtable.h:



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




## Classes

- class HashCell
- class Hashcode_Ptr
- class HashTable

## Macros

- #define hc_capacity hc_union.num_2_or_more.capacity
- #define hc_ent hc_union.ent
- #define hc_ents hc_union.num_2_or_more.ents

### 5.20.1 Macro Definition Documentation

#### 5.20.1.1 #define hc_capacity hc_union.num_2_or_more.capacity

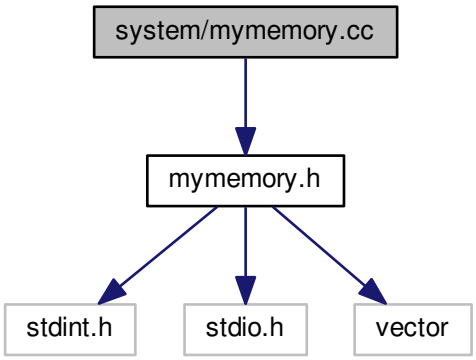
#### 5.20.1.2 #define hc_ent hc_union.ent

#### 5.20.1.3 #define hc_ents hc_union.num_2_or_more.ents

## 5.21 system/mymemory.cc File Reference

```
#include "mymemory.h"
```

Include dependency graph for mymemory.cc:




### Variables

- Memory g_memory

### 5.21.1 Detailed Description

liugang(liugang@ict.ac.cn)

**Version**

0.1

### 5.21.2 DESCRIPTION

Memory is system own manager to alloc and free slab of different size the minmux size is sizeof(void∗), maxsize is given by m_array_list size

interface: int64_t alloc (char ∗&p, int64_t size) int64_t free (char ∗p, int64_t size) you should put down the size of memory allocated, when you free back, you need this data

