# AIMDB

Generated by Doxygen 1.9.4

# 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 AggreCondition Struct Reference

`#include <executor.h>`

### Public Attributes

- int column_rank
- AggregateMethod method

### 4.1.1 Detailed Description

definition of aggregete condition.

### 4.1.2 Member Data Documentation

#### 4.1.2.1 column_rank

`int AggreCondition::column_rank`

the rank of the column has aggregate method

#### 4.1.2.2 method

`AggregateMethod AggreCondition::method`

the aggregate method on that column

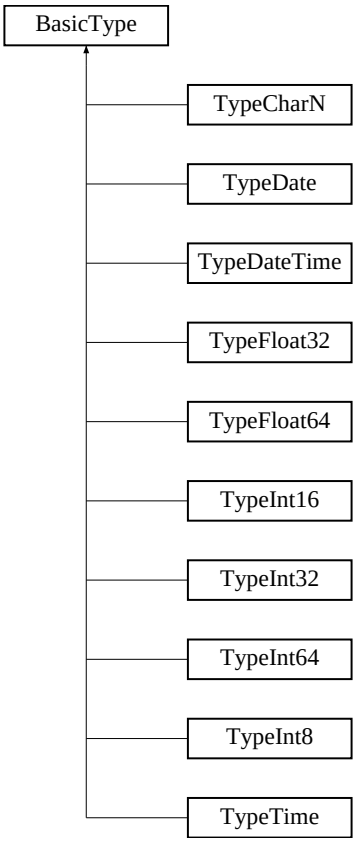
The documentation for this struct was generated from the following file:

- system/executor.h

## 4.2 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

### 4.2.1 Detailed Description

definition of class BasicType.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 BasicType()

```
BasicType::BasicType (
            TypeCode typecode,
            int64_t typesize )  [inline]
```

constructor.

#### 4.2.2.2 ~BasicType()

```
virtual BasicType::~BasicType ( )  [inline], [virtual]
```

destructor.

### 4.2.3 Member Function Documentation

#### 4.2.3.1 cmpEQ()

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

equal to.

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

#### 4.2.3.2 cmpGE()

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

greater than or equal to

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

#### 4.2.3.3 cmpGT()

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

greater than.

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

#### 4.2.3.4 cmpLE()

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

less than or equal to.

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

#### 4.2.3.5 cmpLT()

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

less than.

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

#### 4.2.3.6 copy()

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

copy from data to dest.

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

#### 4.2.3.7 formatBin()

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

extract bin format from data(txt) to dest.

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

#### 4.2.3.8 formatTxt()

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

extract txt format from data(bin) to dest.

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

#### 4.2.3.9 getTypeCode()

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

get type code of this data type.

#### 4.2.3.10 getTypeSize()

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

get data size when stored in bin format.

### 4.2.4 Member Data Documentation

#### 4.2.4.1 b_type_code

```
TypeCode BasicType::b_type_code  [protected]
```

data type code

#### 4.2.4.2 b_type_size

```
int64_t BasicType::b_type_size  [protected]
```

data type size

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

- system/datatype.h

## 4.3 bnode Class Reference

```
#include <pbtree.h>
```

### Public Member Functions

- Pointer8B & ch (int idx)
- char * chEndAddr (int idx)
- key_type & k (int idx)

### Public Attributes

- Pointer8B child [BKEY_NUM+1]
- key_type key [BKEY_NUM+1]

### 4.3.1 Member Function Documentation

#### 4.3.1.1 ch()

```
Pointer8B & bnode::ch (
            int idx )  [inline]
```

#### 4.3.1.2 chEndAddr()

```
char * bnode::chEndAddr (
            int idx )  [inline]
```

#### 4.3.1.3 k()

```
key_type & bnode::k (
            int idx )  [inline]
```

### 4.3.2 Member Data Documentation

#### 4.3.2.1 child

```
Pointer8B bnode::child[BKEY_NUM+1]
```

#### 4.3.2.2 key

```
key_type bnode::key[BKEY_NUM+1]
```

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

- system/pbtree.h

## 4.4 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, int64_t t_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

### 4.4.1 Detailed Description

definition of class Catalog.

### 4.4.2 Member Function Documentation

#### 4.4.2.1 createColumn()

```
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 |

#### 4.4.2.2 createDatabase()

```
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 |

#### 4.4.2.3 createIndex()

```
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 |

#### 4.4.2.4 createTable()

```
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 |

**Return values**

| | |
|---|---|
| *false* | failure |

#### 4.4.2.5 getObjById()

```
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 |

#### 4.4.2.6 getObjByName()

```
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 |

#### 4.4.2.7 init()

```
void Catalog::init (
            void ) [inline]
```

init operation.

#### 4.4.2.8 initColumn()

```
bool Catalog::initColumn (
            int64_t c_id )  [private]
```

init column

**Parameters**

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

**Return values**

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

#### 4.4.2.9 initDatabase()

```
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 |

#### 4.4.2.10 initIndex()

```
bool Catalog::initIndex (
            int64_t i_id,
            int64_t t_id )  [private]
```

init index

**Parameters**

| | |
|---|---|
| *i↩ _id* | which index to prepare |
| *t↩ _id* | index in which table |

**Return values**

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

#### 4.4.2.11 initTable()

```
bool Catalog::initTable (
            int64_t t_id )  [private]
```

init table

**Parameters**

| | |
|---|---|
| *t↩ _id* | which table to prepare |

**Return values**

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

#### 4.4.2.12 obtainId()

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

get a free object identifier

**Return values**

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

#### 4.4.2.13 print()

```
void Catalog::print (
            void  )  [inline]
```

print the catalog

#### 4.4.2.14 registerObj()

```
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 |

#### 4.4.2.15 shut()

```
bool Catalog::shut (
            void  )
```

shut down, free all memory of Objects.

#### 4.4.2.16 shutDatabase()

```
bool Catalog::shutDatabase (
            int64_t d_id )
```

shutdowm database

**Parameters**

| | |
|---|---|
| *d↩ _id* | which database to shut |

**Return values**

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

### 4.4.3 Member Data Documentation

#### 4.4.3.1 cl_id_obj

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

container of Objects

#### 4.4.3.2 cl_name_obj

```
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

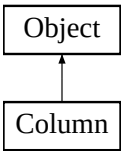
The documentation for this class was generated from the following files:

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

## 4.5 Column Class Reference

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

Inheritance 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 getCoffset (void)
- int64_t getCSize (void)
- ColumnType getCType (void)
- BasicType * getDataType (void)
- virtual bool init (void)
- virtual void print (void)
- int64_t setCoffset (int64_t offset)
- virtual bool shut (void)

## Private Attributes

- BasicType * c_datatype
- int64_t c_offset
- int64_t c_size
- ColumnType c_type

### 4.5.1 Detailed Description

definition of class Column.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 Column()

```
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 |

#### 4.5.2.2 ~Column()

```
virtual Column::~Column (
            void  )  [inline], [virtual]
```

destructor.

### 4.5.3 Member Function Documentation

#### 4.5.3.1 finish()

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

finish column setting.

#### 4.5.3.2 getCoffset()

```
int64_t Column::getCoffset (
            void  )  [inline]
```

get column offset.

#### 4.5.3.3 getCSize()

```
int64_t Column::getCSize (
            void  )  [inline]
```

get data dize.

#### 4.5.3.4 getCType()

```
ColumnType Column::getCType (
            void  )  [inline]
```

get column type.

#### 4.5.3.5 getDataType()

```
BasicType * Column::getDataType (
            void  )  [inline]
```

get data type of the column

#### 4.5.3.6 init()

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

init column.

#### 4.5.3.7 print()

```
virtual void Column::print (
            void  )  [inline], [virtual]
```

print column information

Reimplemented from Object.

#### 4.5.3.8 setCoffset()

```
int64_t Column::setCoffset (
            int64_t offset )  [inline]
```

get column offset.

#### 4.5.3.9 shut()

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

shut down column.

Reimplemented from Object.

### 4.5.4 Member Data Documentation

#### 4.5.4.1 c_datatype

```
BasicType* Column::c_datatype  [private]
```

column data type

#### 4.5.4.2 c_offset

```
int64_t Column::c_offset  [private]
```

column offset in a table, if column table the value is 0

#### 4.5.4.3 c_size

```
int64_t Column::c_size  [private]
```

column size

#### 4.5.4.4 c_type

```
ColumnType Column::c_type  [private]
```

column type

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

- system/schema.h

## 4.6 Condition Struct Reference

`#include <executor.h>`

### Public Attributes

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

### 4.6.1 Detailed Description

definition of compare condition.

### 4.6.2 Member Data Documentation

#### 4.6.2.1 column

`RequestColumn Condition::column`

which column

#### 4.6.2.2 compare

`CompareMethod Condition::compare`

which method

#### 4.6.2.3 value

`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

## 4.7 Conditions Struct Reference

`#include <executor.h>`

### Public Attributes

- Condition condition [4]
- int condition_num

### 4.7.1 Detailed Description

definition of conditions.

### 4.7.2 Member Data Documentation

#### 4.7.2.1 condition

```
Condition Conditions::condition[4]
```

support maximum 4 & conditions

#### 4.7.2.2 condition_num

```
int Conditions::condition_num
```

number of condition in use

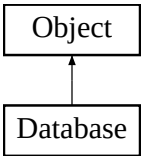
The documentation for this struct was generated from the following file:

- system/executor.h

## 4.8 Database Class Reference

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

Inheritance 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 *columns[])
- virtual bool insert (int64_t table_id, char *source)
- 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

### 4.8.1 Detailed Description

definition of class Database.

### 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 Database()

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

constructor.

**Parameters**

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

#### 4.8.2.2 ~Database()

```
virtual Database::~Database (
            void  )  [inline], [virtual]
```

destructor.

### 4.8.3 Member Function Documentation

#### 4.8.3.1 addTable()

```
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 |

#### 4.8.3.2 finish()

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

finish, important interface for son class

#### 4.8.3.3 getTables()

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

get table identifier container.

#### 4.8.3.4 init()

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

init, important interface for son class

#### 4.8.3.5 insert() [1/2]

```
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 |

#### 4.8.3.6 insert() [2/2]

```
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 |

#### 4.8.3.7 loadData()

```
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 |

#### 4.8.3.8 print()

```
virtual void Database::print (
            void  )  [inline], [virtual]
```

print database information

Reimplemented from Object.

#### 4.8.3.9 shut()

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

shut down this database, free all memory.

Reimplemented from Object.

### 4.8.4 Member Data Documentation

#### 4.8.4.1 d_table

```
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

## 4.9 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*

### 4.9.1 Detailed Description

an array of source file names

### 4.9.2 Constructor & Destructor Documentation

#### 4.9.2.1 ErrorLog()

```
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 |

#### 4.9.2.2 ~ErrorLog()

```
ErrorLog::~ErrorLog ( )
```

destructor

### 4.9.3 Member Function Documentation

#### 4.9.3.1 closeLog()

```
static void ErrorLog::closeLog (
            void  )  [inline], [static]
```

close the log file

#### 4.9.3.2 flushLog()

```
static void ErrorLog::flushLog (
            void  )  [inline], [static]
```

flush the log file

#### 4.9.3.3 getErrorCode()

```
int ErrorLog::getErrorCode (
            void  )  [inline]
```

return the last error code

#### 4.9.3.4 getErrorMsg()

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

return the error message accumulated since last reset()

#### 4.9.3.5 getFuncNameGCC()

```
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 |

**Return values**

| | |
|---|---|
| *-1* | function name is not available |

#### 4.9.3.6 id2Name()

```
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 |

#### 4.9.3.7 init()

```
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 |

#### 4.9.3.8 log()

```
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 |

#### 4.9.3.9 name2Id()

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

#### 4.9.3.10 reset()

```
void ErrorLog::reset ( )
```

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

#### 4.9.3.11 setLevel()

```
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 |

### 4.9.4 Member Data Documentation

#### 4.9.4.1 el_bt_buffer

```
void* ErrorLog::el_bt_buffer[256]  [private]
```

allow up to 256 levels of calls

#### 4.9.4.2 el_demangle_buf

```
char* ErrorLog::el_demangle_buf  [private]
```

buffer needed for demangle

#### 4.9.4.3 el_demangle_len

```
size_t ErrorLog::el_demangle_len  [private]
```

demangle buffer length

#### 4.9.4.4 el_err_code

```
int ErrorLog::el_err_code  [private]
```

current error code

#### 4.9.4.5 el_fp

```
FILE * ErrorLog::el_fp  [static], [private]
```

file handle of el_logfile

#### 4.9.4.6 el_level

```
int ErrorLog::el_level  [static]
```

logging level

#### 4.9.4.7 el_level_name

```
const char * ErrorLog::el_level_name  [static]
```

level => name

#### 4.9.4.8 el_lock

```
pthread_mutex_t ErrorLog::el_lock  [static], [private]
```

protect the global states

#### 4.9.4.9 el_logfile

```
char * ErrorLog::el_logfile  [static], [private]
```

file path to write log to

#### 4.9.4.10 el_msg_buf

```
char* ErrorLog::el_msg_buf  [private]
```

a buffer to hold the error message

#### 4.9.4.11 el_msg_cap

```
int ErrorLog::el_msg_cap  [private]
```

the message buffer size

#### 4.9.4.12 el_msg_cur

```
char* ErrorLog::el_msg_cur  [private]
```

point to the end of the message

#### 4.9.4.13 el_name_2_id

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

file name => id

#### 4.9.4.14 el_thread_name

```
char* ErrorLog::el_thread_name  [private]
```

the thread name

#### 4.9.4.15 el_tloc

```
time_t ErrorLog::el_tloc  [private]
```

local time in seconds at last message

#### 4.9.4.16 el_tm

```
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

## 4.10 Executor Class Reference

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

## Public Member Functions

- int close ()
- int exec (SelectQuery *query, ResultTable *result)
- int findCol (char *table_name, char *column_name)

  *find the rank of selected column in given table*
- int64_t getRank (std::vector< int64_t > &vec, int64_t id)
- Operator * planner (SelectQuery *query)

## Public Attributes

- Operator * root

## Private Attributes

- SelectQuery * current_query

### 4.10.1 Detailed Description

definition of class executor.

### 4.10.2 Member Function Documentation

#### 4.10.2.1 close()

```
int Executor::close ( )
```

close function.

**Parameters**

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

**Return values**

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

#### 4.10.2.2 exec()

```
int Executor::exec (
```

```
            SelectQuery * query,
            ResultTable * result )
```

exec function.

**Parameters**

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

#### 4.10.2.3 findCol()

```
int Executor::findCol (
            char * table_name,
            char * column_name )
```

find the rank of selected column in given table

**Parameters**

| | |
|---|---|
| *table_name* | the name of the given table |
| *column_name* | the name of the selected column |

**Returns**

the rank of the column in table

#### 4.10.2.4 getRank()

```
int64_t Executor::getRank (
            std::vector< int64_t > & vec,
            int64_t id )
```

for a certain id vector, find the rank of a given id

**Parameters**

| | |
|---|---|
| *vec* | the id vector to be searched through |
| *id* | the id to be searched |

#### 4.10.2.5 planner()

```
Operator * Executor::planner (
            SelectQuery * query )
```

generate operator tree and return root

**Parameters**

| | |
|---|---|
| *query* | the query to form an operator tree |

**Return values**

| | |
|---|---|
| *the* | root of the operator tree |

### 4.10.3 Member Data Documentation

#### 4.10.3.1 current_query

```
SelectQuery* Executor::current_query  [private]
```

selectquery to iterately execute

#### 4.10.3.2 root

```
Operator* Executor::root
```

the root of the operator tree

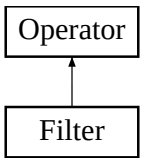
The documentation for this class was generated from the following files:

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

## 4.11 Filter Class Reference

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

Inheritance diagram for Filter:



## Public Member Functions

- Filter ()
- Filter (Operator *child, int64_t c_id[], int64_t num_column, int64_t filt_rank, CompareMethod cmp_mtd, char *value)
- Filter (Operator *child, std::vector< int64_t > input_cid, int64_t filt_rank, CompareMethod cmp_mtd, char *value)
- ~Filter ()
- bool close ()
- bool getNext ()
- bool open ()
- void setChild (Operator *child)
- bool setColumn (int64_t c_id[], int64_t num_column, int64_t filt_rank, CompareMethod cmp_mtd, char *value)
- bool setColumn (std::vector< int64_t > input_cid, int64_t filt_rank, CompareMethod cmp_mtd, char *value)

## Private Member Functions

- void initCmpFunc ()
- void setFiltCond (int64_t filt_rank, CompareMethod cmp_mtd, char *value)

## Static Private Member Functions

- static bool cmpEQ (void *data1, void *data2, BasicType *data_type)
- static bool cmpGE (void *data1, void *data2, BasicType *data_type)
- static bool cmpGT (void *data1, void *data2, BasicType *data_type)
- static bool cmpLE (void *data1, void *data2, BasicType *data_type)
- static bool cmpLT (void *data1, void *data2, BasicType *data_type)
- static bool cmpNE (void *data1, void *data2, BasicType *data_type)

## Private Attributes

- char * buf_for_child
- Operator * child
- int64_t child_buf_size
- bool(* cmp_func )(void *a, void *b, BasicType *data_type)
- CompareMethod cmp_mtd
- bool(* cmp_table [MAX_CM])(void *data1, void *data2, BasicType *data_type)
- int64_t filt_off
- char * filt_pos
- BasicType * filt_type
- int64_t in_tuple_size
- std::vector< int64_t > input_cid
- char value [128]

## Additional Inherited Members

### 4.11.1 Detailed Description

definition of the class Filter the filter operator

### 4.11.2 Constructor & Destructor Documentation

#### 4.11.2.1 Filter() [1/3]

```
Filter::Filter ( )  [inline]
```

Constructor

#### 4.11.2.2 ~Filter()

```
Filter::~Filter ( )  [inline]
```

Destructor

#### 4.11.2.3 Filter() [2/3]

```
Filter::Filter (
            Operator * child,
            std::vector< int64_t > input_cid,
            int64_t filt_rank,
            CompareMethod cmp_mtd,
            char * value )  [inline]
```

Overload constructor, designate some properties when created operator

**Parameters**

| | |
|---|---|
| *child* | child operator to set |
| *input_cid* | ID of input columns |
| *filt_rank* | rank of filtering column from input |
| *cmp_mtd* | compare method |
| *value* | text of value compared with |

#### 4.11.2.4 Filter() [3/3]

```
Filter::Filter (
            Operator * child,
            int64_t c_id[],
            int64_t num_column,
            int64_t filt_rank,
            CompareMethod cmp_mtd,
            char * value )  [inline]
```

Overload constructor

**Parameters**

| | |
|---|---|
| *child* | child operator to set |
| *c_id* | ID of input columns |
| *num_column* | number of input columns |
| *filt_rank* | rank of filtering column from input |
| *cmp_mtd* | compare method |
| *value* | text of value compared with |

### 4.11.3 Member Function Documentation

#### 4.11.3.1 close()

```
bool Filter::close ( )  [virtual]
```

close this operator, release all resources

**Return values**

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

Reimplemented from Operator.