### 5.21.3 Variable Documentation

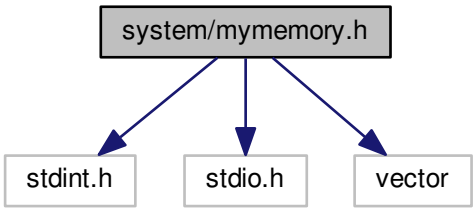
#### 5.21.3.1 Memory g_memory

## 5.22 system/mymemory.d File Reference

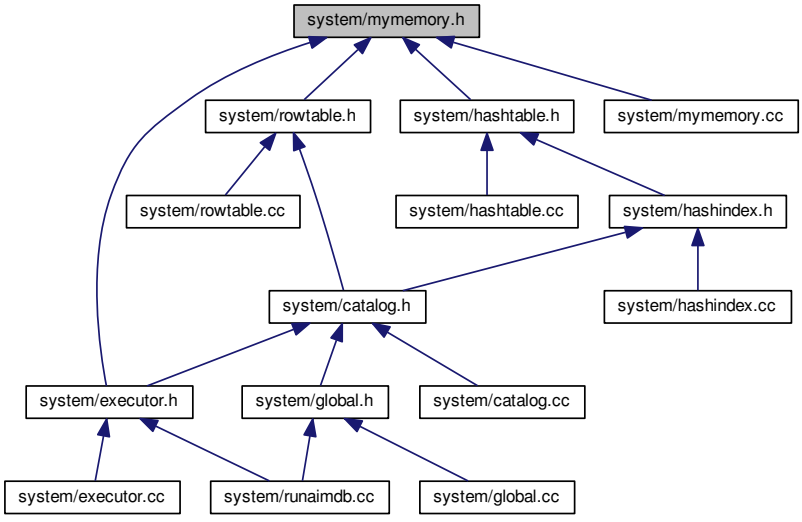
## 5.23 system/mymemory.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <vector>
```

Include dependency graph for mymemory.h:




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



### Classes

- class Memory

### Macros

- #define MEMORY_OK 0

### Variables

- Memory g_memory

## 5.23.1 Detailed Description

liugang(liugang@ict.ac.cn)

**Version**

0.1

## 5.23.2 DESCRIPTION

Memory is system own manager to alloc and free slab of different size the minmux size is sizeof(void∗), maxsize is given by m_array_list size

interface: int64_t alloc (char ∗&p, int64_t size) int64_t free (char ∗p, int64_t size) you should put down the size of memory allocated, when you free back, you need this data

## 5.23.3 Macro Definition Documentation

### 5.23.3.1 #define MEMORY_OK 0

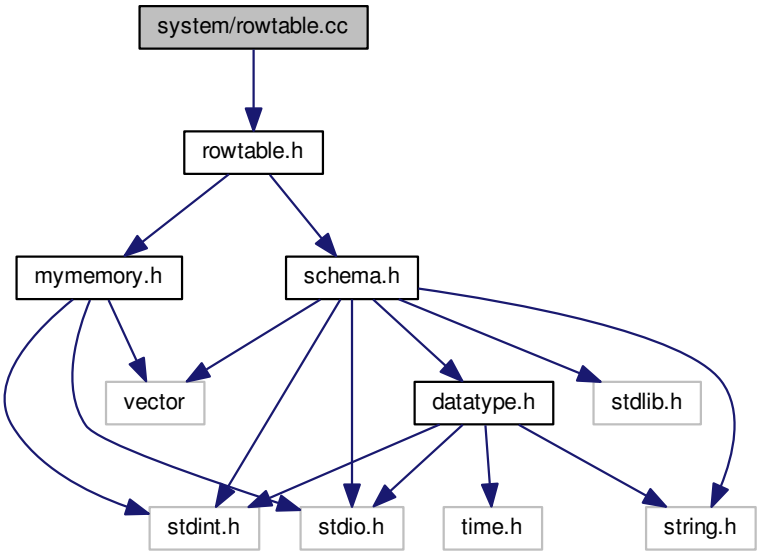
## 5.23.4 Variable Documentation

### 5.23.4.1 Memory g_memory

## 5.24 system/rowtable.cc File Reference

```
#include "rowtable.h"
```

Include dependency graph for rowtable.cc:




### 5.24.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

### 5.24.2 DESCRIPTION

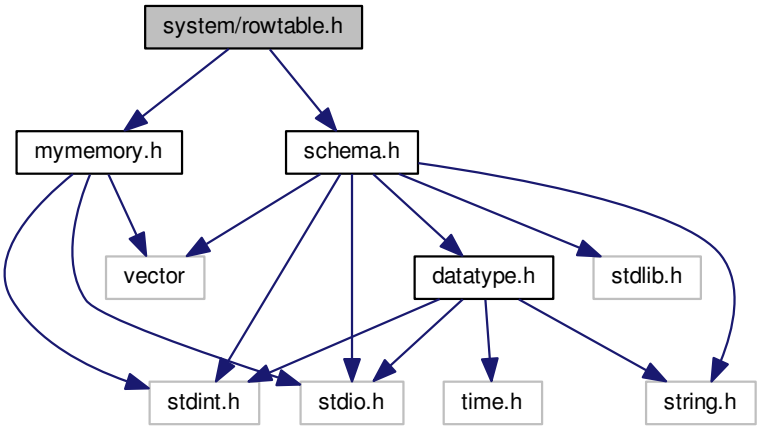
rowtable implementation, this file implement all interface required by class table data space is managed by g_↩ memory, which decreases malloc overhead when create rowtable, I will make one more column to represent its validation. if delete a record, put down the label to set it "invalid" for index in this table, delete the entry inside index

## 5.25 system/rowtable.d File Reference

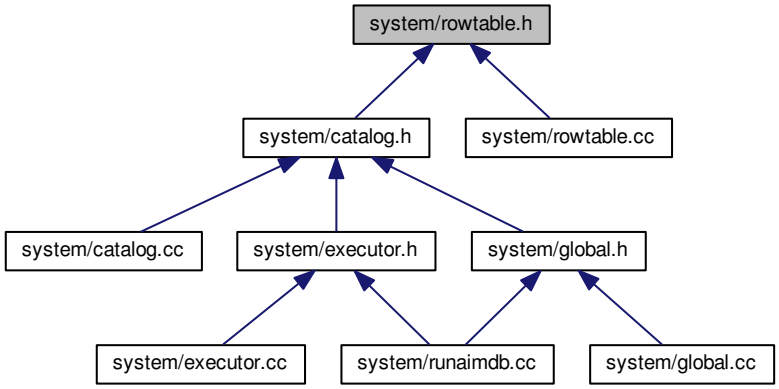
## 5.26 system/rowtable.h File Reference

```
#include "mymemory.h"
#include "schema.h"
```

Include dependency graph for rowtable.h:




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




### Classes

- class MStorage
- class RowTable
- class RPattern

### Variables

- Memory g_memory

### 5.26.1 Detailed Description

#### Author

liugang(liugang@ict.ac.cn)

#### Version

0.1

### 5.26.2 DESCRIPTION

rowtable implementation, this file implement all interface required by class table data space is managed by g_↩ memory, which decreases malloc overhead when create rowtable, I will make one more column to represent its validation. if delete a record, put down the label to set it "invalid" for index in this table, delete the entry inside index

basic usage: using rowtable interface surrounded by "//----------" will be enough for you

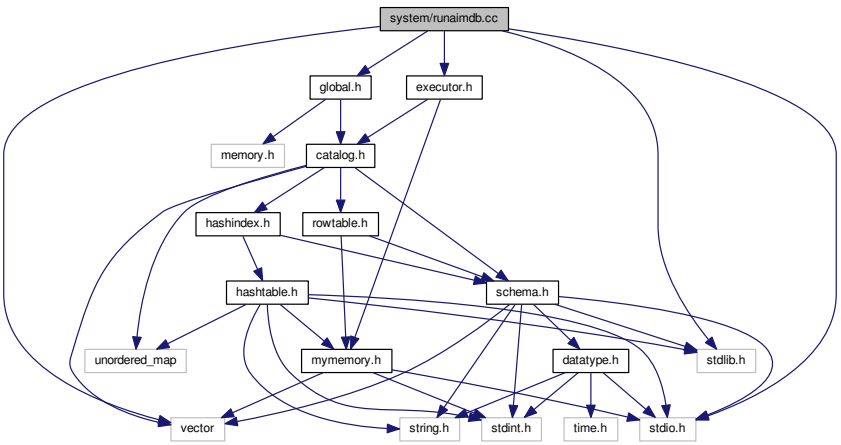
### 5.26.3 Variable Documentation

#### 5.26.3.1 Memory g_memory

## 5.27 system/runaimdb.cc File Reference