#### 4.11.3.2 cmpEQ()

```
static bool Filter::cmpEQ (
            void * data1,
            void * data2,
            BasicType * data_type )  [inline], [static], [private]
```

Equal to

**Parameters**

| | |
|---|---|
| *data1* | first operand |
| *data2* | second operand |
| *data_type* | basic type of these two operands |

**Return values**

| | |
|---|---|
| *true* | data1 == data2 |
| *false* | otherwise |

#### 4.11.3.3 cmpGE()

```
static bool Filter::cmpGE (
            void * data1,
            void * data2,
            BasicType * data_type )  [inline], [static], [private]
```

Greater than or equal to

**Parameters**

| | |
|---|---|
| *data1* | first operand |
| *data2* | second operand |
| *data_type* | basic type of these two operands |

**Return values**

| | |
|---|---|
| *true* | data1 >= data2 |
| *false* | otherwise |

#### 4.11.3.4 cmpGT()

```
static bool Filter::cmpGT (
            void * data1,
            void * data2,
            BasicType * data_type )  [inline], [static], [private]
```

Greater than

**Parameters**

| | |
|---|---|
| *data1* | first operand |
| *data2* | second operand |
| *data_type* | basic type of these two operands |

**Return values**

| | |
|---|---|
| *true* | data1 > data2 |
| *false* | otherwise |

#### 4.11.3.5 cmpLE()

```
static bool Filter::cmpLE (
```

```
            void * data1,
            void * data2,
            BasicType * data_type )  [inline], [static], [private]
```

Less than or equal to

**Parameters**

| | |
|---|---|
| *data1* | first operand |
| *data2* | second operand |
| *data_type* | basic type of these two operands |

**Return values**

| | |
|---|---|
| *true* | data1 <= data2 |
| *false* | otherwise |

#### 4.11.3.6 cmpLT()

```
static bool Filter::cmpLT (
            void * data1,
            void * data2,
            BasicType * data_type )  [inline], [static], [private]
```

Comparision function for this class (wrapped method of BasicType) Less than

**Parameters**

| | |
|---|---|
| *data1* | first operand |
| *data2* | second operand |
| *data_type* | basic type of these two operands |

**Return values**

| | |
|---|---|
| *true* | data1 < data2 |
| *false* | otherwise |

#### 4.11.3.7 cmpNE()

```
static bool Filter::cmpNE (
            void * data1,
            void * data2,
            BasicType * data_type )  [inline], [static], [private]
```

Not equal to

**Parameters**

| | |
|---|---|
| *data1* | first operand |
| *data2* | second operand |
| *data_type* | basic type of these two operands |

**Return values**

| | |
|---|---|
| *true* | data1 != data2 |
| *false* | otherwise |

#### 4.11.3.8 getNext()

```
bool Filter::getNext ( )  [virtual]
```

get next tuple iterately

**Return values**

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

Reimplemented from Operator.

#### 4.11.3.9 initCmpFunc()

```
void Filter::initCmpFunc ( )  [inline], [private]
```

Initialize compare function

#### 4.11.3.10 open()

```
bool Filter::open ( )  [virtual]
```

open a Filter operator

**Return values**

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

Reimplemented from Operator.

#### 4.11.3.11 setChild()

```
void Filter::setChild (
            Operator * child )  [inline]
```

Set child operator, shoud be called before open

**Parameters**

| | |
|---|---|
| *child* | child operator |

#### 4.11.3.12 setColumn() [1/2]

```
bool Filter::setColumn (
            int64_t c_id[],
            int64_t num_column,
            int64_t filt_rank,
            CompareMethod cmp_mtd,
            char * value )  [inline]
```

Save input column for later use, should be called before open

**Parameters**

| | |
|---|---|
| *c_id* | the column id of input |
| *num_column* | number of columns in the input |
| *filt_rank* | the filter column is which column from the input |
| *cmp_mtd* | compare method |
| *value* | text of designated value |

#### 4.11.3.13 setColumn() [2/2]

```
bool Filter::setColumn (
            std::vector< int64_t > input_cid,
            int64_t filt_rank,
            CompareMethod cmp_mtd,
            char * value )  [inline]
```

Overload, set input_cid

**Parameters**

| | |
|---|---|
| *input_cid* | the column id of input |
| *filt_rank* | the filter column is which column from the input |
| *cmp_mtd* | compare method |
| *value* | text of designated value |

**Return values**

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

#### 4.11.3.14 setFiltCond()

```
void Filter::setFiltCond (
            int64_t filt_rank,
            CompareMethod cmp_mtd,
            char * value )  [private]
```

Set properties for filter

**Parameters**

| | |
|---|---|
| *filt_rank* | the filter column is which column from the input |
| *cmp_mtd* | compare method |
| *value* | text value to compare with |

### 4.11.4 Member Data Documentation

#### 4.11.4.1 buf_for_child

```
char* Filter::buf_for_child  [private]
```

child operator

#### 4.11.4.2 child

```
Operator* Filter::child  [private]
```

#### 4.11.4.3 child_buf_size

```
int64_t Filter::child_buf_size  [private]
```

buffer allocated for child

#### 4.11.4.4 cmp_func

```
bool(* Filter::cmp_func) (void *a, void *b, BasicType *data_type)  [private]
```

datatype of column to filter on

#### 4.11.4.5 cmp_mtd

```
CompareMethod Filter::cmp_mtd  [private]
```

offset of column to filter on (with respect to the input)

#### 4.11.4.6 cmp_table

```
bool(* Filter::cmp_table[MAX_CM])(void *data1, void *data2, BasicType *data_type)  [private]
```

save the binary value to compare with Jump table for comparison, data_type is used for dynamic binding

#### 4.11.4.7 filt_off

```
int64_t Filter::filt_off  [private]
```

position of column to filter on

#### 4.11.4.8 filt_pos

```
char* Filter::filt_pos  [private]
```

identifiers of columns to operate

#### 4.11.4.9 filt_type

```
BasicType* Filter::filt_type  [private]
```

store compare method

#### 4.11.4.10 in_tuple_size

```
int64_t Filter::in_tuple_size  [private]
```

compare function for filter

#### 4.11.4.11 input_cid

```
std::vector< int64_t > Filter::input_cid  [private]
```

size of buffer for child

#### 4.11.4.12 value

```
char Filter::value[128]  [private]
```

size of the input tuple

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

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

## 4.12 GrAggRecord Class Reference

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

### Public Member Functions

- GrAggRecord (char *middle_record, int64_t num_aggr)

### Public Attributes

- std::vector< int64_t > count
- char * middle_record
- std::vector< int64_t > sum

### 4.12.1 Detailed Description

Definition of class GrAggRecord an intermediate record of GroupbyAggre

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 GrAggRecord()

```
GrAggRecord::GrAggRecord (
            char * middle_record,
            int64_t num_aggr )  [inline]
```

Count of this class, use int64 to store Constructor

**Parameters**

| | |
|---|---|
| *middle_record* | initial middle_record |
| *num_aggr* | number of aggr, size of sum/count |

### 4.12.3 Member Data Documentation

#### 4.12.3.1 count

```
std::vector<int64_t> GrAggRecord::count
```

Sum of this class, use int64 to store

#### 4.12.3.2 middle_record

```
char* GrAggRecord::middle_record
```

#### 4.12.3.3 sum

```
std::vector<int64_t> GrAggRecord::sum
```

Points to a class of middle record

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

- system/executor.h

## 4.13 GroupbyAggre::group_by_hash Struct Reference

### Public Member Functions

- size_t operator() (const group_by_key_t &key) const

### 4.13.1 Member Function Documentation

#### 4.13.1.1 operator()()

```
size_t GroupbyAggre::group_by_hash::operator() (
            const group_by_key_t & key ) const  [inline]
```

A buffer used to compare a new record with existing record

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

- system/executor.h

## 4.14 GroupbyAggre::group_by_key Struct Reference

### Public Member Functions

- bool operator== (const group_by_key &k) const

### Public Attributes

- group_by_type_t type_array
- std::vector< char * > value_array

### 4.14.1 Detailed Description

Position of aggr columns

### 4.14.2 Member Function Documentation

#### 4.14.2.1 operator==()

```
bool GroupbyAggre::group_by_key::operator== (
            const group_by_key & k ) const  [inline]
```

### 4.14.3 Member Data Documentation

#### 4.14.3.1 type_array

```
group_by_type_t GroupbyAggre::group_by_key::type_array
```

#### 4.14.3.2 value_array

```
std::vector<char *> GroupbyAggre::group_by_key::value_array
```

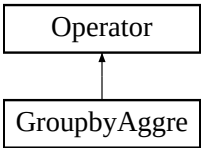
The documentation for this struct was generated from the following file:

- system/executor.h

## 4.15 GroupbyAggre Class Reference

`#include <executor.h>`

Inheritance diagram for GroupbyAggre:




### Classes

- struct group_by_hash
- struct group_by_key

### Public Member Functions

- GroupbyAggre ()
- ~GroupbyAggre ()
- bool close ()
- bool getNext ()
- bool open ()
- void set (std::vector< int64_t > input_colid, std::vector< int64_t > groupby_rank, std::vector< AggreCondition > conditions, std::vector< int64_t > output_colid)
- void setChild (Operator *child)

### Private Types

- typedef struct GroupbyAggre::group_by_hash group_by_hash_t
- typedef struct GroupbyAggre::group_by_key group_by_key_t
- typedef std::vector< BasicType * > group_by_type_t

### Static Private Member Functions

- static void avgFloat32 (void *sum, void *count, void *x)
- static void avgFloat64 (void *sum, void *count, void *x)
- static void avgInt16 (void *sum, void *count, void *x)
- static void avgInt32 (void *sum, void *count, void *x)
- static void avgInt64 (void *sum, void *count, void *x)
- static void avgInt8 (void *sum, void *count, void *x)
- static void count (void *sum, void *count, void *x)
- static void finalCount (void *sum, void *count, void *result)
- static void finalFloat32Avg (void *sum, void *count, void *result)
- static void finalFloat32Sum (void *sum, void *count, void *result)
- static void finalFloat64Avg (void *sum, void *count, void *result)
- static void finalFloat64Sum (void *sum, void *count, void *result)
- static void finalInt16Sum (void *sum, void *count, void *result)
- static void finalInt32Sum (void *sum, void *count, void *result)

- static void finalInt64Sum (void *sum, void *count, void *result)
- static void finalInt8Sum (void *sum, void *count, void *result)
- static void finalIntAvg (void *sum, void *count, void *result)
- static void initAvg (void *sum, void *count)
- static void initCount (void *sum, void *count)
- static void initFloat32Max (void *sum, void *count)
- static void initFloat32Min (void *sum, void *count)
- static void initFloat64Max (void *sum, void *count)
- static void initFloat64Min (void *sum, void *count)
- static void initInt16Max (void *sum, void *count)
- static void initInt16Min (void *sum, void *count)
- static void initInt32Max (void *sum, void *count)
- static void initInt32Min (void *sum, void *count)
- static void initInt64Max (void *sum, void *count)
- static void initInt64Min (void *sum, void *count)
- static void initInt8Max (void *sum, void *count)
- static void initInt8Min (void *sum, void *count)
- static void initSum (void *sum, void *count)
- static void maxFloat32 (void *sum, void *count, void *x)
- static void maxFloat64 (void *sum, void *count, void *x)
- static void maxInt16 (void *sum, void *count, void *x)
- static void maxInt32 (void *sum, void *count, void *x)
- static void maxInt64 (void *sum, void *count, void *x)
- static void maxInt8 (void *sum, void *count, void *x)
- static void minFloat32 (void *sum, void *count, void *x)
- static void minFloat64 (void *sum, void *count, void *x)
- static void minInt16 (void *sum, void *count, void *x)
- static void minInt32 (void *sum, void *count, void *x)
- static void minInt64 (void *sum, void *count, void *x)
- static void minInt8 (void *sum, void *count, void *x)
- static void sumFloat32 (void *sum, void *count, void *x)
- static void sumFloat64 (void *sum, void *count, void *x)
- static void sumInt16 (void *sum, void *count, void *x)
- static void sumInt32 (void *sum, void *count, void *x)
- static void sumInt64 (void *sum, void *count, void *x)
- static void sumInt8 (void *sum, void *count, void *x)

## Private Attributes

- void(* aggr_method [4])(void *sum, void *count, void *x)
- std::vector< char * > aggr_pos
- std::vector< BasicType * > aggr_type
- void(* avg_table [MAXTYPE_TC])(void *sum, void *count, void *x)
- char * buf_for_child
- Operator * child
- int64_t child_buf_size
- int64_t child_tuple_size
- std::vector< AggreCondition > conditions
- void(* final_avg_table [MAXTYPE_TC])(void *sum, void *count, void *result)
- void(* final_method [4])(void *sum, void *count, void *result)
- void(* final_sum_table [MAXTYPE_TC])(void *sum, void *count, void *result)
- std::vector< char * > group_by_pos
- std::vector< int64_t > group_by_size
- group_by_type_t group_by_type

- std::vector< int64_t > groupby_rank
- std::unordered_map< group_by_key_t, GrAggRecord *, group_by_hash_t > hash_group
- std::vector< int64_t > in_cid
- void(* init_max_table [MAXTYPE_TC])(void *sum, void *count)
- void(* init_method [4])(void *sum, void *count)
- void(* init_min_table [MAXTYPE_TC])(void *sum, void *count)
- void(* max_table [MAXTYPE_TC])(void *sum, void *count, void *x)
- std::vector< char * > middle_buf_array
- int64_t middle_buf_size
- int64_t middle_tuple_size
- void(* min_table [MAXTYPE_TC])(void *sum, void *count, void *x)
- std::unordered_map< group_by_key_t, GrAggRecord * >::iterator next_iter
- std::vector< int64_t > out_cid
- void(* sum_table [MAXTYPE_TC])(void *sum, void *count, void *x)

## Additional Inherited Members

### 4.15.1 Detailed Description

Definition of class GroupbyAggre the groupby-aggregation operator

### 4.15.2 Member Typedef Documentation

#### 4.15.2.1 group_by_hash_t

```
typedef struct GroupbyAggre::group_by_hash GroupbyAggre::group_by_hash_t  [private]
```

#### 4.15.2.2 group_by_key_t

```
typedef struct GroupbyAggre::group_by_key GroupbyAggre::group_by_key_t  [private]
```

Position of aggr columns

#### 4.15.2.3 group_by_type_t

```
typedef std::vector< BasicType * > GroupbyAggre::group_by_type_t  [private]
```

Each element points to an intermediate record

### 4.15.3 Constructor & Destructor Documentation

#### 4.15.3.1 GroupbyAggre()

```
GroupbyAggre::GroupbyAggre ( )  [inline]
```

Finalize method table Constructor Initialize some data structure

#### 4.15.3.2 ~GroupbyAggre()

```
GroupbyAggre::~GroupbyAggre ( )  [inline]
```

Destructor

### 4.15.4 Member Function Documentation

#### 4.15.4.1 avgFloat32()

```
static void GroupbyAggre::avgFloat32 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate AVG method for FLOAT32

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.2 avgFloat64()

```
static void GroupbyAggre::avgFloat64 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate AVG method for FLOAT64

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.3 avgInt16()

```
static void GroupbyAggre::avgInt16 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate AVG method for INT16

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.4 avgInt32()

```
static void GroupbyAggre::avgInt32 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate AVG method for INT32

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.5 avgInt64()

```
static void GroupbyAggre::avgInt64 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate AVG method for INT64

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.6 avgInt8()

```
static void GroupbyAggre::avgInt8 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate AVG method for INT8

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.7 close()

```
bool GroupbyAggre::close ( )  [virtual]
```

close this operator, release all resources

**Return values**

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

Reimplemented from Operator.

#### 4.15.4.8 count()

```
static void GroupbyAggre::count (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate COUNT method

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to original value of sum |

#### 4.15.4.9 finalCount()

```
static void GroupbyAggre::finalCount (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for COUNT method

**Parameters**

| | |
|---|---|
| *sum* | sum to be finalized |
| *count* | count to be finalized |
| *result* | place to save result |

#### 4.15.4.10 finalFloat32Avg()

```
static void GroupbyAggre::finalFloat32Avg (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for AVG method of FLOAT32

**Parameters**

| | |
|---|---|
| *sum* | sum to be finalized |
| *count* | count to be finalized |
| *result* | place to save result |

#### 4.15.4.11 finalFloat32Sum()

```
static void GroupbyAggre::finalFloat32Sum (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for SUM method of FLOAT32

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.12 finalFloat64Avg()

```
static void GroupbyAggre::finalFloat64Avg (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for AVG method of FLOAT64

**Parameters**

| | |
|---|---|
| *sum* | sum to be finalized |
| *count* | count to be finalized |
| *result* | place to save result |

#### 4.15.4.13 finalFloat64Sum()

```
static void GroupbyAggre::finalFloat64Sum (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for SUM method of FLOAT64

**Parameters**

| | |
|---|---|
| *sum* | sum to be finalized |
| *count* | count to be finalized |
| *result* | place to save result |

#### 4.15.4.14 finalInt16Sum()

```
static void GroupbyAggre::finalInt16Sum (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for SUM method of INT16

**Parameters**

| | |
|---|---|
| *sum* | sum to be finalized |
| *count* | count to be finalized |
| *result* | place to save result |

#### 4.15.4.15 finalInt32Sum()

```
static void GroupbyAggre::finalInt32Sum (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for SUM method of INT32

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.16 finalInt64Sum()

```
static void GroupbyAggre::finalInt64Sum (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for SUM method of INT64

**Parameters**

| | |
|---|---|
| *sum* | sum to be finalized |
| *count* | count to be finalized |
| *result* | place to save result |

#### 4.15.4.17 finalInt8Sum()

```
static void GroupbyAggre::finalInt8Sum (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Init method table Finalize for SUM method of INT8

**Parameters**

| | |
|---|---|
| *sum* | sum to be finalized |
| *count* | count to be finalized |
| *result* | place to save result |

#### 4.15.4.18 finalIntAvg()

```
static void GroupbyAggre::finalIntAvg (
            void * sum,
            void * count,
            void * result )  [inline], [static], [private]
```

Finalize for AVG method of INT

**Parameters**

| | |
|---|---|
| *sum* | sum to be finalized |
| *count* | count to be finalized |
| *result* | place to save result |

#### 4.15.4.19 getNext()

```
bool GroupbyAggre::getNext ( )  [virtual]
```

get next record from the table iterately

**Return values**

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

Reimplemented from Operator.