```
#include "global.h"
#include "executor.h"
#include <stdio.h>
#include <stdlib.h>
#include <vector>
```

Include dependency graph for runaimdb.cc:



## Functions

- int load_data (const char *tablename[ ], const char *data_dir, int number)
- int load_schema (const char *filename)
- int main (int argc, char *argv[ ])
- int test (void)
- int testOne (int which)

## Variables

- int print_flag = false
- SelectQuery querys [22]

  ----------*tpch test*----------
- const char * table_name [ ]

### 5.27.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

### 5.27.2 DESCRIPTION

the main entrance of AIMDB

### 5.27.3 Function Documentation

#### 5.27.3.1 int load_data ( const char * *tablename[ ],* const char * *data_dir,* int *number* )

load table data from txt files

**Parameters**

| | |
|---|---|
| *tablename* | names of tables |
| *data_dir* | the dir of data files corresponding to the table, end with '/' |
| *number* | number of tables to load data |

**Return values**

| | |
|---|---|
| *0* | success |
| *<0* | faliure |

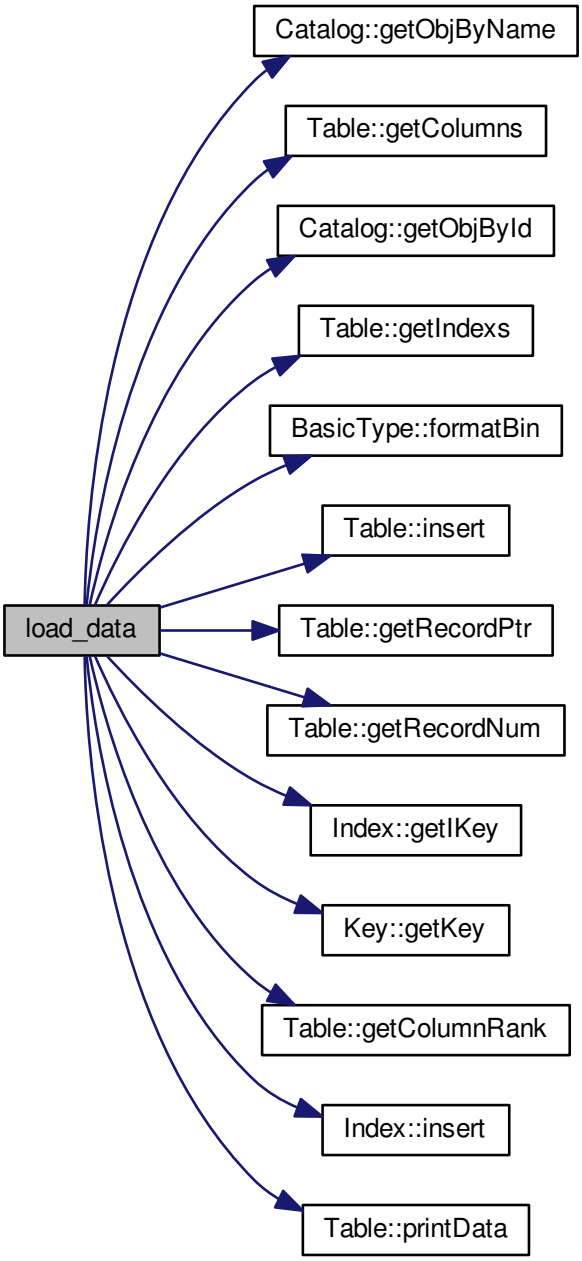
naming rule of data files

each data file should be named "table_name.tab" and meet the following requirement

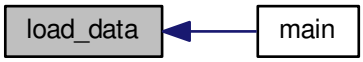
data format

(1) split by one '', a row ends with '
' (2) no empty row

Here is the call graph for this function:



Here is the caller graph for this function:




#### 5.27.3.2 int load_schema ( const char * *filename* )

load a database schema from a txt file.

**Parameters**

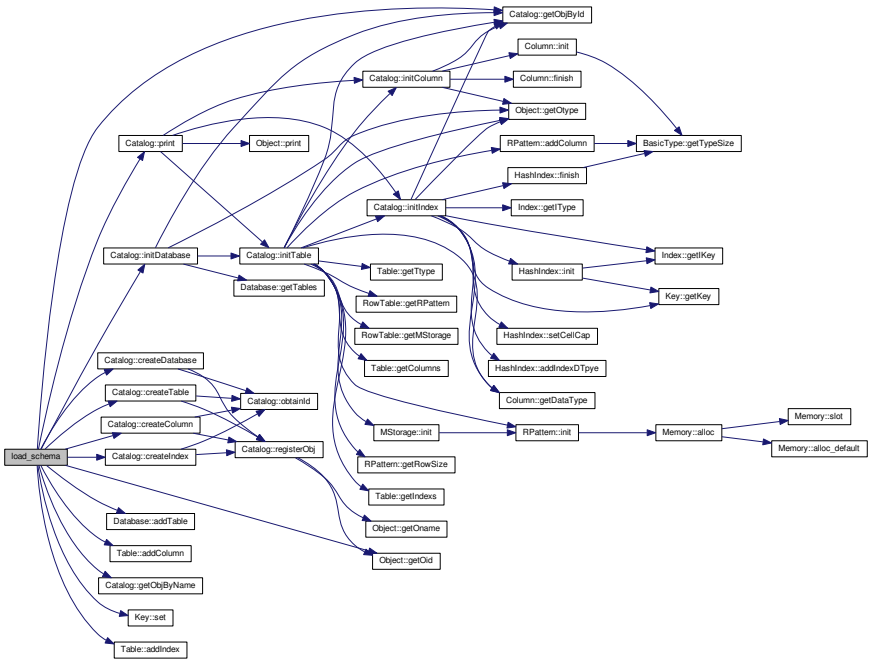
| | |
|---|---|
| *filename* | name of schema file, the schema must meet the folowing condition |

**Return values**

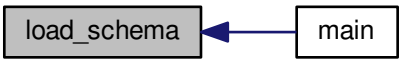
| | |
|---|---|
| *0* | success #retval <0 faliure |

schema format (1) split by one '', a row ends with '
' (2) claim Database,Table,column,index in order (3) no empty row

Here is the call graph for this function:

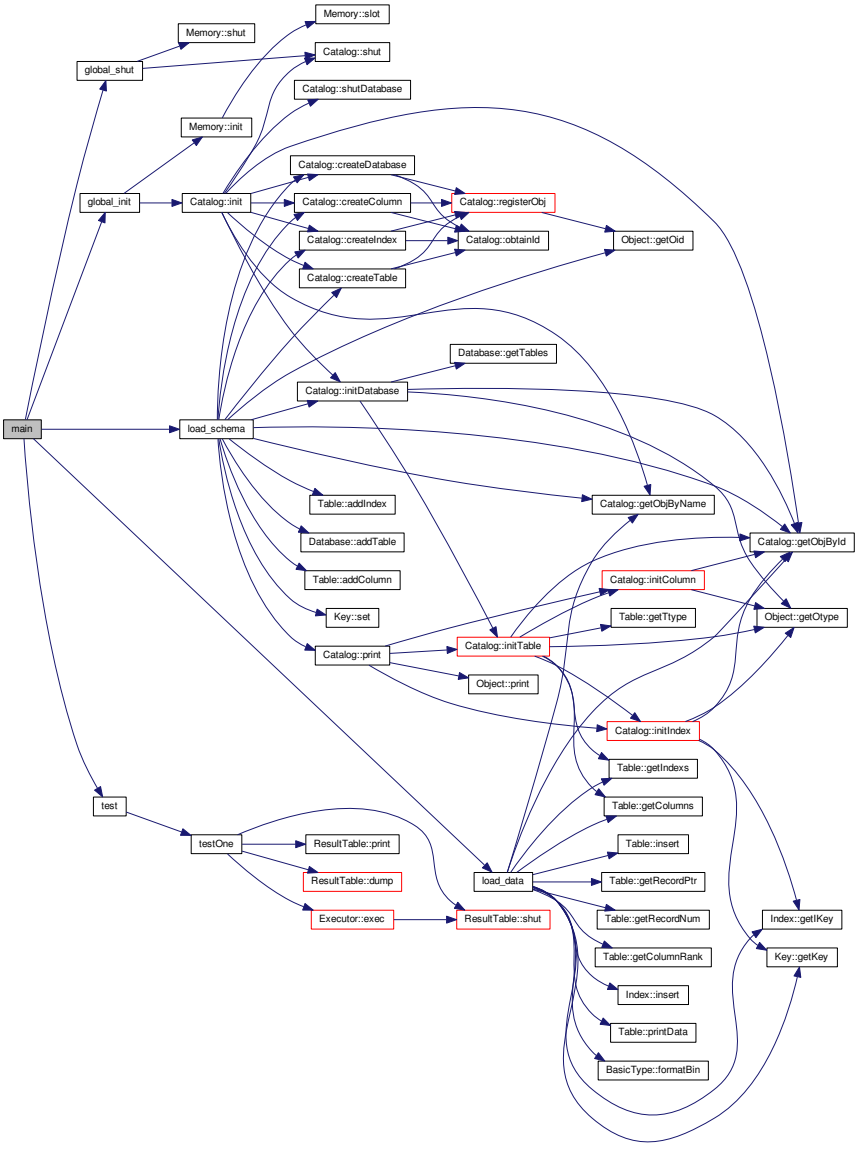


Here is the caller graph for this function:



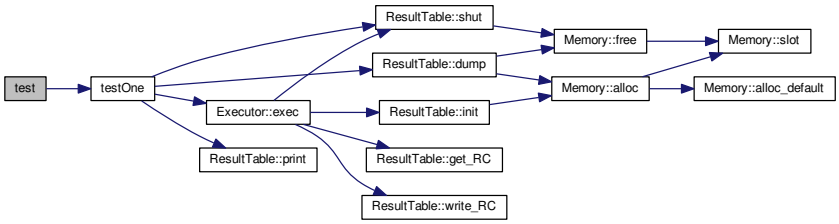

#### 5.27.3.3 int main ( int *argc*, char * *argv[]* )

Here is the call graph for this function:



#### 5.27.3.4 int test ( void )

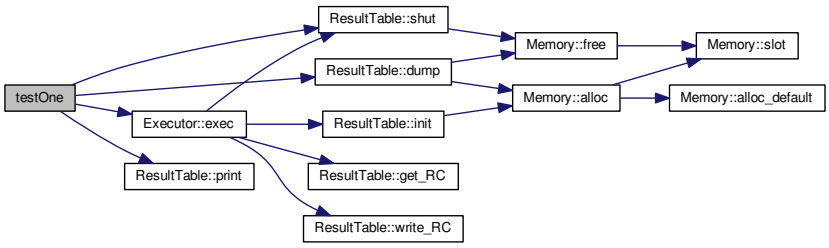
Here is the call graph for this function:



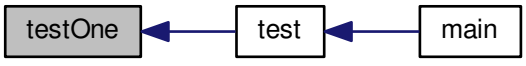

Here is the caller graph for this function:


#### 5.27.3.5 int testOne ( int *which* )

Here is the call graph for this function:




Here is the caller graph for this function:



### 5.27.4 Variable Documentation

#### 5.27.4.1 int print_flag = false

#### 5.27.4.2 SelectQuery querys[22]

----------tpch test----------

#### 5.27.4.3 const char* table_name[ ]

**Initial value:**

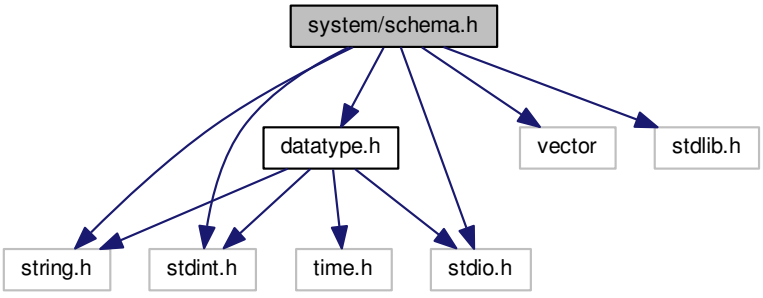
```
= {
    "part",
    "supplier",
    "partsupp",
    "customer",
    "nation",
    "lineitem",
    "region",
    "orders"
}
```

## 5.28 system/runaimdb.d File Reference

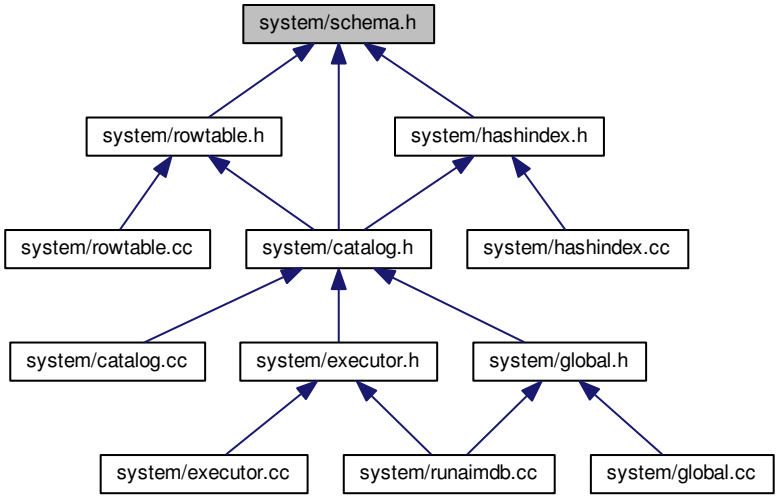
## 5.29 system/schema.h File Reference