#### 4.15.4.20 initAvg()

```
static void GroupbyAggre::initAvg (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for AVG method, note that float 0.0 is encoded as 0

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.21 initCount()

```
static void GroupbyAggre::initCount (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for count method

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.22 initFloat32Max()

```
static void GroupbyAggre::initFloat32Max (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MAX method of FLOAT32

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.23 initFloat32Min()

```
static void GroupbyAggre::initFloat32Min (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MIN method of FLOAT32

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.24 initFloat64Max()

```
static void GroupbyAggre::initFloat64Max (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MAX method of FLOAT64

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.25 initFloat64Min()

```
static void GroupbyAggre::initFloat64Min (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MIN method of FLOAT64

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.26 initInt16Max()

```
static void GroupbyAggre::initInt16Max (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MAX method of INT16

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.27 initInt16Min()

```
static void GroupbyAggre::initInt16Min (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MIN method of INT16

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.28 initInt32Max()

```
static void GroupbyAggre::initInt32Max (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MAX method of INT32

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.29 initInt32Min()

```
static void GroupbyAggre::initInt32Min (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MIN method of INT32

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.30 initInt64Max()

```
static void GroupbyAggre::initInt64Max (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MAX method of INT64

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.31 initInt64Min()

```
static void GroupbyAggre::initInt64Min (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MIN method of INT64

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.32 initInt8Max()

```
static void GroupbyAggre::initInt8Max (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MAX method of INT8

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.33 initInt8Min()

```
static void GroupbyAggre::initInt8Min (
            void * sum,
            void * count )  [inline], [static], [private]
```

Init for MIN method of INT8

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.34 initSum()

```
static void GroupbyAggre::initSum (
            void * sum,
            void * count )  [inline], [static], [private]
```

Aggregate method table Init for SUM method, note that float 0.0 is encoded as 0

**Parameters**

| | |
|---|---|
| *sum* | sum to be initialized |
| *count* | count to be initialized |

#### 4.15.4.35 maxFloat32()

```
static void GroupbyAggre::maxFloat32 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MAX method for FLOAT32

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.36 maxFloat64()

```
static void GroupbyAggre::maxFloat64 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MAX method for FLOAT64

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.37 maxInt16()

```
static void GroupbyAggre::maxInt16 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MAX method for INT16

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.38 maxInt32()

```
static void GroupbyAggre::maxInt32 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MAX method for INT32

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.39 maxInt64()

```
static void GroupbyAggre::maxInt64 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MAX method for INT64

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.40 maxInt8()

```
static void GroupbyAggre::maxInt8 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MAX method for INT8

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.41 minFloat32()

```
static void GroupbyAggre::minFloat32 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MIN method for FLOAT32

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.42 minFloat64()

```
static void GroupbyAggre::minFloat64 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MIN method for FLOAT64

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.43 minInt16()

```
static void GroupbyAggre::minInt16 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MIN method for INT16

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.44 minInt32()

```
static void GroupbyAggre::minInt32 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MIN method for INT32

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.45 minInt64()

```
static void GroupbyAggre::minInt64 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MIN method for INT64

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.46 minInt8()

```
static void GroupbyAggre::minInt8 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate MIN method for INT8

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.47 open()

```
bool GroupbyAggre::open ( )  [virtual]
```

open a scan operator

**Return values**

| | |
|---|---|
| *true* | successfully opened |
| *false* | failed |

Reimplemented from Operator.

#### 4.15.4.48 set()

```
void GroupbyAggre::set (
            std::vector< int64_t > input_colid,
            std::vector< int64_t > groupby_rank,
            std::vector< AggreCondition > conditions,
            std::vector< int64_t > output_colid )  [inline]
```

Set input and output columns

**Parameters**

| | |
|---|---|
| *input_colid* | ID of input columns |
| *groupby_rank* | rank of group-by columns from input |
| *conditions* | aggregate method with its column |
| *output_colid* | ID of output columns |

#### 4.15.4.49 setChild()

```
void GroupbyAggre::setChild (
            Operator * child )  [inline]
```

Set child operator, shoud be called before open

**Parameters**

| | |
|---|---|
| *child* | child operator |

#### 4.15.4.50 sumFloat32()

```
static void GroupbyAggre::sumFloat32 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate SUM method for FLOAT32

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.51 sumFloat64()

```
static void GroupbyAggre::sumFloat64 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate SUM method for FLOAT64

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.52 sumInt16()

```
static void GroupbyAggre::sumInt16 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate SUM method for INT16

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.53 sumInt32()

```
static void GroupbyAggre::sumInt32 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate SUM method for INT32

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.54 sumInt64()

```
static void GroupbyAggre::sumInt64 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Aggregate SUM method for INT64

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

#### 4.15.4.55 sumInt8()

```
static void GroupbyAggre::sumInt8 (
            void * sum,
            void * count,
            void * x )  [inline], [static], [private]
```

Iterator for next record Aggregate SUM method for INT8

**Parameters**

| | |
|---|---|
| *sum* | pointed to original value of sum |
| *count* | pointed to original value of count |
| *x* | pointed to x to be added |

### 4.15.5 Member Data Documentation

#### 4.15.5.1 aggr_method

```
void(* GroupbyAggre::aggr_method[4])(void *sum, void *count, void *x)  [private]
```

#### 4.15.5.2 aggr_pos

```
std::vector< char * > GroupbyAggre::aggr_pos  [private]
```

Basic type of aggr columns

#### 4.15.5.3 aggr_type

```
std::vector< BasicType * > GroupbyAggre::aggr_type  [private]
```

Size of group-by columns

#### 4.15.5.4 avg_table

```
void(* GroupbyAggre::avg_table[MAXTYPE_TC])(void *sum, void *count, void *x)  [private]
```

Search table for AVG

#### 4.15.5.5 buf_for_child

```
char* GroupbyAggre::buf_for_child  [private]
```

Aggregate method with column

#### 4.15.5.6 child

```
Operator* GroupbyAggre::child  [private]
```

#### 4.15.5.7 child_buf_size

```
int64_t GroupbyAggre::child_buf_size  [private]
```

Buffer for child

#### 4.15.5.8 child_tuple_size

```
int64_t GroupbyAggre::child_tuple_size  [private]
```

Buffer size allocated for child

#### 4.15.5.9 conditions

```
std::vector<AggreCondition> GroupbyAggre::conditions  [private]
```

Output column ID

#### 4.15.5.10 final_avg_table

```
void(* GroupbyAggre::final_avg_table[MAXTYPE_TC])(void *sum, void *count, void *result)  [private]
```

Search table for AVG final

#### 4.15.5.11 final_method

```
void(* GroupbyAggre::final_method[4])(void *sum, void *count, void *result)  [private]
```

#### 4.15.5.12 final_sum_table

```
void(* GroupbyAggre::final_sum_table[MAXTYPE_TC])(void *sum, void *count, void *result)  [private]
```

Search table for SUM final

#### 4.15.5.13 group_by_pos

```
std::vector< char * > GroupbyAggre::group_by_pos  [private]
```

Basic type of group-by columns

#### 4.15.5.14 group_by_size

```
std::vector< int64_t > GroupbyAggre::group_by_size  [private]
```

Position of group-by columns

#### 4.15.5.15 group_by_type

```
group_by_type_t GroupbyAggre::group_by_type  [private]
```

#### 4.15.5.16 groupby_rank

```
std::vector<int64_t> GroupbyAggre::groupby_rank  [private]
```

Input column ID

#### 4.15.5.17 hash_group

```
std::unordered_map<group_by_key_t, GrAggRecord *, group_by_hash_t> GroupbyAggre::hash_group
[private]
```

#### 4.15.5.18 in_cid

```
std::vector<int64_t> GroupbyAggre::in_cid  [private]
```

Child operator

#### 4.15.5.19 init_max_table

```
void(* GroupbyAggre::init_max_table[MAXTYPE_TC])(void *sum, void *count)  [private]
```

Search table for MAX init

#### 4.15.5.20 init_method

```
void(* GroupbyAggre::init_method[4])(void *sum, void *count)  [private]
```

#### 4.15.5.21 init_min_table

```
void(* GroupbyAggre::init_min_table[MAXTYPE_TC])(void *sum, void *count)  [private]
```

Search table for MIN init

#### 4.15.5.22 max_table

```
void(* GroupbyAggre::max_table[MAXTYPE_TC])(void *sum, void *count, void *x)  [private]
```

Search table for MAX

#### 4.15.5.23 middle_buf_array

```
std::vector< char * > GroupbyAggre::middle_buf_array  [private]
```

Size of buffer allocated for intermediate record

#### 4.15.5.24 middle_buf_size

```
int64_t GroupbyAggre::middle_buf_size  [private]
```

Size of an intermediate record, not including c and s

#### 4.15.5.25 middle_tuple_size

```
int64_t GroupbyAggre::middle_tuple_size  [private]
```

Size of a tuple from child

#### 4.15.5.26 min_table

```
void(* GroupbyAggre::min_table[MAXTYPE_TC])(void *sum, void *count, void *x)  [private]
```

Search table for MIN

#### 4.15.5.27 next_iter

```
std::unordered_map<group_by_key_t,GrAggRecord*>::iterator GroupbyAggre::next_iter  [private]
```

Hash map for groupby

#### 4.15.5.28 out_cid

```
std::vector<int64_t> GroupbyAggre::out_cid  [private]
```

Rank of group-by columns

#### 4.15.5.29 sum_table

```
void(* GroupbyAggre::sum_table[MAXTYPE_TC])(void *sum, void *count, void *x)  [private]
```

Search table for SUM

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

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

## 4.16 HashCell Class Reference

```
#include <hashtable.h>
```

### Public Attributes

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

### 4.16.1 Detailed Description

definition of class HashCell.

### 4.16.2 Member Data Documentation

#### 4.16.2.1 capacity

```
int HashCell::capacity
```

maximun number of array in this structure

#### 4.16.2.2 ent

```
Hashcode_Ptr HashCell::ent
```

Hashcode_Ptr, hit,decrease cache miss

#### 4.16.2.3 ents

```
Hashcode_Ptr* HashCell::ents
```

pointer of Hashcode_Ptr array

#### 4.16.2.4 hc_num

```
int HashCell::hc_num
```

Hashcode_Ptr number in this HashCell

#### 4.16.2.5

```
union { ... } HashCell::hc_union
```

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

#### 4.16.2.6

```
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

## 4.17 Hashcode_Ptr Class Reference

```
#include <hashtable.h>
```

### Public Attributes

- int64_t hash_code
- char * tuple

### 4.17.1 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.

### 4.17.2 Member Data Documentation

#### 4.17.2.1 hash_code

```
int64_t Hashcode_Ptr::hash_code
```

hash_code of specific data

#### 4.17.2.2 tuple

```
char* Hashcode_Ptr::tuple
```

pointer of a record tuple

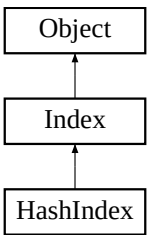
The documentation for this class was generated from the following file:

- system/hashtable.h

## 4.18 HashIndex Class Reference

```
#include <hashindex.h>
```

Inheritance diagram for HashIndex:



## Public Member Functions

- HashIndex (int64_t h_id, const char *i_name, Key &i_key)
- bool addIndexDTpye (BasicType *i_dt, int64_t offset)
- 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)
- void print (void)
- 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)
- 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
- int64_t * ih_table_offset

## Additional Inherited Members

### 4.18.1 Detailed Description

definition of HashIndex.

### 4.18.2 Constructor & Destructor Documentation

#### 4.18.2.1 HashIndex()

```
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 |

### 4.18.3 Member Function Documentation

#### 4.18.3.1 addIndexDTpye()

```
bool HashIndex::addIndexDTpye (
            BasicType * i_dt,
            int64_t offset )
```

add indexed column's data type.

**Parameters**

| | |
|---|---|
| *i_dt* | data type of indexed column |
| *offset* | offset of column in a rowtable |

#### 4.18.3.2 cmpEQ() [1/2]

```
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 |

#### 4.18.3.3 cmpEQ() [2/2]

```
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 |

#### 4.18.3.4 del() [1/2]

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

del an entry in hash index.

**Parameters**

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

**Return values**

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

Reimplemented from Index.

#### 4.18.3.5 del() [2/2]

```
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.

#### 4.18.3.6 finish()

```
bool HashIndex::finish (
            void  )  [virtual]
```

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

**Return values**

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

Reimplemented from Index.

#### 4.18.3.7 hash()

```
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 |

#### 4.18.3.8 init()

```
bool HashIndex::init (
            void  )  [virtual]
```

init hashindex, to calculate initial value.

**Return values**

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

Reimplemented from Index.

#### 4.18.3.9 insert() [1/2]

```
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.

#### 4.18.3.10 insert() [2/2]

```
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.

#### 4.18.3.11 lookup() [1/2]

```
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.

#### 4.18.3.12 lookup() [2/2]

```
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.

#### 4.18.3.13 print()

```
void HashIndex::print (
            void  )  [virtual]
```

print hash index information.

Reimplemented from Index.

#### 4.18.3.14 set_ls() [1/2]

```
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.

#### 4.18.3.15 set_ls() [2/2]

```
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.

#### 4.18.3.16 setCellCap()

```
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 |

#### 4.18.3.17 shut()

```
bool HashIndex::shut (
            void  )  [virtual]
```

free memory of hash table and other strucure.

**Return values**

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

Reimplemented from Index.

#### 4.18.3.18 tranToInt64() [1/2]

```
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.

#### 4.18.3.19 tranToInt64() [2/2]

```
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.

### 4.18.4 Member Data Documentation

#### 4.18.4.1 ih_cell_capbits

```
int64_t HashIndex::ih_cell_capbits  [private]
```

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

#### 4.18.4.2 ih_column_cap

```
int64_t HashIndex::ih_column_cap  [private]
```

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

#### 4.18.4.3 ih_column_num

```
int64_t HashIndex::ih_column_num  [private]
```

current number of added columns

#### 4.18.4.4 ih_datatype

```
BasicType** HashIndex::ih_datatype  [private]
```

each column data type

#### 4.18.4.5 ih_hash_bits

```
int64_t* HashIndex::ih_hash_bits  [private]
```

each column assigend bits when hashing

#### 4.18.4.6 ih_hashtable

```
HashTable* HashIndex::ih_hashtable  [private]
```

main part hash table

#### 4.18.4.7 ih_table_offset

```
int64_t* HashIndex::ih_table_offset  [private]
```

offsets of column key in rowtable

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

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

## 4.19 HashInfo Struct Reference

```
#include <hashindex.h>
```

## Public Attributes

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

### 4.19.1 Detailed Description

definition of HashInfo.

### 4.19.2 Member Data Documentation

#### 4.19.2.1 hash

```
int64_t HashInfo::hash
```

hashcell position in HashTable

#### 4.19.2.2 last

```
int HashInfo::last
```

retval of HashTable lookup

#### 4.19.2.3 ppos

```
int HashInfo::ppos
```

pair with last, |last|

#### 4.19.2.4 result

```
char* HashInfo::result[HASHINFO_CAPICITY]
```

buffer for value of void *pointer

#### 4.19.2.5 rnum

```
int HashInfo::rnum
```

current result number

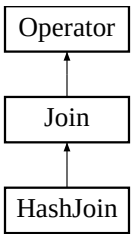
The documentation for this struct was generated from the following file:

- system/hashindex.h

## 4.20 HashJoin Class Reference

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

Inheritance diagram for HashJoin:




### Public Member Functions

- HashJoin ()
- HashJoin (std::vector< int64_t > left_cid, std::vector< int64_t > right_cid, int64_t left_rank, int64_t right_rank)
- ~HashJoin ()
- bool close ()
- bool getNext ()
- bool open ()

### Private Attributes

- std::multimap< std::string, char * > hash_index
- std::multimap< std::string, char * >::iterator last_iter
- char * left_buf
- int64_t left_key_off
- BasicType * left_key_type
- int64_t left_tuple_size
- std::vector< char * > middle_buf_array
- int64_t middle_buf_size
- char * right_buf
- int64_t right_buf_size
- bool right_has_next
- char * right_key_pos
- BasicType * right_key_type
- int64_t right_tuple_size
- char txt_buf [128]
- std::multimap< std::string, char * >::iterator upper_iter

## Additional Inherited Members

### 4.20.1 Detailed Description

Definition of class HashJoin the simple `HashJoin` operator with arbitrary left and right

### 4.20.2 Constructor & Destructor Documentation

#### 4.20.2.1 HashJoin() [1/2]

```
HashJoin::HashJoin ( )  [inline]
```

Upper bound of iterator for current key Constructor

#### 4.20.2.2 ~HashJoin()

```
HashJoin::~HashJoin ( )  [inline]
```

Destructor

#### 4.20.2.3 HashJoin() [2/2]

```
HashJoin::HashJoin (
            std::vector< int64_t > left_cid,
            std::vector< int64_t > right_cid,
            int64_t left_rank,
            int64_t right_rank )  [inline]
```

Overload constructor, set some attributes if already known

**Parameters**

| | |
|---|---|
| *left_cid* | column ID from left |
| *right_cid* | column ID from right |
| *left_rank* | rank of join key from left |
| *right_rank* | rank of join key from right |

### 4.20.3 Member Function Documentation

#### 4.20.3.1 close()

```
bool HashJoin::close ( )  [virtual]
```

Close this operator

**Return values**

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

Implements Join.

#### 4.20.3.2 getNext()

```
bool HashJoin::getNext ( )  [virtual]
```

Generate next record

**Return values**

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

Implements Join.

#### 4.20.3.3 open()

```
bool HashJoin::open ( )  [virtual]
```

Open a HashJoin operator

**Return values**

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

Implements Join.

### 4.20.4 Member Data Documentation

#### 4.20.4.1 hash_index

```
std::multimap<std::string, char *> HashJoin::hash_index  [private]
```

A buffer to receive text value

#### 4.20.4.2 last_iter

```
std::multimap<std::string,char*>::iterator HashJoin::last_iter  [private]
```

Newly created hash index, will not be added to catalog

#### 4.20.4.3 left_buf

```
char* HashJoin::left_buf  [private]
```

#### 4.20.4.4 left_key_off

```
int64_t HashJoin::left_key_off  [private]
```

Position for join key from the right

#### 4.20.4.5 left_key_type

```
BasicType* HashJoin::left_key_type  [private]
```

Basic type of join key on the right

#### 4.20.4.6 left_tuple_size

```
int64_t HashJoin::left_tuple_size  [private]
```

Size of each middle buffer

#### 4.20.4.7 middle_buf_array

```
std::vector<char *> HashJoin::middle_buf_array  [private]
```

Buffer allocated for right child

#### 4.20.4.8 middle_buf_size

```
int64_t HashJoin::middle_buf_size  [private]
```

Each element points to an intermidiate record

#### 4.20.4.9 right_buf

```
char* HashJoin::right_buf  [private]
```

Points to current area for left child to write

#### 4.20.4.10 right_buf_size

```
int64_t HashJoin::right_buf_size  [private]
```

Size of each record from right

#### 4.20.4.11 right_has_next

```
bool HashJoin::right_has_next  [private]
```

Size of buffer allocated for right

#### 4.20.4.12 right_key_pos

```
char* HashJoin::right_key_pos  [private]
```

Whether read to the end of right input

#### 4.20.4.13 right_key_type

```
BasicType* HashJoin::right_key_type  [private]
```

Offset for join key from the left

#### 4.20.4.14 right_tuple_size

```
int64_t HashJoin::right_tuple_size  [private]
```

Size of each record from left

#### 4.20.4.15 txt_buf

```
char HashJoin::txt_buf[128]  [private]
```

Basic type of join key on the left

#### 4.20.4.16 upper_iter

```
std::multimap<std::string,char*>::iterator HashJoin::upper_iter  [private]
```

Last time iterator when searching from the left

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

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

## 4.21 HashTable Class Reference

```
#include <hashtable.h>
```

### 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

### 4.21.1 Detailed Description

definition of class HashTable.

### 4.21.2 Constructor & Destructor Documentation

#### 4.21.2.1 HashTable()

```
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 |

#### 4.21.2.2 ~HashTable()

```
HashTable::~HashTable ( )
```

destructor, free HashTable memory to g_memory.

### 4.21.3 Member Function Documentation

#### 4.21.3.1 add()

```
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 |

#### 4.21.3.2 allocate()

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

mymemory alloc interface like malloc

#### 4.21.3.3 del()

```
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 |

#### 4.21.3.4 free()

```
void HashTable::free (
            void * mem )  [private]
```

mymemory free interface like free in stdlib

#### 4.21.3.5 probe()

```
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 |

#### 4.21.3.6 probe_contd()

```
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 |

#### 4.21.3.7 show()

```
void HashTable::show ( )
```

display data in this hash table, for debug use.

#### 4.21.3.8 size_to_slot()

```
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

#### 4.21.3.9 utilization()

```
void HashTable::utilization ( )
```

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

### 4.21.4 Member Data Documentation

#### 4.21.4.1 avail

```
Hashcode_Ptr* HashTable::avail
```

pointer of next available Hashcode_Ptr

#### 4.21.4.2 begin

```
char* HashTable::begin
```

start pointer of HashCells

#### 4.21.4.3 end

```
Hashcode_Ptr* HashTable::end
```

the end pointer of Hashcode_Ptr in array

#### 4.21.4.4 estimated_duplicates_per_key

```
double HashTable::estimated_duplicates_per_key
```

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

#### 4.21.4.5 estimated_num_distinct_keys

```
int HashTable::estimated_num_distinct_keys
```

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

#### 4.21.4.6 free_header

```
Hashcode_Ptr* HashTable::free_header[16]
```

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

#### 4.21.4.7 initial_array_size

```
int HashTable::initial_array_size
```

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

#### 4.21.4.8 more_allocated

```
int HashTable::more_allocated
```

analysis of more memory allocated from g_memory

#### 4.21.4.9 pointer2size

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

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

#### 4.21.4.10 table

```
HashCell* HashTable::table
```

pointer of an array of HashCell

#### 4.21.4.11 table_size

```
int HashTable::table_size
```

the number of HashCells in this table

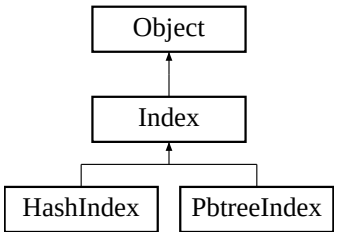
The documentation for this class was generated from the following files:

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

## 4.22 Index Class Reference

`#include <schema.h>`

Inheritance 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, void *p_del)
- virtual bool del (void *i_data[ ])
- virtual bool del (void *i_data[ ], void *p_del)
- virtual bool finish (void)
- Key & getIKey (void)
- virtual int64_t getIndexTid (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 *info, void *&result)
- virtual bool lookup (void *i_data[ ], void *&result)
- virtual bool lookup (void *i_data[ ], void *info, void *&result)
- virtual void print (void)
- virtual bool scan (void *info, void *&result)
- 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 void setIndexTid (int64_t tid)
- 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)

### Protected Attributes

- Key i_key
- int64_t i_t_id
- IndexType i_type

### 4.22.1 Detailed Description

definition of class Index

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 Index()

```
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 |

#### 4.22.2.2 ~Index()

```
virtual Index::~Index (
            void  )  [inline], [virtual]
```

destructor.

### 4.22.3 Member Function Documentation

#### 4.22.3.1 del() [1/4]

```
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.

#### 4.22.3.2 del() [2/4]

```
virtual bool Index::del (
            void * i_data,
            void * p_del )  [inline], [virtual]
```

del an entry in Index.

**Parameters**

| | |
|---|---|
| *i_data* | char buffer to store data in pattern |
| *p_del* | address of specified row |

**Return values**

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

Reimplemented in PbtreeIndex.

#### 4.22.3.3 del() [3/4]

```
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.

#### 4.22.3.4 del() [4/4]

```
virtual bool Index::del (
            void * i_data[],
            void * p_del )  [inline], [virtual]
```

del an entry in BptreeIndex.

**Parameters**

| | |
|---|---|
| *i_data* | each element of the array stores a pointer to column key |
| *p_del* | address of specified row |

**Return values**

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

#### 4.22.3.5 finish()

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

finish index, important interface for son class

Reimplemented in HashIndex.

#### 4.22.3.6 getIKey()

```
Key & Index::getIKey (
            void  )  [inline]
```

get index key

#### 4.22.3.7 getIndexTid()

```
virtual int64_t Index::getIndexTid (
            void  )  [inline], [virtual]
```

get index in which table

#### 4.22.3.8 getIType()

```
IndexType Index::getIType (
            void  )  [inline]
```

get index type

#### 4.22.3.9 init()

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

init index, important interface for son class

Reimplemented in HashIndex, and PbtreeIndex.

#### 4.22.3.10 insert() [1/2]

```
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, and PbtreeIndex.

#### 4.22.3.11 insert() [2/2]

```
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.

#### 4.22.3.12 lookup() [1/4]

```
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 |

#### 4.22.3.13 lookup() [2/4]

```
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, and PbtreeIndex.

#### 4.22.3.14 lookup() [3/4]

```
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 |

#### 4.22.3.15 lookup() [4/4]

```
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.

#### 4.22.3.16 print()

```
virtual void Index::print (
            void  )  [inline], [virtual]
```

print index information

Reimplemented from Object.

Reimplemented in HashIndex, and PbtreeIndex.

#### 4.22.3.17 scan()

```
virtual bool Index::scan (
            void * info,
            void *& result )  [inline], [virtual]
```

iterate on calling to scan for values.

**Parameters**

| | |
|---|---|
| *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 PbtreeIndex.

#### 4.22.3.18 scan_1() [1/2]

```
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 |

#### 4.22.3.19 scan_1() [2/2]

```
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 |

#### 4.22.3.20 scan_2() [1/2]

```
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 |

#### 4.22.3.21 scan_2() [2/2]

```
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 |

#### 4.22.3.22 set_ls() [1/2]

```
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, and PbtreeIndex.

#### 4.22.3.23 set_ls() [2/2]

```
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.

#### 4.22.3.24 setIndexTid()

```
virtual void Index::setIndexTid (
            int64_t tid )  [inline], [virtual]
```

set index in which table

#### 4.22.3.25 shut()

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

shut dowm the index, free memory.

Reimplemented from Object.

Reimplemented in HashIndex, and PbtreeIndex.

#### 4.22.3.26 tranToInt64() [1/2]

```
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.

#### 4.22.3.27 tranToInt64() [2/2]

```
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.

#### 4.22.3.28 update() [1/2]

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

#### 4.22.3.29 update() [2/2]

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

### 4.22.4 Member Data Documentation

#### 4.22.4.1 i_key

```
Key Index::i_key  [protected]
```

index keys

#### 4.22.4.2 i_t_id

```
int64_t Index::i_t_id  [protected]
```

in which table

#### 4.22.4.3 i_type

```
IndexType Index::i_type  [protected]
```

index type

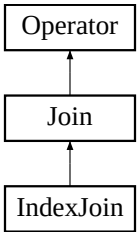
The documentation for this class was generated from the following file:

- system/schema.h

## 4.23 IndexJoin Class Reference

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

Inheritance diagram for IndexJoin:




### Public Member Functions

- IndexJoin ()
- IndexJoin (std::vector< int64_t > left_cid, std::vector< int64_t > right_cid, int64_t left_rank, int64_t right_rank)
- ~IndexJoin ()
- bool close ()
- bool getNext ()
- bool open ()

### Private Attributes

- void * current_key
- char * left_buf
- int64_t left_buf_size
- int64_t left_tuple_size
- char * right_buf
- int64_t right_buf_size
- bool right_has_next
- int64_t right_tuple_size

### Additional Inherited Members

## 4.23.1 Detailed Description

Definition of the class IndexJoin, the `IndexNestedLoopJoin` operator

## 4.23.2 Constructor & Destructor Documentation

#### 4.23.2.1 IndexJoin() [1/2]

```
IndexJoin::IndexJoin ( )  [inline]
```

Whether read to the end of right input Constructor

#### 4.23.2.2 ~IndexJoin()

```
IndexJoin::~IndexJoin ( )  [inline]
```

Destructor

#### 4.23.2.3 IndexJoin() [2/2]

```
IndexJoin::IndexJoin (
            std::vector< int64_t > left_cid,
            std::vector< int64_t > right_cid,
            int64_t left_rank,
            int64_t right_rank )  [inline]
```

Overload constructor, set some attributes if already known

**Parameters**

| | |
|---|---|
| *left_cid* | column ID from left |
| *right_cid* | column ID from left |
| *left_rank* | rank of join key from left |
| *right_rank* | rank of join key from right |

### 4.23.3 Member Function Documentation

#### 4.23.3.1 close()

```
bool IndexJoin::close ( )  [virtual]
```

Close this operator

**Return values**

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

Implements Join.

#### 4.23.3.2 getNext()

```
bool IndexJoin::getNext ( )  [virtual]
```

Generate next record

**Return values**

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

Implements Join.

#### 4.23.3.3 open()

```
bool IndexJoin::open ( )  [virtual]
```

Open a IndexJoin operator

**Return values**

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

Implements Join.

### 4.23.4 Member Data Documentation

#### 4.23.4.1 current_key

```
void* IndexJoin::current_key  [private]
```

Right buffer size Note that we only support single-key index here Keeps static after opening

#### 4.23.4.2 left_buf

```
char* IndexJoin::left_buf  [private]
```

Size of each record from right

#### 4.23.4.3 left_buf_size

```
int64_t IndexJoin::left_buf_size  [private]
```

Buffer for right child

#### 4.23.4.4 left_tuple_size

```
int64_t IndexJoin::left_tuple_size  [private]
```

#### 4.23.4.5 right_buf

```
char* IndexJoin::right_buf  [private]
```

Buffer allocated for left child

#### 4.23.4.6 right_buf_size

```
int64_t IndexJoin::right_buf_size  [private]
```

Left buffer size

#### 4.23.4.7 right_has_next

```
bool IndexJoin::right_has_next  [private]
```

Current key for searching, that is, data in join key

#### 4.23.4.8 right_tuple_size

```
int64_t IndexJoin::right_tuple_size  [private]
```

Size of each record from left

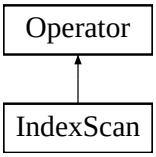
The documentation for this class was generated from the following files:

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

## 4.24 IndexScan Class Reference

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

Inheritance diagram for IndexScan:




### Public Member Functions

- IndexScan ()
- IndexScan (Table *table, Index *index)
- ~IndexScan ()
- bool close ()
- bool getNext ()
- bool open ()
- void setTabIdx (Table *table, Index *index)
- void updateKey (void *search_key)

### Private Attributes

- void * current_key
- Table * from
- IndexType i_type
- Index * index
- void * info_ptr
- bool key_end

### Additional Inherited Members

### 4.24.1 Detailed Description

Definition of class IndexScan

## 4.24.2 Constructor & Destructor Documentation

### 4.24.2.1 IndexScan() [1/2]

```
IndexScan::IndexScan ( )  [inline]
```

Empty constructor, used together with `setTabIdx`

### 4.24.2.2 IndexScan() [2/2]

```
IndexScan::IndexScan (
            Table * table,
            Index * index )  [inline]
```

Constructor, set table and index at the same time

**Parameters**

| | |
|---|---|
| *table* | table to scan |
| *index* | index to use when scanning |

### 4.24.2.3 ~IndexScan()

```
IndexScan::~IndexScan ( )  [inline]
```

Destructor

## 4.24.3 Member Function Documentation

### 4.24.3.1 close()

```
bool IndexScan::close ( )  [virtual]
```

Close this operator

**Return values**

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

Reimplemented from Operator.

#### 4.24.3.2 getNext()

```
bool IndexScan::getNext ( )  [virtual]
```

Get next record from operator

**Return values**

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

Reimplemented from Operator.

#### 4.24.3.3 open()

```
bool IndexScan::open ( )  [virtual]
```

Open an IndexScan operator

**Return values**

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

Reimplemented from Operator.

#### 4.24.3.4 setTabIdx()

```
void IndexScan::setTabIdx (
            Table * table,
            Index * index )  [inline]
```

Searched to the end of current key Set designated table and index

**Parameters**

| | |
|---|---|
| *table* | table to scan |
| *index* | index to use when scanning |

#### 4.24.3.5 updateKey()

```
void IndexScan::updateKey (
            void * search_key )
```

Update search key for IndexScan Please use it before getNext

**Parameters**

| | |
|---|---|
| *search_key* | new key to use |

### 4.24.4 Member Data Documentation

#### 4.24.4.1 current_key

```
void* IndexScan::current_key  [private]
```

Pointer of info, general type

#### 4.24.4.2 from

```
Table* IndexScan::from  [private]
```

#### 4.24.4.3 i_type

```
IndexType IndexScan::i_type  [private]
```

Use which index

#### 4.24.4.4 index

```
Index* IndexScan::index  [private]
```

From which table

#### 4.24.4.5 info_ptr

```
void* IndexScan::info_ptr  [private]
```

Type of index

#### 4.24.4.6 key_end

```
bool IndexScan::key_end  [private]
```

Current key for searching

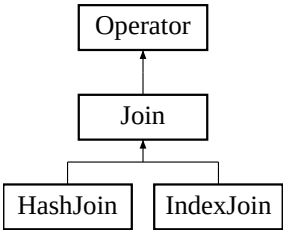
The documentation for this class was generated from the following files:

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

## 4.25 Join Class Reference

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

Inheritance diagram for Join:




### Public Member Functions

- Join ()
- virtual ~Join ()
- virtual bool close ()=0
- std::vector< int64_t > & getLeftCol ()
- Operator *& getLeftOp ()
- int64_t getLeftRank ()
- virtual bool getNext ()=0
- std::vector< int64_t > & getRightCol ()
- Operator *& getRightOp ()
- int64_t getRightRank ()
- virtual bool open ()=0
- void setJoinCol (std::vector< int64_t > left_cid, std::vector< int64_t > right_cid, int64_t left_rank, int64_t right_rank)
- void setLeftOp (Operator *lchild)
- void setRightOp (Operator *rchild)

### Private Attributes

- Operator * left
- std::vector< int64_t > left_cid
- int64_t left_rank
- Operator * right
- std::vector< int64_t > right_cid
- int64_t right_rank

## Additional Inherited Members

### 4.25.1 Detailed Description

Definition of Join operator

### 4.25.2 Constructor & Destructor Documentation

#### 4.25.2.1 Join()

```
Join::Join ( )  [inline]
```

Join Key is which column from the right child Constructor

#### 4.25.2.2 ~Join()

```
virtual Join::~Join ( )  [inline], [virtual]
```

Destructor

### 4.25.3 Member Function Documentation

#### 4.25.3.1 close()

```
virtual bool Join::close ( )  [pure virtual]
```

Close this operator

**Return values**

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

Reimplemented from Operator.

Implemented in IndexJoin, and HashJoin.

#### 4.25.3.2 getLeftCol()

```
std::vector< int64_t > & Join::getLeftCol ( )  [inline]
```

Get left cid

**Return values**

| | |
|---|---|
| *left_cid* | |

#### 4.25.3.3 getLeftOp()

```
Operator *& Join::getLeftOp ( )  [inline]
```

Get left child operator

**Return values**

| | |
|---|---|
| *reference* | of left child |

#### 4.25.3.4 getLeftRank()

```
int64_t Join::getLeftRank ( )  [inline]
```

Get left rank

**Return values**

| | |
|---|---|
| *left_rank* | |

#### 4.25.3.5 getNext()

```
virtual bool Join::getNext ( )  [pure virtual]
```

Generate next record

**Return values**

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

Reimplemented from Operator.

Implemented in IndexJoin, and HashJoin.

#### 4.25.3.6 getRightCol()

```
std::vector< int64_t > & Join::getRightCol ( )  [inline]
```

Get right cid

**Return values**

| | |
|---|---|
| *right_cid* | |

#### 4.25.3.7 getRightOp()

```
Operator *& Join::getRightOp ( )  [inline]
```

Get right child operator

**Return values**

| | |
|---|---|
| *reference* | of right child |

#### 4.25.3.8 getRightRank()

```
int64_t Join::getRightRank ( )  [inline]
```

Get right rank

**Return values**

| | |
|---|---|
| *right_rank* | |

#### 4.25.3.9 open()

```
virtual bool Join::open ( )  [pure virtual]
```

Open a Join operator, IndexJoin or HashJoin

**Return values**

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

Reimplemented from Operator.

Implemented in IndexJoin, and HashJoin.

#### 4.25.3.10 setJoinCol()

```
void Join::setJoinCol (
            std::vector< int64_t > left_cid,
            std::vector< int64_t > right_cid,
            int64_t left_rank,
            int64_t right_rank )  [inline]
```

Set input column arrangement

**Parameters**

| | |
|---|---|
| *left_cid* | column ID of left input |
| *right_cid* | column ID of right input |
| *left_rank* | left rank to set |
| *right_rank* | right rank to set |

#### 4.25.3.11 setLeftOp()

```
void Join::setLeftOp (
            Operator * lchild )  [inline]
```

Set left child operator

**Parameters**

| | |
|---|---|
| *lchild* | operator to set |

#### 4.25.3.12 setRightOp()

```
void Join::setRightOp (
            Operator * rchild )  [inline]
```

Set right child operator

**Parameters**

| | |
|---|---|
| *rchild* | operator to set |

### 4.25.4 Member Data Documentation

#### 4.25.4.1 left

```
Operator* Join::left  [private]
```

#### 4.25.4.2 left_cid

```
std::vector< int64_t > Join::left_cid  [private]
```

Right child operator

#### 4.25.4.3 left_rank

```
int64_t Join::left_rank  [private]
```

Input column ID of right child

#### 4.25.4.4 right

```
Operator* Join::right  [private]
```

Left child operator, can only be scan

#### 4.25.4.5 right_cid

```
std::vector< int64_t > Join::right_cid  [private]
```

Input column ID of left child

#### 4.25.4.6 right_rank

```
int64_t Join::right_rank  [private]
```

Join Key is which column from the left child

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

- system/executor.h

## 4.26 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

### 4.26.1 Detailed Description

definition of class Key.

### 4.26.2 Constructor & Destructor Documentation

#### 4.26.2.1 Key()

```
Key::Key (
            void  )  [inline]
```

constructor.

### 4.26.3 Member Function Documentation

#### 4.26.3.1 contain()

```
bool Key::contain (
            int64_t col_id )  [inline]
```

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

**Return values**

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

#### 4.26.3.2 getKey()

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

get key data.

#### 4.26.3.3 operator=()

```
Key & Key::operator= (
            const Key & p )  [inline]
```

over write operator(=).

#### 4.26.3.4 print()

```
void Key::print (
            void  )  [inline]
```

print key information.