```
#include <string.h>
#include <vector>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include "datatype.h"
```

Include dependency graph for schema.h:



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




## Classes

- class Column
- class Database
- class Index
- class Key
- class Object
- class Table

## Macros

- #define OBJ_NAME_MAX (128)

## Enumerations

- enum ColumnType {
  INVID_C = 0, INT8, INT16, INT32,
  INT64, FLOAT32, FLOAT64, CHARN,
  DATE, TIME, DATETIME, MAXTYPE_C }
- enum IndexType {
  INVID_I = 0, HASHINDEX, BPTREEINDEX, ARTTREEINDEX,
  MAXTYPE_I }
- enum ObjectType {
  INVID_O = 0, DATABASE, TABLE, COLUMN,
  INDEX, MAXTYPE_O }
- enum TableType { INVID_T = 0, ROWTABLE, COLTABLE, MAXTYPE_T }

### 5.29.1 Detailed Description

**Author**

liugang(liugang@ict.ac.cn)

**Version**

0.1

### 5.29.2 DESCRIPTION

this file defines the abstract class of four primary elements of database system, these abstract classes provide uniform interface for upper application

basic interface: init,finish,shut,select,insert,update,del,selectCol,lookup,scan

notice: for insert,del,update, input data requires to be processed by BasicType method formatBin, then call above function to actually put the into table example: char date[10] = 1970-01-01 TypeDate type; char buff[10]; type.format (buff, date); // in buff, it's stored as time_t with 4 Byte then, you can perform insert ()

### 5.29.3 Macro Definition Documentation

#### 5.29.3.1 #define OBJ_NAME_MAX (128)

### 5.29.4 Enumeration Type Documentation

#### 5.29.4.1 enum ColumnType

an enum for column.

**Enumerator**

***INVID_C***

***INT8*** int8

***INT16*** int16

***INT32*** int32

***INT64*** int64

***FLOAT32*** float32

***FLOAT64*** float64

***CHARN*** charn, fixed length string

***DATE*** days from 1970-01-01 till current DATE

***TIME*** seconds from 00:00:00 till current TIME

***DATETIME*** seconds from 1970-01-01 00:00:00 till current DATETIME

***MAXTYPE_C***

#### 5.29.4.2 enum IndexType

an enum for Index.

**Enumerator**

***INVID_I***

***HASHINDEX*** hash index

***BPTREEINDEX*** bptree index

***ARTTREEINDEX*** art tree index

***MAXTYPE_I***

#### 5.29.4.3 enum ObjectType

an enum for ObjectType label.

**Enumerator**

***INVID_O***

***DATABASE*** database

***TABLE*** table

***COLUMN*** column

***INDEX*** index

***MAXTYPE_O***

#### 5.29.4.4 enum TableType

an enum for Table.

**Enumerator**

***INVID_T***

***ROWTABLE*** row table

***COLTABLE*** column table

***MAXTYPE_T***