#### 4.26.3.5 set()

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

set key.

**Parameters**

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

### 4.26.4 Member Data Documentation

#### 4.26.4.1 key

```
std::vector< int64_t > Key::key  [private]
```

index iidentifier container

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

- system/schema.h

## 4.27 Memory Class Reference

```
#include <mymemory.h>
```

### Public Member Functions

- int64_t alloc (char *&p, int64_t size)
- int allocTableAddr (char *&p)
- 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_table_addr
- char * m_tail
- int64_t m_total

### 4.27.1 Member Function Documentation

#### 4.27.1.1 alloc()

```
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 |

#### 4.27.1.2 alloc_default()

```
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 |

#### 4.27.1.3 allocTableAddr()

```
int Memory::allocTableAddr (
            char *& p )
```

alloc table address

**Parameters**

| | |
|---|---|
| *p* | the pointer of memory to free |

**Return values**

| | |
|---|---|
| *0* | successfuly free to db memory |

#### 4.27.1.4 free()

```
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 |

#### 4.27.1.5 init()

```
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 |

#### 4.27.1.6 print()

```
int Memory::print (
            void  )
```

print memory usage.

#### 4.27.1.7 shut()

```
int Memory::shut (
            void  )
```

free db memory to operate system.

#### 4.27.1.8 slot()

```
unsigned int Memory::slot (
            int64_t size )  [private]
```

calculate position of free list.

### 4.27.2 Member Data Documentation

#### 4.27.2.1 m_array_list

```
char** Memory::m_array_list  [private]
```

free arrray list

#### 4.27.2.2 m_curr

```
char* Memory::m_curr  [private]
```

pointer of db memory already in use

#### 4.27.2.3 m_head

```
char* Memory::m_head  [private]
```

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

#### 4.27.2.4 m_mins

```
int64_t Memory::m_mins  [private]
```

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

#### 4.27.2.5 m_table_addr

```
char* Memory::m_table_addr  [private]
```

current mmap table addr usage

#### 4.27.2.6 m_tail

```
char* Memory::m_tail  [private]
```

end pointer of db memory allocated from operate system

#### 4.27.2.7 m_total

```
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

## 4.28 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)
- void shut (void)

### Private Member Functions

- bool expand (void)

### Private Attributes

- char * ms_memory
- char * ms_memory_cur
- int64_t ms_memory_size
- int64_t ms_record_max
- int64_t ms_record_num
- int64_t ms_record_size
- char pad [128-4 *sizeof(int64_t) -sizeof(void *)]

### 4.28.1 Detailed Description

definition of MStorage, table storage manager.

## 4.28.2 Member Function Documentation

### 4.28.2.1 allocRow()

```
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 |

### 4.28.2.2 expand()

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

expand slots for more storage avaliable for this table.

**Return values**

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

### 4.28.2.3 getRecordNum()

```
int64_t MStorage::getRecordNum (
            void  )  [inline]
```

get the last record rank till now.

### 4.28.2.4 getRow()

```
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 |

#### 4.28.2.5 init()

```
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 |

#### 4.28.2.6 shut()

```
void MStorage::shut (
            void  )  [inline]
```

shut down, free memory to system.

### 4.28.3 Member Data Documentation

#### 4.28.3.1 ms_memory

```
char* MStorage::ms_memory  [private]
```

memory address

#### 4.28.3.2 ms_memory_cur

```
char* MStorage::ms_memory_cur  [private]
```

curent memory address

#### 4.28.3.3 ms_memory_size

```
int64_t MStorage::ms_memory_size  [private]
```

memory size actually mmaped

#### 4.28.3.4 ms_record_max

```
int64_t MStorage::ms_record_max  [private]
```

max records with actual mmaped memory

#### 4.28.3.5 ms_record_num

```
int64_t MStorage::ms_record_num  [private]
```

current record used

#### 4.28.3.6 ms_record_size

```
int64_t MStorage::ms_record_size  [private]
```

size per record

#### 4.28.3.7 pad

```
char MStorage::pad[128-4 *sizeof(int64_t) -sizeof(void *)]  [private]
```

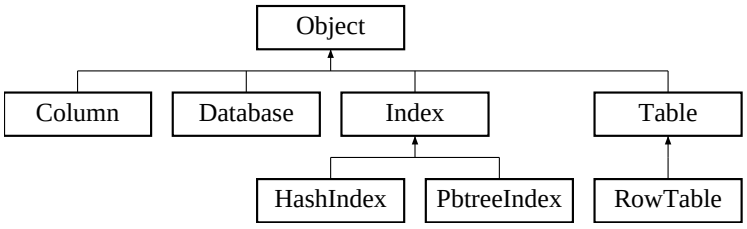
The documentation for this class was generated from the following file:

- system/rowtable.h

## 4.29 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

### 4.29.1 Detailed Description

definition of Object, basic element in database.

### 4.29.2 Constructor & Destructor Documentation

#### 4.29.2.1 Object()

```
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 |

### 4.29.3 Member Function Documentation

#### 4.29.3.1 changeName()

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

change object name(not in use).

#### 4.29.3.2 getOid()

```
int64_t Object::getOid (
            void  )  [inline]
```

get identifier of object.

#### 4.29.3.3 getOname()

```
char * Object::getOname (
            void  )  [inline]
```

get object name.

#### 4.29.3.4 getOtype()

```
ObjectType Object::getOtype (
            void  )  [inline]
```

get object type.

#### 4.29.3.5 print()

```
virtual void Object::print (
            void  )  [inline], [virtual]
```

print the object infomation.

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

#### 4.29.3.6 shut()

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

shut down the object.

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

### 4.29.4 Member Data Documentation

#### 4.29.4.1 o_id

```
int64_t Object::o_id  [private]
```

object identifier

#### 4.29.4.2 o_name

```
char Object::o_name[OBJ_NAME_MAX]  [private]
```

object name, max length OBJ_NAME_MAX

#### 4.29.4.3 o_type

```
ObjectType Object::o_type  [private]
```

object type

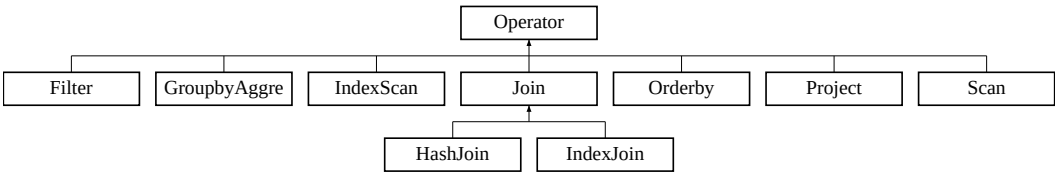
The documentation for this class was generated from the following file:

- system/schema.h

## 4.30 Operator Class Reference

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

Inheritance diagram for Operator:




### Public Member Functions

- Operator ()
- virtual ~Operator ()
- virtual bool close ()
- char * getBuffer ()
- virtual bool getNext ()
- virtual bool open ()
- void setBuffer (char *buffer_allocated)

## Public Attributes

- char * buffer_from_father

### 4.30.1 Constructor & Destructor Documentation

#### 4.30.1.1 Operator()

```
Operator::Operator ( )  [inline]
```

keep this buffer for passing record Constructor

#### 4.30.1.2 ~Operator()

```
virtual Operator::~Operator ( )  [inline], [virtual]
```

Destructor

### 4.30.2 Member Function Documentation

#### 4.30.2.1 close()

```
virtual bool Operator::close ( )  [inline], [virtual]
```

close this operator, release all resources

**Return values**

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

Reimplemented in Scan, IndexScan, Filter, IndexJoin, HashJoin, Project, GroupbyAggre, Orderby, and Join.

#### 4.30.2.2 getBuffer()

```
char * Operator::getBuffer ( )  [inline]
```

get current operator's buffer to write

#### 4.30.2.3 getNext()

```
virtual bool Operator::getNext ( )  [inline], [virtual]
```

get next record from the table iterately

**Return values**

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

Reimplemented in Scan, IndexScan, Filter, IndexJoin, HashJoin, Project, GroupbyAggre, Orderby, and Join.

#### 4.30.2.4 open()

```
virtual bool Operator::open ( )  [inline], [virtual]
```

open an operator

**Return values**

| | |
|---|---|
| *true* | successfully opened |
| *false* | failed |

Reimplemented in Scan, IndexScan, Filter, IndexJoin, HashJoin, Project, GroupbyAggre, Orderby, and Join.

#### 4.30.2.5 setBuffer()

```
void Operator::setBuffer (
            char * buffer_allocated )  [inline]
```

Save allocated buffer from father should be called before this operator's open

**Parameters**

| | |
|---|---|
| *buffer_allocated* | a static buffer to store the next record |

### 4.30.3 Member Data Documentation

#### 4.30.3.1 buffer_from_father

```
char* Operator::buffer_from_father
```

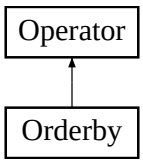
The documentation for this class was generated from the following file:

- system/executor.h

## 4.31 Orderby Class Reference

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

Inheritance diagram for Orderby:




### Public Member Functions

- Orderby ()
- ~Orderby ()
- bool close ()
- bool getNext ()
- bool open ()
- void set (std::vector< int64_t > input_colid, std::vector< int > orderby_rank)
- void setChild (Operator *child)

### Private Attributes

- int arrayid
- Operator * child
- char * child_buffer
- std::vector< int64_t > colid
- std::vector< int > coloff
- std::vector< int > colrank
- std::vector< BasicType * > coltype
- std::vector< char * > middle_buf_array
- int64_t middle_buf_size
- int orderby_num
- int64_t self_buf_size
- int64_t tuple_size

### Additional Inherited Members

### 4.31.1 Constructor & Destructor Documentation

#### 4.31.1.1 Orderby()

```
Orderby::Orderby ( )  [inline]
```

Constructor

#### 4.31.1.2 ~Orderby()

```
Orderby::~Orderby ( )  [inline]
```

Destructor

### 4.31.2 Member Function Documentation

#### 4.31.2.1 close()

```
bool Orderby::close ( )  [virtual]
```

close this operator, release all resources

**Return values**

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

Reimplemented from Operator.

#### 4.31.2.2 getNext()

```
bool Orderby::getNext ( )  [virtual]
```

get next record from the table iterately

**Return values**

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

Reimplemented from Operator.

#### 4.31.2.3 open()

```
bool Orderby::open ( )  [virtual]
```

open a Orderby operator

**Return values**

| | |
|---|---|
| *true* | successfully opened |
| *false* | failed |

Reimplemented from Operator.

#### 4.31.2.4 set()

```
void Orderby::set (
            std::vector< int64_t > input_colid,
            std::vector< int > orderby_rank )  [inline]
```

set the variables of Orderby

**Parameters**

| | |
|---|---|
| *input_colid* | the input column id |
| *orderby_rank* | the rank of the column needed to order by |

#### 4.31.2.5 setChild()

```
void Orderby::setChild (
            Operator * child )  [inline]
```

Set child operator, shoud be called before open

**Parameters**

| | |
|---|---|
| *child* | child operator |

### 4.31.3 Member Data Documentation

#### 4.31.3.1 arrayid

```
int Orderby::arrayid  [private]
```

the id to indicate which one in the array shoud be the next result

#### 4.31.3.2 child

```
Operator* Orderby::child  [private]
```

child of this operator on the operator tree

#### 4.31.3.3 child_buffer

```
char* Orderby::child_buffer  [private]
```

the buffer for the child result

#### 4.31.3.4 colid

```
std::vector<int64_t> Orderby::colid  [private]
```

the vector of input column id

#### 4.31.3.5 coloff

```
std::vector<int> Orderby::coloff  [private]
```

the offset of column needed to order by

#### 4.31.3.6 colrank

```
std::vector<int> Orderby::colrank  [private]
```

the rank of column needed to order by

#### 4.31.3.7 coltype

```
std::vector<BasicType*> Orderby::coltype  [private]
```

the data type of column needed to order by

#### 4.31.3.8 middle_buf_array

```
std::vector<char*> Orderby::middle_buf_array  [private]
```

array of the middle result buffer

#### 4.31.3.9 middle_buf_size

```
int64_t Orderby::middle_buf_size  [private]
```

the size of the middle result buffer

#### 4.31.3.10 orderby_num

```
int Orderby::orderby_num  [private]
```

the number of the column needed to order by

#### 4.31.3.11 self_buf_size

```
int64_t Orderby::self_buf_size  [private]
```

the size of the self buffer

#### 4.31.3.12 tuple_size

```
int64_t Orderby::tuple_size  [private]
```

the size of one row from child result

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

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

## 4.32 Pbtree Class Reference

```
#include <pbtree.h>
```

### Public Member Functions

- bool allocate (char *&p, int leve)
- int cap2leve (int cap)
- bool del (key_type key, void *ptr)
- bool free (char *p, int leve)
- int get_recptr (void *p, void *match[ ], int capicity, int &pos)
- void init (void)
- bool insert (key_type key, void *ptr)
- int leve2cap (int leve)
- int leve2size (int leve)
- void * lookup (key_type)
- void * lookup_s (key_type key, int *pos)
- void print (void)
- int scan (void **p, int *spos, key_type startkey, key_type endkey, void *area[ ], int *num)
- void shut (void)
- int size2leve (int size)

## Private Attributes

- char * p_free_header [16]
- pbtree p_pbtree

### 4.32.1 Member Function Documentation

#### 4.32.1.1 allocate()

```
bool Pbtree::allocate (
            char *& p,
            int leve )  [inline]
```

#### 4.32.1.2 cap2leve()

```
int Pbtree::cap2leve (
            int cap )  [inline]
```

#### 4.32.1.3 del()

```
bool Pbtree::del (
            key_type key,
            void * ptr )
```

#### 4.32.1.4 free()

```
bool Pbtree::free (
            char * p,
            int leve )  [inline]
```

#### 4.32.1.5 get_recptr()

```
int Pbtree::get_recptr (
            void * p,
            void * match[],
            int capicity,
            int & pos )
```

#### 4.32.1.6 init()

```
void Pbtree::init (
            void  )
```

#### 4.32.1.7 insert()

```
bool Pbtree::insert (
            key_type key,
            void * ptr )
```

#### 4.32.1.8 leve2cap()

```
int Pbtree::leve2cap (
            int leve )  [inline]
```

#### 4.32.1.9 leve2size()

```
int Pbtree::leve2size (
            int leve )  [inline]
```

#### 4.32.1.10 lookup()

```
void * Pbtree::lookup (
            key_type key )
```

#### 4.32.1.11 lookup_s()

```
void * Pbtree::lookup_s (
            key_type key,
            int * pos )
```

#### 4.32.1.12 print()

```
void Pbtree::print (
            void  )
```

#### 4.32.1.13 scan()

```
int Pbtree::scan (
            void ** p,
            int * spos,
            key_type startkey,
            key_type endkey,
            void * area[],
            int * num )
```

#### 4.32.1.14 shut()

```
void Pbtree::shut (
            void  )
```

#### 4.32.1.15 size2leve()

```
int Pbtree::size2leve (
            int size )  [inline]
```

### 4.32.2 Member Data Documentation

#### 4.32.2.1 p_free_header

```
char* Pbtree::p_free_header[16]  [private]
```

#### 4.32.2.2 p_pbtree

```
pbtree Pbtree::p_pbtree  [private]
```

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

- system/pbtree.h
- system/pbtree.cc

## 4.33 Pbtree Class Reference

```
#include <pbtree.h>
```

## Public Member Functions

- bool allocate (char ∗&p, int leve)
- int cap2leve (int cap)
- bool del (key_type key, void ∗ptr)
- bool free (char ∗p, int leve)
- int get_recptr (void ∗p, void ∗match[ ], int capicity, int &pos)
- void init (void)
- bool insert (key_type key, void ∗ptr)
- int leve2cap (int leve)
- int leve2size (int leve)
- void ∗ lookup (key_type)
- void ∗ lookup_s (key_type key, int ∗pos)
- void print (void)
- int scan (void ∗∗p, int ∗spos, key_type startkey, key_type endkey, void ∗area[ ], int ∗num)
- void shut (void)
- int size2leve (int size)

## Private Attributes

- char ∗ p_free_header [16]
- pbtree p_pbtree

### 4.33.1 Member Function Documentation

#### 4.33.1.1 allocate()

```
bool Pbtree::allocate (
            char *& p,
            int leve )  [inline]
```

#### 4.33.1.2 cap2leve()

```
int Pbtree::cap2leve (
            int cap )  [inline]
```

#### 4.33.1.3 del()

```
bool Pbtree::del (
            key_type key,
            void * ptr )
```

#### 4.33.1.4 free()

```
bool Pbtree::free (
            char * p,
            int leve )  [inline]
```

#### 4.33.1.5 get_recptr()

```
int Pbtree::get_recptr (
            void * p,
            void * match[],
            int capicity,
            int & pos )
```

#### 4.33.1.6 init()

```
void Pbtree::init (
            void  )
```

#### 4.33.1.7 insert()

```
bool Pbtree::insert (
            key_type key,
            void * ptr )
```

#### 4.33.1.8 leve2cap()

```
int Pbtree::leve2cap (
            int leve )  [inline]
```

#### 4.33.1.9 leve2size()

```
int Pbtree::leve2size (
            int leve )  [inline]
```

#### 4.33.1.10 lookup()

```
void * Pbtree::lookup (
            key_type key )
```

#### 4.33.1.11 lookup_s()

```
void * Pbtree::lookup_s (
            key_type key,
            int * pos )
```

#### 4.33.1.12 print()

```
void Pbtree::print (
            void  )
```

#### 4.33.1.13 scan()

```
int Pbtree::scan (
            void ** p,
            int * spos,
            key_type startkey,
            key_type endkey,
            void * area[],
            int * num )
```

#### 4.33.1.14 shut()

```
void Pbtree::shut (
            void  )
```

#### 4.33.1.15 size2leve()

```
int Pbtree::size2leve (
            int size )  [inline]
```

### 4.33.2 Member Data Documentation

#### 4.33.2.1 p_free_header

```
char* Pbtree::p_free_header[16]  [private]
```

#### 4.33.2.2 p_pbtree

```
pbtree Pbtree::p_pbtree  [private]
```

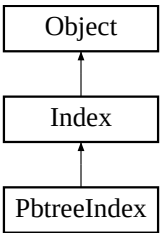
The documentation for this class was generated from the following files:

- system/pbtree.h
- system/pbtree.cc

## 4.34 PbtreeIndex Class Reference

```
#include <pbtreeindex.h>
```

Inheritance diagram for PbtreeIndex:




### Public Member Functions

- PbtreeIndex (int64_t pi_id, const char *i_name, Key &i_key)
- bool del (void *i_data, void *p_del)
- bool init (void)
- bool insert (void *i_data, void *p_in)
- bool lookup (void *i_data, void *info, void *&result)
- void print (void)
- bool scan (void *info, void *&result)
- bool set_ls (void *i_data1, void *i_data2, void *info)
- bool setIndexDTpye (BasicType *i_dt)
- bool shut (void)

## Private Attributes

- BasicType * pi_datatype
- Pbtree pi_pbtree

## Additional Inherited Members

### 4.34.1 Constructor & Destructor Documentation

#### 4.34.1.1 PbtreeIndex()

```
PbtreeIndex::PbtreeIndex (
            int64_t pi_id,
            const char * i_name,
            Key & i_key )  [inline]
```

constructor.

**Parameters**

| | |
|---|---|
| *pi_id* | pbtree index identifier |
| *i_name* | index name |
| *i_key* | key of this index |

### 4.34.2 Member Function Documentation

#### 4.34.2.1 del()

```
bool PbtreeIndex::del (
            void * i_data,
            void * p_del )  [virtual]
```

del an entry pbtree index.

**Parameters**

| | |
|---|---|
| *i_data* | buffer of column data |
| *p_del* | pointer of row to delete |

**Return values**

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

Reimplemented from Index.

#### 4.34.2.2 init()

```
bool PbtreeIndex::init (
            void  )  [virtual]
```

init PbtreeIndex

**Return values**

| | |
|---|---|
| *true* | successfully inited |

Reimplemented from Index.

#### 4.34.2.3 insert()

```
bool PbtreeIndex::insert (
            void * i_data,
            void * p_in )  [virtual]
```

insert an entry to pbtree index.

**Parameters**

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

**Return values**

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

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.

#### 4.34.2.4 lookup()

```
bool PbtreeIndex::lookup (
            void * i_data,
            void * info,
            void *& result )  [virtual]
```

setup for pbtree index lookup.

**Parameters**

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

**Return values**

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

lookup pbtree index.

**Parameters**

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

**Return values**

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

Reimplemented from Index.

#### 4.34.2.5 print()

```
void PbtreeIndex::print (
            void  )  [virtual]
```

print Pbtree index information.

print pbtree index information.

Reimplemented from Index.

#### 4.34.2.6 scan()

```
bool PbtreeIndex::scan (
            void * info,
            void *& result )  [virtual]
```

iterate on calling to scan for values, > left value & < right value

**Parameters**

| | |
|---|---|
| *info* | pointer of an index info |
| *result* | return the pointer to the indexed row |

**Return values**

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

iterate on calling to scan for values.

**Parameters**

| | |
|---|---|
| *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 from Index.

#### 4.34.2.7 set_ls()

```
bool PbtreeIndex::set_ls (
            void * i_data1,
            void * i_data2,
            void * info )  [virtual]
```

setup for pbtree index lookup.

**Parameters**

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

**Return values**

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

Reimplemented from Index.

#### 4.34.2.8 setIndexDTpye()

```
bool PbtreeIndex::setIndexDTpye (
            BasicType * i_dt )
```

add indexed column's data type.

**Parameters**

| | |
|---|---|
| *i_dt* | data type of indexed column |
| *offset* | offset of column in a rowtable |

set indexed column's data type.

**Parameters**

| | |
|---|---|
| *i_dt* | data type of indexed column |
| *offset* | offset of column in a rowtable |

#### 4.34.2.9 shut()

```
bool PbtreeIndex::shut (
            void  )  [virtual]
```

free memory of Pbtree and other strucure.

**Return values**

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

free memory of hash table and other strucure.

**Return values**

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

Reimplemented from Index.

### 4.34.3 Member Data Documentation

#### 4.34.3.1 pi_datatype

```
BasicType* PbtreeIndex::pi_datatype  [private]
```

datatype of this column

#### 4.34.3.2 pi_pbtree

```
Pbtree PbtreeIndex::pi_pbtree  [private]
```

Pbtree, allow duplicated elements

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

- system/pbtreeindex.h
- system/pbtreeindex.cc

## 4.35 PbtreeInfo Struct Reference

```
#include <pbtreeindex.h>
```

### Public Attributes

- void * area [PBTREEINFO_CAPICITY]
- int cr_area
- int cr_resu
- void * l_ptr
- int le_resu
- key_type left
- int pos_resu
- void * result [PBTREEINFO_CAPICITY]
- key_type right
- int s_end
- int s_num
- int s_pos
- void * s_ptr

### 4.35.1 Detailed Description

definition of PbtreeInfo.

## 4.35.2 Member Data Documentation

### 4.35.2.1 area

```
void* PbtreeInfo::area[PBTREEINFO_CAPICITY]
```

buffer for Pbtree

### 4.35.2.2 cr_area

```
int PbtreeInfo::cr_area
```

current area array pos in use

### 4.35.2.3 cr_resu

```
int PbtreeInfo::cr_resu
```

current result array pos in use

### 4.35.2.4 l_ptr

```
void* PbtreeInfo::l_ptr
```

lookup elements ptr

### 4.35.2.5 le_resu

```
int PbtreeInfo::le_resu
```

len of current result

### 4.35.2.6 left

```
key_type PbtreeInfo::left
```

lookup key or scan left edge

### 4.35.2.7 pos_resu

```
int PbtreeInfo::pos_resu
```

pos in match array, init 0

#### 4.35.2.8 result

```
void* PbtreeInfo::result[PBTREEINFO_CAPICITY]
```

buffer for element

#### 4.35.2.9 right

```
key_type PbtreeInfo::right
```

scan right edge

#### 4.35.2.10 s_end

```
int PbtreeInfo::s_end
```

scan tag, scan has more? 0 means more

#### 4.35.2.11 s_num

```
int PbtreeInfo::s_num
```

scan area buffer number, should be init, return scan num

#### 4.35.2.12 s_pos

```
int PbtreeInfo::s_pos
```

scan pos in bnode, acquired by lookup_s

#### 4.35.2.13 s_ptr

```
void* PbtreeInfo::s_ptr
```

scan bnode ptr, acquired by lookup_s

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

- system/pbtreeindex.h

## 4.36 Pointer8B Class Reference

```
#include <pbtree.h>
```

## Public Member Functions

- operator char * ()
- operator struct bleaf * ()
- operator struct bnode * ()
- operator unsigned long long ()
- operator void * ()
- Pointer8B & operator= (const Pointer8B &p)
- Pointer8B & operator= (const void *ptr)
- void print (void)

## Public Attributes

- unsigned long long value

## 4.36.1 Member Function Documentation

### 4.36.1.1 operator char *()

```
Pointer8B::operator char * ( )  [inline]
```

### 4.36.1.2 operator struct bleaf *()

```
Pointer8B::operator struct bleaf * ( )  [inline]
```

### 4.36.1.3 operator struct bnode *()

```
Pointer8B::operator struct bnode * ( )  [inline]
```

### 4.36.1.4 operator unsigned long long()

```
Pointer8B::operator unsigned long long ( )  [inline]
```

#### 4.36.1.5 operator void *()

```
Pointer8B::operator void * ( )  [inline]
```

#### 4.36.1.6 operator=() [1/2]

```
Pointer8B & Pointer8B::operator= (
            const Pointer8B & p )  [inline]
```

#### 4.36.1.7 operator=() [2/2]

```
Pointer8B & Pointer8B::operator= (
            const void * ptr )  [inline]
```

#### 4.36.1.8 print()

```
void Pointer8B::print (
            void  )  [inline]
```

### 4.36.2 Member Data Documentation

#### 4.36.2.1 value

```
unsigned long long Pointer8B::value
```

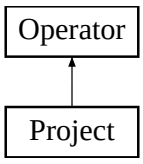
The documentation for this class was generated from the following file:

- system/pbtree.h

## 4.37 Project Class Reference

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

Inheritance diagram for Project:



## Public Member Functions

- Project ()
- Project (std::vector< int64_t > in_cid, std::vector< int64_t > out_cid)
- ~Project ()
- bool close ()
- int64_t getColnum ()
- bool getNext ()
- BasicType ** getSchema ()
- bool open ()
- void setChild (Operator *child)
- void setProjCol (std::vector< int64_t > in_cid, std::vector< int64_t > out_cid)
- bool top ()

## Private Attributes

- char * buf_for_child
- Operator * child
- int64_t in_buf_size
- int64_t in_tuple_size
- std::vector< int64_t > input_cid
- std::vector< int64_t > input_off
- std::vector< char * > input_pos
- std::vector< BasicType * > input_type
- std::vector< int64_t > out_to_in
- std::vector< int64_t > output_cid
- char * output_type
- int64_t output_type_buf_size
- int64_t output_type_size
- int64_t self_buf_size
- bool topid

## Additional Inherited Members

### 4.37.1 Detailed Description

Definition of class Project the projection operator

### 4.37.2 Constructor & Destructor Documentation

#### 4.37.2.1 Project() [1/2]

```
Project::Project ( )  [inline]
```

the indicator to show that if this is the top of the operator tree Constructor

#### 4.37.2.2 ~Project()

```
Project::~Project ( )  [inline]
```

Destructor

#### 4.37.2.3 Project() [2/2]

```
Project::Project (
            std::vector< int64_t > in_cid,
            std::vector< int64_t > out_cid )  [inline]
```

Overload constructor, set some attributes if already known

**Parameters**

| | |
|---|---|
| *in_cid* | input column ID |
| *out_cid* | output column ID |

### 4.37.3 Member Function Documentation

#### 4.37.3.1 close()

```
bool Project::close ( )  [virtual]
```

Close this operator

**Return values**

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

Reimplemented from Operator.

#### 4.37.3.2 getColnum()

```
int64_t Project::getColnum ( )  [inline]
```

Get output column number

**Return values**

| | |
|---|---|
| *size* | of output_cid |

#### 4.37.3.3 getNext()

```
bool Project::getNext ( )  [virtual]
```

Generate next record

**Return values**

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

Reimplemented from Operator.

#### 4.37.3.4 getSchema()

```
BasicType ** Project::getSchema ( )  [inline]
```

Get output data type

**Return values**

| | |
|---|---|
| *array* | of pointers to basic type of output columns |

#### 4.37.3.5 open()

```
bool Project::open ( )  [virtual]
```

Open a Project operator

**Return values**

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

Reimplemented from Operator.

#### 4.37.3.6 setChild()

```
void Project::setChild (
            Operator * child )  [inline]
```

Set child operator

**Parameters**

| | |
|---|---|
| *child* | child operator to set |

#### 4.37.3.7 setProjCol()

```
void Project::setProjCol (
            std::vector< int64_t > in_cid,
            std::vector< int64_t > out_cid )  [inline]
```

Set input and output columns

**Parameters**

| | |
|---|---|
| *in_cid* | input column ID |
| *out_cid* | output column ID |

#### 4.37.3.8 top()

```
bool Project::top ( )
```

when this operator be the top of the operator tree, do this, set the buffer for self

**Returns**

true success

false failure

### 4.37.4 Member Data Documentation

#### 4.37.4.1 buf_for_child

```
char* Project::buf_for_child  [private]
```

child operator

#### 4.37.4.2 child

```
Operator* Project::child  [private]
```

#### 4.37.4.3 in_buf_size

```
int64_t Project::in_buf_size  [private]
```

input tuple size

#### 4.37.4.4 in_tuple_size

```
int64_t Project::in_tuple_size  [private]
```

an output column is which column from input

#### 4.37.4.5 input_cid

```
std::vector< int64_t > Project::input_cid  [private]
```

buffer size allocated for operator on the top/self

#### 4.37.4.6 input_off

```
std::vector< int64_t > Project::input_off  [private]
```

basic type of each column from input

#### 4.37.4.7 input_pos

```
std::vector< char * > Project::input_pos  [private]
```

offset of each column from input

#### 4.37.4.8 input_type

```
std::vector< BasicType * > Project::input_type  [private]
```

input buffer size

#### 4.37.4.9 out_to_in

```
std::vector< int64_t > Project::out_to_in  [private]
```

output column ID, can be reordered

#### 4.37.4.10 output_cid

```
std::vector< int64_t > Project::output_cid  [private]
```

input column ID

#### 4.37.4.11 output_type

```
char* Project::output_type  [private]
```

postion of each column from input

#### 4.37.4.12 output_type_buf_size

```
int64_t Project::output_type_buf_size  [private]
```

size of output type array

#### 4.37.4.13 output_type_size

```
int64_t Project::output_type_size  [private]
```

basic type of each output column

#### 4.37.4.14 self_buf_size

```
int64_t Project::self_buf_size  [private]
```

buffer allocated for child

#### 4.37.4.15 topid

```
bool Project::topid  [private]
```

size of buffer allocated for *output_type

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

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

## 4.38 RequestColumn Struct Reference

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

### Public Attributes

- AggregateMethod aggregate_method
- char name [128]

### 4.38.1 Detailed Description

definition of request column.

### 4.38.2 Member Data Documentation

#### 4.38.2.1 aggregate_method

```
AggregateMethod RequestColumn::aggregate_method
```

#### 4.38.2.2 name

```
char RequestColumn::name[128]
```

name of column

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

- system/executor.h

## 4.39 RequestTable Struct Reference

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

### Public Attributes

- char name [128]

### 4.39.1 Detailed Description

definition of request table.

### 4.39.2 Member Data Documentation

#### 4.39.2.1 name

```
char RequestTable::name[128]
```

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

- system/executor.h

## 4.40 ResultTable Class Reference

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

### Public Member Functions

- int append (char *src)
  *write a row into result table*
- int dump (FILE *fp)
- char * getRC (int row, int column)
- int init (BasicType *col_types[ ], int col_num, int64_t capicity=1024)
- int print (void)
- int shut (void)
- int writeRC (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

### 4.40.1 Detailed Description

definition of result table.

### 4.40.2 Member Function Documentation

#### 4.40.2.1 append()

```
int ResultTable::append (
            char * src )
```

write a row into result table

**Parameters**

| | |
|---|---|
| *src* | the row buffer to be written |

**Returns**

1 success

0 error

#### 4.40.2.2 dump()

```
int ResultTable::dump (
            FILE * fp )
```

write to file with FILE *fp

#### 4.40.2.3 getRC()

```
char * ResultTable::getRC (
            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 |

#### 4.40.2.4 init()

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

init alloc memory and set initial value

**Parameters**

| | |
|---|---|
| *col_types* | array of column type pointers |
| *col_num* | number of columns in this ResultTable |
| *capicity* | buffer_size, power of 2 |

**Return values**

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

#### 4.40.2.5 print()

```
int ResultTable::print (
            void  )
```

print result table, split by '\t', output a line per row

**Return values**

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

#### 4.40.2.6 shut()

```
int ResultTable::shut (
            void  )
```

free memory of this result table to g_memory

#### 4.40.2.7 writeRC()

```
int ResultTable::writeRC (
            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* | data pointer of a column |

**Return values**

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

### 4.40.3 Member Data Documentation

#### 4.40.3.1 buffer

```
char* ResultTable::buffer
```

pointer of buffer alloced from g_memory

#### 4.40.3.2 buffer_size

```
int64_t ResultTable::buffer_size
```

size of buffer, power of 2

#### 4.40.3.3 column_number

```
int ResultTable::column_number
```

columns number that a result row consist of

#### 4.40.3.4 column_type

```
BasicType** ResultTable::column_type
```

each column data type

#### 4.40.3.5 offset

```
int* ResultTable::offset
```

#### 4.40.3.6 offset_size

```
int ResultTable::offset_size
```

#### 4.40.3.7 row_capicity

```
int ResultTable::row_capicity
```

maximum capicity of rows according to buffer size and length of row

#### 4.40.3.8 row_length

```
int ResultTable::row_length
```

length per result row

#### 4.40.3.9 row_number

```
int ResultTable::row_number
```

current usage of rows

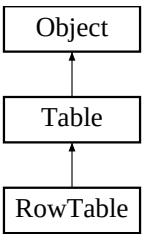
The documentation for this class was generated from the following files:

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

## 4.41 RowTable Class Reference

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

Inheritance diagram for RowTable:



## Public Member Functions

- RowTable (int64_t r_id, const char *r_name)
- bool del (char *row_pointer)
- bool del (int64_t record_rank)
- 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 *columns[])
- bool insert (char *source)
- bool loadData (const char *filename)
- bool printData (void)
- bool select (char *row_pointer, char *dest)
- bool select (int64_t record_rank, char *dest)
- bool selectCol (char *row_pointer, int64_t column_rank, char *dest)
- bool selectCol (int64_t record_rank, int64_t column_rank, char *dest)
- bool selectCols (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *dest)
- bool selectCols (int64_t record_rank, 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 (char *row_pointer, 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 (int64_t record_rank, 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

### 4.41.1 Detailed Description

definition of class RowTable.

### 4.41.2 Constructor & Destructor Documentation

#### 4.41.2.1 RowTable()

```
RowTable::RowTable (
            int64_t r_id,
            const char * r_name )  [inline]
```

constructor.

**Parameters**

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

### 4.41.3 Member Function Documentation

#### 4.41.3.1 access()

```
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 |

#### 4.41.3.2 accessCol()

```
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 |

**Return values**

| | |
|---|---|
| *false* | failure |

#### 4.41.3.3 del() [1/2]

```
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.

#### 4.41.3.4 del() [2/2]

```
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.

#### 4.41.3.5 finish()

```
bool RowTable::finish (
            void  )  [virtual]
```

finish, leave it empty.

Reimplemented from Table.

#### 4.41.3.6 getMStorage()

```
MStorage & RowTable::getMStorage (
            void  )  [inline]
```

get storage of table.

#### 4.41.3.7 getRecordNum()

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

get the last record rank.

Reimplemented from Table.

#### 4.41.3.8 getRecordPtr()

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

get row record pointer.

**Parameters**

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

**Return values**

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

Reimplemented from Table.

#### 4.41.3.9 getRPattern()

```
RPattern & RowTable::getRPattern (
            void  )  [inline]
```

get pattern of table.

#### 4.41.3.10 init()

```
bool RowTable::init (
            void  )  [virtual]
```

init, leave it empty.

Reimplemented from Table.

#### 4.41.3.11 insert() [1/2]

```
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.

#### 4.41.3.12 insert() [2/2]

```
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.

#### 4.41.3.13 invalid()

```
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 |

#### 4.41.3.14 isValid()

```
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 |

#### 4.41.3.15 loadData()

```
bool RowTable::loadData (
```

```
            const char * filename )  [virtual]
```

load data of the table(not in use).

Reimplemented from Table.

#### 4.41.3.16 printData()

```
bool RowTable::printData (
            void  )  [virtual]
```

print table data, for debug.

Reimplemented from Table.

#### 4.41.3.17 select() [1/2]

```
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.

#### 4.41.3.18 select() [2/2]

```
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.

#### 4.41.3.19 selectCol() [1/2]

```
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 |
| *false* | failure |

Reimplemented from Table.

#### 4.41.3.20 selectCol() [2/2]

```
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.

#### 4.41.3.21 selectCols() [1/2]

```
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.

#### 4.41.3.22 selectCols() [2/2]

```
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.

#### 4.41.3.23 shut()

```
bool RowTable::shut (
            void  )  [virtual]
```

shut down r_pattern and r_storage, free their memory.

Reimplemented from Table.

#### 4.41.3.24 updateCol() [1/2]

```
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.

#### 4.41.3.25 updateCol() [2/2]

```
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.

#### 4.41.3.26 updateCols() [1/4]

```
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.

#### 4.41.3.27 updateCols() [2/4]

```
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.

#### 4.41.3.28 updateCols() [3/4]

```
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.

#### 4.41.3.29 updateCols() [4/4]

```
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.

### 4.41.4 Member Data Documentation

#### 4.41.4.1 r_pattern

```
RPattern RowTable::r_pattern  [private]
```

pattern of row

#### 4.41.4.2 r_storage

```
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

## 4.42 RPattern Class Reference

`#include <rowtable.h>`

### 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

### 4.42.1 Detailed Description

definition of class RPattern, describe row struture.

### 4.42.2 Member Function Documentation

#### 4.42.2.1 addColumn()

```
bool RPattern::addColumn (
            BasicType * col_type )  [inline]
```

add column infomation.

**Parameters**

| | |
|---|---|
| *col_type* | data type of the column |

#### 4.42.2.2 getColumnOffset()

```
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 |

#### 4.42.2.3 getColumnType()

```
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 |

#### 4.42.2.4 getRowSize()

```
int64_t RPattern::getRowSize (
            void ) [inline]
```

get size of a row record.

**Return values**

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

#### 4.42.2.5 init()

```
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 |

#### 4.42.2.6 print()

```
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 |

#### 4.42.2.7 reset()

```
void RPattern::reset (
            void  )  [inline]
```

reset if addcolumn error happen.

#### 4.42.2.8 shut()

```
void RPattern::shut (
            void  )  [inline]
```

shut down, free memory allocated from g_memory.

### 4.42.3 Member Data Documentation

#### 4.42.3.1 par

```
char RPattern::par[128 - 3 *sizeof(void *) - 4 *sizeof(int64_t)]  [private]
```

make the sizeof RPattern 128, managed by g_memory

#### 4.42.3.2 rp_colnum

```
int64_t RPattern::rp_colnum  [private]
```

total columns

#### 4.42.3.3 rp_current

```
int64_t RPattern::rp_current  [private]
```

currrent columns already set offset and datatype

#### 4.42.3.4 rp_dtype

```
BasicType** RPattern::rp_dtype  [private]
```

array of pointers to each column's data type

#### 4.42.3.5 rp_mem_sz

```
int64_t RPattern::rp_mem_sz  [private]
```

memory size

#### 4.42.3.6 rp_memory

```
char* RPattern::rp_memory  [private]
```

memory to store rp_offfset and rp_datatype

#### 4.42.3.7 rp_offset

```
int64_t* RPattern::rp_offset  [private]
```

offset of column in a row

#### 4.42.3.8 rp_row_sz

```
int64_t RPattern::rp_row_sz  [private]
```

size of a row record

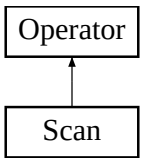
The documentation for this class was generated from the following file:

- system/rowtable.h

## 4.43 Scan Class Reference

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

Inheritance diagram for Scan:




### Public Member Functions

- Scan ()
- ~Scan ()
- bool close ()
- bool getNext ()
- bool open ()
- void setTable (Table *table)

### Private Attributes

- int64_t next_record
- Table * scan_table
- int64_t total_record

### Additional Inherited Members

### 4.43.1 Detailed Description

definition of the scan operator located at the buttom of the op-tree

### 4.43.2 Constructor & Destructor Documentation

#### 4.43.2.1 Scan()

```
Scan::Scan ( )  [inline]
```

next record to check Constructor

#### 4.43.2.2 ~Scan()

```
Scan::~Scan ( )  [inline]
```

Destructor

### 4.43.3 Member Function Documentation

#### 4.43.3.1 close()

```
bool Scan::close ( )  [virtual]
```

close this operator, release all resources

**Return values**

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

Reimplemented from Operator.

#### 4.43.3.2 getNext()

```
bool Scan::getNext ( )  [virtual]
```

get next record from the table iterately

**Return values**

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

Reimplemented from Operator.

#### 4.43.3.3 open()

```
bool Scan::open ( )  [virtual]
```

open a scan operator

**Return values**

| | |
|---|---|
| *true* | successfully opened |
| *false* | failed |

Note: the dest buffer should be allocated when first opened and keep static

Reimplemented from Operator.

#### 4.43.3.4 setTable()

```
void Scan::setTable (
            Table * table )  [inline]
```

Set the table to scan should be called before open

**Parameters**

| | |
|---|---|
| *table* | the designated table |

### 4.43.4 Member Data Documentation

#### 4.43.4.1 next_record

```
int64_t Scan::next_record  [private]
```

total record num of this table when first opened

#### 4.43.4.2 scan_table

```
Table* Scan::scan_table  [private]
```

#### 4.43.4.3 total_record

```
int64_t Scan::total_record  [private]
```

the designated table

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

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

## 4.44 SelectQuery Class Reference

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

### 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

### 4.44.1 Detailed Description

definition of selectquery.

### 4.44.2 Member Data Documentation

#### 4.44.2.1 database_id

```
int64_t SelectQuery::database_id
```

database to execute

#### 4.44.2.2 from_number

```
int SelectQuery::from_number
```

number of tables to select from

#### 4.44.2.3 from_table

```
RequestTable SelectQuery::from_table[4]
```

tables to select from, maximum 4

#### 4.44.2.4 groupby

```
RequestColumn SelectQuery::groupby[4]
```

columns to groupby

#### 4.44.2.5 groupby_number

```
int SelectQuery::groupby_number
```

number of columns to groupby

#### 4.44.2.6 having

```
Conditions SelectQuery::having
```

groupby conditions

#### 4.44.2.7 orderby

```
RequestColumn SelectQuery::orderby[4]
```

columns to orderby

#### 4.44.2.8 orderby_number

```
int SelectQuery::orderby_number
```

number of columns to orderby

#### 4.44.2.9 select_column

```
RequestColumn SelectQuery::select_column[4]
```

columns to select, maximum 4

#### 4.44.2.10 select_number

```
int SelectQuery::select_number
```

number of column to select

#### 4.44.2.11 where

```
Conditions SelectQuery::where
```

where meets conditions, maximum 4 & conditions

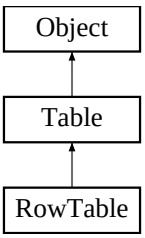
The documentation for this class was generated from the following file:

- system/executor.h

## 4.45 Table Class Reference

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

Inheritance 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 (char *columns[])
- virtual bool del (char *row_pointer)
- virtual bool del (int64_t record_rank)
- 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 ∗columns[ ])
- virtual bool insert (char ∗source)
- virtual bool loadData (const char ∗filename)
- virtual void print (void)
- virtual bool printData (void)
- virtual bool select (char ∗row_pointer, char ∗dest)
- virtual bool select (int64_t record_rank, char ∗dest)
- virtual bool selectCol (char ∗row_pointer, int64_t column_rank, char ∗dest)
- virtual bool selectCol (int64_t record_rank, int64_t column_rank, char ∗dest)
- virtual bool selectCols (char ∗row_pointer, int64_t column_total, int64_t ∗column_ranks, char ∗dest)
- virtual bool selectCols (int64_t record_rank, int64_t column_total, int64_t ∗column_ranks, char ∗dest)
- virtual bool shut (void)
- virtual bool updateCol (char ∗row_pointer, int64_t column_rank, char ∗source)
- virtual bool updateCol (int64_t record_rank, int64_t column_rank, char ∗source)
- virtual bool updateCols (char ∗row_pointer, 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 (int64_t record_rank, 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

### 4.45.1 Detailed Description

definition of class Table.

### 4.45.2 Constructor & Destructor Documentation

#### 4.45.2.1 ∼Table()

```
virtual Table::∼Table (
            void  )  [inline], [virtual]
```

destructor.

#### 4.45.2.2 Table()

```
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 |

## 4.45.3 Member Function Documentation

### 4.45.3.1 addColumn()

```
virtual bool Table::addColumn (
            int64_t column_id )  [inline], [virtual]
```

add column identifier to this table.

### 4.45.3.2 addIndex()

```
virtual bool Table::addIndex (
            int64_t index_id )  [inline], [virtual]
```

add index identifier to this table.

### 4.45.3.3 del() [1/3]

```
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 |

### 4.45.3.4 del() [2/3]

```
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.

#### 4.45.3.5 del() [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.

#### 4.45.3.6 finish()

```
virtual bool Table::finish (
            void  )  [inline], [virtual]
```

finish, important interface for son class

Reimplemented in RowTable.

#### 4.45.3.7 getColumnRank()

```
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 |

#### 4.45.3.8 getColumns()

```
std::vector< int64_t > & Table::getColumns (
            void  )  [inline]
```

get column identifier vector.

**Return values**

| | |
|---|---|
| *vector* | of column identifiers |

#### 4.45.3.9 getIndexRank()

```
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 |

#### 4.45.3.10 getIndexs()

```
std::vector< int64_t > & Table::getIndexs (
            void  )  [inline]
```

get index identifier vector.

#### 4.45.3.11 getRank()

```
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 |

#### 4.45.3.12 getRecordNum()

```
virtual int64_t Table::getRecordNum (
            void  )  [inline], [virtual]
```

get record number.

Reimplemented in RowTable.

#### 4.45.3.13 getRecordPtr()

```
virtual void * Table::getRecordPtr (
            int64_t row_rank )  [inline], [virtual]
```

get record pointer.

Reimplemented in RowTable.

#### 4.45.3.14 getTtype()

```
TableType Table::getTtype (
            void  )  [inline]
```

get table type.

#### 4.45.3.15 init()

```
virtual bool Table::init (
            void  )  [inline], [virtual]
```

init, important interface for son class

Reimplemented in RowTable.

#### 4.45.3.16 insert() [1/2]

```
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.

#### 4.45.3.17 insert() [2/2]

```
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.

#### 4.45.3.18 loadData()

```
virtual bool Table::loadData (
            const char * filename )  [inline], [virtual]
```

load data(not in use).

Reimplemented in RowTable.

#### 4.45.3.19 print()

```
virtual void Table::print (
            void  )  [inline], [virtual]
```

print table information.

Reimplemented from Object.

#### 4.45.3.20 printData()

```
virtual bool Table::printData (
            void  )  [inline], [virtual]
```

print data in table.

Reimplemented in RowTable.

#### 4.45.3.21 select() [1/2]

```
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.

#### 4.45.3.22 select() [2/2]

```
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.

#### 4.45.3.23 selectCol() [1/2]

```
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.

#### 4.45.3.24 selectCol() [2/2]

```
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.

#### 4.45.3.25 selectCols() [1/2]

```
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.

#### 4.45.3.26 selectCols() [2/2]

```
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.

#### 4.45.3.27 shut()

```
virtual bool Table::shut (
            void ) [inline], [virtual]
```

shut, important interface for son class

Reimplemented from Object.

Reimplemented in RowTable.

#### 4.45.3.28 updateCol() [1/2]

```
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 |
| *false* | failure |

Reimplemented in RowTable.

#### 4.45.3.29 updateCol() [2/2]

```
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.

#### 4.45.3.30 updateCols() [1/4]

```
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.

#### 4.45.3.31 updateCols() [2/4]

```
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.

#### 4.45.3.32 updateCols() [3/4]

```
virtual bool Table::updateCols (
            int64_t record_rank,
            int64_t column_total,
            int64_t * column_ranks,
            char * source )  [inline], [virtual]
```

Reimplemented in RowTable.

#### 4.45.3.33 updateCols() [4/4]

```
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.

### 4.45.4 Member Data Documentation

#### 4.45.4.1 t_columns

```
std::vector< int64_t > Table::t_columns  [private]
```

vector of columns' identifier

#### 4.45.4.2 t_index

```
std::vector< int64_t > Table::t_index  [private]
```

vector of index's identifier

#### 4.45.4.3 t_type

```
TableType Table::t_type  [private]
```

table type

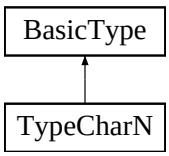
The documentation for this class was generated from the following file:

- system/schema.h

## 4.46 TypeCharN Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.46.1 Detailed Description

definition of class TypeCharN,please refer to BasicType,it's same.

### 4.46.2 Constructor & Destructor Documentation

#### 4.46.2.1 TypeCharN() [1/2]

```
TypeCharN::TypeCharN (
            int64_t typesize )  [inline]
```

constructor.

#### 4.46.2.2 TypeCharN() [2/2]

```
TypeCharN::TypeCharN (
            TypeCode typecode = CHARN_TC,
            int64_t typesize = 32 )  [inline]
```

## 4.46.3 Member Function Documentation

### 4.46.3.1 cmpEQ()

```
bool TypeCharN::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

### 4.46.3.2 cmpGE()

```
bool TypeCharN::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

### 4.46.3.3 cmpGT()

```
bool TypeCharN::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

### 4.46.3.4 cmpLE()

```
bool TypeCharN::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.46.3.5 cmpLT()

```
bool TypeCharN::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.46.3.6 copy()

```
int TypeCharN::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.46.3.7 formatBin()

```
int TypeCharN::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.46.3.8 formatTxt()

```
int TypeCharN::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

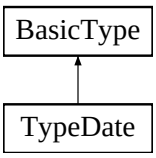
The documentation for this class was generated from the following file:

- system/datatype.h

# 4.47 TypeDate Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.47.1 Detailed Description

definition of class TypeDate,please refer to BasicType,it's same.

### 4.47.2 Constructor & Destructor Documentation

#### 4.47.2.1 TypeDate()

```
TypeDate::TypeDate (
            TypeCode typecode = DATE_TC,
            int64_t typesize = sizeof(time_t) )  [inline]
```

constructor.

### 4.47.3 Member Function Documentation

#### 4.47.3.1 cmpEQ()

```
bool TypeDate::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.47.3.2 cmpGE()

```
bool TypeDate::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.47.3.3 cmpGT()

```
bool TypeDate::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.47.3.4 cmpLE()

```
bool TypeDate::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.47.3.5 cmpLT()

```
bool TypeDate::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.47.3.6 copy()

```
int TypeDate::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.47.3.7 formatBin()

```
int TypeDate::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.47.3.8 formatTxt()

```
int TypeDate::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

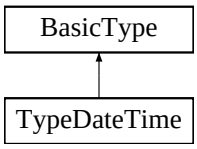
The documentation for this class was generated from the following file:

- system/datatype.h

## 4.48 TypeDateTime Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.48.1 Detailed Description

definition of class TypeDateTime,please refer to BasicType,it's same.

### 4.48.2 Constructor & Destructor Documentation

#### 4.48.2.1 TypeDateTime()

```
TypeDateTime::TypeDateTime (
            TypeCode typecode = DATETIME_TC,
            int64_t typesize = sizeof(time_t) )  [inline]
```

constructor.

### 4.48.3 Member Function Documentation

#### 4.48.3.1 cmpEQ()

```
bool TypeDateTime::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.48.3.2 cmpGE()

```
bool TypeDateTime::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.48.3.3 cmpGT()

```
bool TypeDateTime::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.48.3.4 cmpLE()

```
bool TypeDateTime::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.48.3.5 cmpLT()

```
bool TypeDateTime::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.48.3.6 copy()

```
int TypeDateTime::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.48.3.7 formatBin()

```
int TypeDateTime::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.48.3.8 formatTxt()

```
int TypeDateTime::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

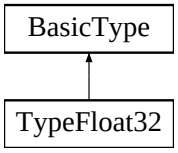
The documentation for this class was generated from the following file:

- system/datatype.h

# 4.49 TypeFloat32 Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.49.1 Detailed Description

definition of class TypeFloat32,please refer to BasicType,it's same.

### 4.49.2 Constructor & Destructor Documentation

#### 4.49.2.1 TypeFloat32()

```
TypeFloat32::TypeFloat32 (
            TypeCode typecode = FLOAT32_TC,
            int64_t typesize = sizeof(float) )  [inline]
```

constructor.

### 4.49.3 Member Function Documentation

#### 4.49.3.1 cmpEQ()

```
bool TypeFloat32::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.49.3.2 cmpGE()

```
bool TypeFloat32::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.49.3.3 cmpGT()

```
bool TypeFloat32::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.49.3.4 cmpLE()

```
bool TypeFloat32::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.49.3.5 cmpLT()

```
bool TypeFloat32::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.49.3.6 copy()

```
int TypeFloat32::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.49.3.7 formatBin()

```
int TypeFloat32::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.49.3.8 formatTxt()

```
int TypeFloat32::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

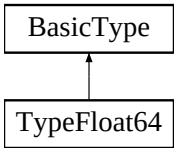
The documentation for this class was generated from the following file:

- system/datatype.h

## 4.50 TypeFloat64 Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.50.1 Detailed Description

definition of class TypeFloat64,please refer to BasicType,it's same.

### 4.50.2 Constructor & Destructor Documentation

#### 4.50.2.1 TypeFloat64()

```
TypeFloat64::TypeFloat64 (
            TypeCode typecode = FLOAT64_TC,
            int64_t typesize = sizeof(double) )  [inline]
```

constructor.

### 4.50.3 Member Function Documentation

#### 4.50.3.1 cmpEQ()

```
bool TypeFloat64::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.50.3.2 cmpGE()

```
bool TypeFloat64::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.50.3.3 cmpGT()

```
bool TypeFloat64::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.50.3.4 cmpLE()

```
bool TypeFloat64::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.50.3.5 cmpLT()

```
bool TypeFloat64::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.50.3.6 copy()

```
int TypeFloat64::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.50.3.7 formatBin()

```
int TypeFloat64::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.50.3.8 formatTxt()

```
int TypeFloat64::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

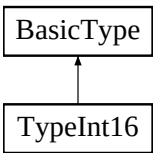
The documentation for this class was generated from the following file:

- system/datatype.h

# 4.51 TypeInt16 Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.51.1 Detailed Description

definition of class TypeInt16,please refer to BasicType,it's same.

### 4.51.2 Constructor & Destructor Documentation

#### 4.51.2.1 TypeInt16()

```
TypeInt16::TypeInt16 (
            TypeCode typecode = INT16_TC,
            int64_t typesize = sizeof(int16_t) )  [inline]
```

constructor.

### 4.51.3 Member Function Documentation

#### 4.51.3.1 cmpEQ()

```
bool TypeInt16::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.51.3.2 cmpGE()

```
bool TypeInt16::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.51.3.3 cmpGT()

```
bool TypeInt16::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.51.3.4 cmpLE()

```
bool TypeInt16::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.51.3.5 cmpLT()

```
bool TypeInt16::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.51.3.6 copy()

```
int TypeInt16::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.51.3.7 formatBin()

```
int TypeInt16::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.51.3.8 formatTxt()

```
int TypeInt16::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

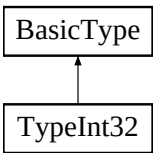
The documentation for this class was generated from the following file:

- system/datatype.h

## 4.52 TypeInt32 Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.52.1 Detailed Description

definition of class TypeInt32,please refer to BasicType,it's same.

### 4.52.2 Constructor & Destructor Documentation

#### 4.52.2.1 TypeInt32()

```
TypeInt32::TypeInt32 (
            TypeCode typecode = INT32_TC,
            int64_t typesize = sizeof(int32_t) )  [inline]
```

constructor.

### 4.52.3 Member Function Documentation

#### 4.52.3.1 cmpEQ()

```
bool TypeInt32::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.52.3.2 cmpGE()

```
bool TypeInt32::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.52.3.3 cmpGT()

```
bool TypeInt32::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.52.3.4 cmpLE()

```
bool TypeInt32::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.52.3.5 cmpLT()

```
bool TypeInt32::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.52.3.6 copy()

```
int TypeInt32::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.52.3.7 formatBin()

```
int TypeInt32::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.52.3.8 formatTxt()

```
int TypeInt32::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

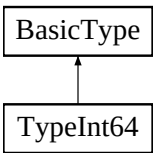
The documentation for this class was generated from the following file:

- system/datatype.h

# 4.53 TypeInt64 Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.53.1 Detailed Description

definition of class TypeInt64,please refer to BasicType,it's same.

### 4.53.2 Constructor & Destructor Documentation

#### 4.53.2.1 TypeInt64()

```
TypeInt64::TypeInt64 (
            TypeCode typecode = INT64_TC,
            int64_t typesize = sizeof(int64_t) )  [inline]
```

constructor.

### 4.53.3 Member Function Documentation

#### 4.53.3.1 cmpEQ()

```
bool TypeInt64::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.53.3.2 cmpGE()

```
bool TypeInt64::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.53.3.3 cmpGT()

```
bool TypeInt64::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.53.3.4 cmpLE()

```
bool TypeInt64::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.53.3.5 cmpLT()

```
bool TypeInt64::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.53.3.6 copy()

```
int TypeInt64::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.53.3.7 formatBin()

```
int TypeInt64::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.53.3.8 formatTxt()

```
int TypeInt64::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

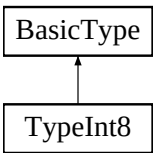
The documentation for this class was generated from the following file:

- system/datatype.h

## 4.54 TypeInt8 Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.54.1 Detailed Description

definition of class TypeInt8,please refer to BasicType,it's same.

### 4.54.2 Constructor & Destructor Documentation

#### 4.54.2.1 TypeInt8()

```
TypeInt8::TypeInt8 (
            TypeCode typecode = INT8_TC,
            int64_t typesize = sizeof(int8_t) )  [inline]
```

constructor.

### 4.54.3 Member Function Documentation

#### 4.54.3.1 cmpEQ()

```
bool TypeInt8::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.54.3.2 cmpGE()

```
bool TypeInt8::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.54.3.3 cmpGT()

```
bool TypeInt8::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.54.3.4 cmpLE()

```
bool TypeInt8::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.54.3.5 cmpLT()

```
bool TypeInt8::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.54.3.6 copy()

```
int TypeInt8::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.54.3.7 formatBin()

```
int TypeInt8::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.54.3.8 formatTxt()

```
int TypeInt8::formatTxt (
            void * dest,
            void * data )  [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from BasicType.

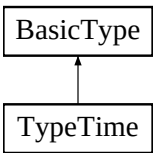
The documentation for this class was generated from the following file:

- system/datatype.h

# 4.55 TypeTime Class Reference

`#include <datatype.h>`

Inheritance 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

### 4.55.1 Detailed Description

definition of class TypeTime,please refer to BasicType,it's same.

### 4.55.2 Constructor & Destructor Documentation

#### 4.55.2.1 TypeTime()

```
TypeTime::TypeTime (
            TypeCode typecode = TIME_TC,
            int64_t typesize = sizeof(time_t) )  [inline]
```

constructor.

### 4.55.3 Member Function Documentation

#### 4.55.3.1 cmpEQ()

```
bool TypeTime::cmpEQ (
            void * data1,
            void * data2 )  [inline], [virtual]
```

equal to.

Reimplemented from BasicType.

#### 4.55.3.2 cmpGE()

```
bool TypeTime::cmpGE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than or equal to

Reimplemented from BasicType.

#### 4.55.3.3 cmpGT()

```
bool TypeTime::cmpGT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

greater than.

Reimplemented from BasicType.

#### 4.55.3.4 cmpLE()

```
bool TypeTime::cmpLE (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than or equal to.

Reimplemented from BasicType.

#### 4.55.3.5 cmpLT()

```
bool TypeTime::cmpLT (
            void * data1,
            void * data2 )  [inline], [virtual]
```

less than.

Reimplemented from BasicType.

#### 4.55.3.6 copy()

```
int TypeTime::copy (
            void * dest,
            void * data )  [inline], [virtual]
```

copy from data to dest.

Reimplemented from BasicType.

#### 4.55.3.7 formatBin()

```
int TypeTime::formatBin (
            void * dest,
            void * data )  [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from BasicType.

#### 4.55.3.8 formatTxt()

```
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 system/catalog.cc File Reference

```
#include "catalog.h"
```

### Variables

- Catalog g_catalog

### 5.1.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.1.2 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

### 5.1.3 Variable Documentation

#### 5.1.3.1 g_catalog

```
Catalog g_catalog
```

## 5.2 system/catalog.d File Reference

## 5.3 system/catalog.h File Reference

```
#include <vector>
#include <unordered_map>
#include "schema.h"
#include "rowtable.h"
#include "hashindex.h"
#include "pbtreeindex.h"
```

### Classes

- class Catalog

### Variables

- Catalog g_catalog

### 5.3.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.3.2 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

### 5.3.3 Variable Documentation

#### 5.3.3.1 g_catalog

```
Catalog g_catalog  [extern]
```

## 5.4 catalog.h

Go to the documentation of this file.

```
1
23 #ifndef _CATALOG_H
24 #define _CATALOG_H
25
26 #include <vector>
27 #include <unordered_map>
28 #include "schema.h"
29 #include "rowtable.h"
30 #include "hashindex.h"
31 #include "pbtreeindex.h"
32
34 class Catalog {
35   private:
36     std::vector <Object *> cl_id_obj;
37     std::unordered_map <std::string, Object *> cl_name_obj;
39   public:
43     void init(void) {
44         cl_id_obj.push_back(NULL);
45     }
49     bool shut(void);
57     bool createDatabase(const char *name, int64_t & d_id);
66     bool createTable(const char *name, TableType type, int64_t & t_id);
76     bool createColumn(const char *name, ColumnType type,int64_t option_size, int64_t & c_id);
86     bool createIndex(const char *name, IndexType type, Key i_key,int64_t & i_id);
93     bool initDatabase(int64_t d_id);    //  this function will truly init the inherited class
100     bool shutDatabase(int64_t d_id);
107     Object *getObjById(int64_t o_id);
114     Object *getObjByName(char *o_name);
118     void print(void) {
119         for (unsigned int ii = 0; ii < cl_id_obj.size(); ii++) {
120             Object *obj = cl_id_obj[ii];
121             if (obj == NULL)
122                 continue;
123             obj->print();
124         }
125     }
126
127   private:
134     bool initTable(int64_t t_id);
141     bool initColumn(int64_t c_id);
149     bool initIndex(int64_t i_id,int64_t t_id);
156     int64_t registerObj(Object * obj) {
157         cl_id_obj[obj->getOid()] = obj;
158         cl_name_obj.insert(std::pair < std::string,
159                            Object * >(obj->getOname(), obj));
160         return obj->getOid();
161     }
167     int64_t obtainId(void) {
168         int64_t id = cl_id_obj.size();
169         cl_id_obj.push_back(NULL);
170         return id;
171     }
172 };  // class Catalog
173
174 extern Catalog g_catalog;
175 #endif
```

## 5.5 system/datatype.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <time.h>
```

### 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 TypeCode

```
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 datatype.h

Go to the documentation of this file.

```
1
12 #ifndef _DATATYPE_H
13 #define _DATATYPE_H
14
15 #include <stdio.h>
16 #include <stdint.h>
17 #include <string.h>
18 #include <time.h>
19
21 enum TypeCode {
22     INVID_TC = 0,
23     INT8_TC,
24     INT16_TC,
25     INT32_TC,
26     INT64_TC,
27     FLOAT32_TC,
28     FLOAT64_TC,
29     CHARN_TC,
30     DATE_TC,
31     TIME_TC,
32     DATETIME_TC,
33     MAXTYPE_TC
34 };
35
37 class BasicType {
38   protected:
39     TypeCode b_type_code;
40     int64_t  b_type_size;
41   public:
45     BasicType(TypeCode typecode, int64_t typesize) {
46         b_type_code = typecode;
47         b_type_size = typesize;
48     }
52     virtual ~BasicType () {}
56     virtual int copy(void *dest, void *data) {
57         printf("[BasicType][ERROR][copy]: not support!\n");
58         return -1;
59     }
63     virtual bool cmpLT(void *data1, void *data2) {
64         printf("[BasicType][ERROR][cmpLT]: not support!\n");
65         return false;
66     }
70     virtual bool cmpEQ(void *data1, void *data2) {
71         printf("[BasicType][ERROR][cmpEQ]: not support!\n");
72         return false;
73     }
77     virtual bool cmpLE(void *data1, void *data2) {
78         printf("[BasicType][ERROR][cmpLT]: not support!\n");
79         return false;
80     }
84     virtual bool cmpGT(void *data1, void *data2) {
```

```
85          printf("[BasicType][ERROR][cmpGT: not support!\n");
86          return false;
87      }
91      virtual bool cmpGE(void *data1, void *data2) {
92          printf("[BasicType][ERROR][cmpGE]: not support!\n");
93          return false;
94      }
98      virtual int formatTxt(void *dest, void *data) {
99          printf("[BasicType][ERROR][formatTxt]: not support!\n");
100         return -2;
101     }
105     virtual int formatBin(void *dest, void *data) {
106         printf("[BasicType][ERROR][formatBin]: not support!\n");
107         return -3;
108     }
112     virtual int64_t getTypeSize(void) {
113         return b_type_size;
114     }
118     virtual TypeCode getTypeCode(void) {
119         return b_type_code;
120     }
121 };
122
124 class TypeInt8:public BasicType {
125   public:
129     TypeInt8(TypeCode typecode = INT8_TC, int64_t typesize = sizeof(int8_t)):BasicType(typecode,
130              typesize)
131     {
132     }
136     int copy(void *dest, void *data) {
137         *(int8_t *) dest = *(int8_t *) data;
138         return b_type_size;
139     }
140     int formatTxt(void *dest, void *data) {
141         return sprintf((char *) dest, "%d", (int) (*(int8_t *) data));
142     }
143     int formatBin(void *dest, void *data) {
144         int tmp;
145         sscanf((char *) data, "%d", &tmp);
146         if (tmp < 128 && tmp >= -128) {
147             *(int8_t *) dest = tmp;
148             return b_type_size;
149         } else {
150             printf("[TypeInt8][ERROR][formatBin]: data exceed range!\n");
151             return -1;
152         }
153     }
154     bool cmpLT(void *data1, void *data2) {
155         return *(int8_t *) data1 < *(int8_t *) data2;
156     }
157     bool cmpLE(void *data1, void *data2) {
158         return *(int8_t *) data1 <= *(int8_t *) data2;
159     }
160     bool cmpEQ(void *data1, void *data2) {
161         return *(int8_t *) data1 == *(int8_t *) data2;
162     }
163     bool cmpGT(void *data1, void *data2) {
164         return *(int8_t *) data1 > *(int8_t *) data2;
165     }
166     bool cmpGE(void *data1, void *data2) {
167         return *(int8_t *) data1 >= *(int8_t *) data2;
168     }
169 };
170
172 class TypeInt16:public BasicType {
173   public:
177   TypeInt16(TypeCode typecode = INT16_TC, int64_t typesize = sizeof(int16_t)):BasicType(typecode,
178              typesize)
179     {
180     }
184     int copy(void *dest, void *data) {
185         *(int16_t *) dest = *(int16_t *) data;
186         return b_type_size;
187     }
188     int formatTxt(void *dest, void *data) {
189         return sprintf((char *) dest, "%d", (int) (*(int16_t *) data));
190     }
191     int formatBin(void *dest, void *data) {
192         int tmp;
193         sscanf((char *) data, "%d", &tmp);
194         if (tmp < (1 << 16) && tmp >= -(1 << 16)) {
195             *(int16_t *) dest = tmp;
196             return b_type_size;
197         } else {
198             printf("[TypeInt16][ERROR][formatBin]: data exceed range!\n");
199             return -1;
200         }
```

```
201     }
202     bool cmpLT(void *data1, void *data2) {
203         return *(int16_t *) data1 < *(int16_t *) data2;
204     }
205     bool cmpLE(void *data1, void *data2) {
206         return *(int16_t *) data1 <= *(int16_t *) data2;
207     }
208     bool cmpEQ(void *data1, void *data2) {
209         return *(int16_t *) data1 == *(int16_t *) data2;
210     }
211     bool cmpGT(void *data1, void *data2) {
212         return *(int16_t *) data1 > *(int16_t *) data2;
213     }
214     bool cmpGE(void *data1, void *data2) {
215         return *(int16_t *) data1 >= *(int16_t *) data2;
216     }
217 };
218
220 class TypeInt32:public BasicType {
221   public:
225   TypeInt32(TypeCode typecode = INT32_TC, int64_t typesize = sizeof(int32_t)):BasicType(typecode,
226               typesize)
227     {
228     }
232     int copy(void *dest, void *data) {
233         *(int32_t *) dest = *(int32_t *) data;
234         return b_type_size;
235     }
236     int formatTxt(void *dest, void *data) {
237         return sprintf((char *) dest, "%d", (int) (*(int32_t *) data));
238     }
239     int formatBin(void *dest, void *data) {
240         int tmp;
241         sscanf((char *) data, "%d", &tmp);
242         *(int32_t *) dest = tmp;
243         return b_type_size;
244     }
245     bool cmpLT(void *data1, void *data2) {
246         return *(int32_t *) data1 < *(int32_t *) data2;
247     }
248     bool cmpLE(void *data1, void *data2) {
249         return *(int32_t *) data1 <= *(int32_t *) data2;
250     }
251     bool cmpEQ(void *data1, void *data2) {
252         return *(int32_t *) data1 == *(int32_t *) data2;
253     }
254     bool cmpGT(void *data1, void *data2) {
255         return *(int32_t *) data1 > *(int32_t *) data2;
256     }
257     bool cmpGE(void *data1, void *data2) {
258         return *(int32_t *) data1 >= *(int32_t *) data2;
259     }
260 };
261
263 class TypeInt64:public BasicType {
264   public:
268   TypeInt64(TypeCode typecode = INT64_TC, int64_t typesize = sizeof(int64_t)):BasicType(typecode,
269               typesize)
270     {
271     }
275     int copy(void *dest, void *data) {
276         *(int64_t *) dest = *(int64_t *) data;
277         return b_type_size;
278     }
279     int formatTxt(void *dest, void *data) {
280         return sprintf((char *) dest, "%ld",
281                        (int64_t) (*(int64_t *) data));
282     }
283     int formatBin(void *dest, void *data) {
284         int64_t tmp;
285         sscanf((char *) data, "%ld", &tmp);
286         *(int64_t *) dest = tmp;
287         return b_type_size;
288     }
289     bool cmpLT(void *data1, void *data2) {
290         return *(int64_t *) data1 < *(int64_t *) data2;
291     }
292     bool cmpLE(void *data1, void *data2) {
293         return *(int64_t *) data1 <= *(int64_t *) data2;
294     }
295     bool cmpEQ(void *data1, void *data2) {
296         return *(int64_t *) data1 == *(int64_t *) data2;
297     }
298     bool cmpGT(void *data1, void *data2) {
299         return *(int64_t *) data1 > *(int64_t *) data2;
300     }
301     bool cmpGE(void *data1, void *data2) {
```

```
302         return *(int64_t *) data1 >= *(int64_t *) data2;
303     }
304 };
305
307 class TypeFloat32:public BasicType {
308   public:
312   TypeFloat32(TypeCode typecode = FLOAT32_TC, int64_t typesize = sizeof(float)):BasicType(typecode,
313               typesize)
314     {
315     }
319     int copy(void *dest, void *data) {
320         *(float *) dest = *(float *) data;
321         return b_type_size;
322     }
323     int formatTxt(void *dest, void *data) {
324         return sprintf((char *) dest, "%f", (float) (*(float *) data));
325     }
326     int formatBin(void *dest, void *data) {
327         float tmp;
328         sscanf((char *) data, "%f", &tmp);
329         *(float *) dest = tmp;
330         return b_type_size;
331     }
332     bool cmpLT(void *data1, void *data2) {
333         return *(float *) data1 < *(float *) data2;
334     }
335     bool cmpLE(void *data1, void *data2) {
336         return *(float *) data1 <= *(float *) data2;
337     }
338     bool cmpEQ(void *data1, void *data2) {
339         return *(float *) data1 == *(float *) data2;
340     }
341     bool cmpGT(void *data1, void *data2) {
342         return *(float *) data1 > *(float *) data2;
343     }
344     bool cmpGE(void *data1, void *data2) {
345         return *(float *) data1 >= *(float *) data2;
346     }
347 };
348
350 class TypeFloat64:public BasicType {
351   public:
355   TypeFloat64(TypeCode typecode = FLOAT64_TC, int64_t typesize = sizeof(double)):BasicType(typecode,
356               typesize)
357     {
358     }
362     int copy(void *dest, void *data) {
363         *(double *) dest = *(double *) data;
364         return b_type_size;
365     }
366     int formatTxt(void *dest, void *data) {
367         return sprintf((char *) dest, "%lf", (double) (*(double *) data));
368     }
369     int formatBin(void *dest, void *data) {
370         double tmp;
371         sscanf((char *) data, "%lf", &tmp);
372         *(double *) dest = tmp;
373         return b_type_size;
374     }
375     bool cmpLT(void *data1, void *data2) {
376         return *(double *) data1 < *(double *) data2;
377     }
378     bool cmpLE(void *data1, void *data2) {
379         return *(double *) data1 <= *(double *) data2;
380     }
381     bool cmpEQ(void *data1, void *data2) {
382         return *(double *) data1 == *(double *) data2;
383     }
384     bool cmpGT(void *data1, void *data2) {
385         return *(double *) data1 > *(double *) data2;
386     }
387     bool cmpGE(void *data1, void *data2) {
388         return *(double *) data1 >= *(double *) data2;
389     }
390 };
391
393 class TypeCharN:public BasicType {
394   public:
398     TypeCharN(int64_t typesize):BasicType(CHARN_TC, typesize) {
399     }
400     TypeCharN(TypeCode typecode = CHARN_TC, int64_t typesize = 32):BasicType(typecode,
401               typesize)
402     {
403     }
407     int copy(void *dest, void *data) {
408         strncpy((char *) dest, (char *) data, b_type_size);
409         return b_type_size;
```

```
410     }
411     int formatTxt(void *dest, void *data) {
412         strncpy((char *) dest, (char *) data, b_type_size);
413         return b_type_size;
414     }
415     int formatBin(void *dest, void *data) {
416         strncpy((char *) dest, (char *) data, b_type_size);
417         return b_type_size;
418     }
419     bool cmpLT(void *data1, void *data2) {
420         return strncmp((char *) data1, (char *) data2,
421                        b_type_size) < 0 ? true : false;
422     }
423     bool cmpLE(void *data1, void *data2) {
424         return strncmp((char *) data1, (char *) data2,
425                        b_type_size) <= 0 ? true : false;
426     }
427     bool cmpEQ(void *data1, void *data2) {
428         return strncmp((char *) data1, (char *) data2,
429                        b_type_size) == 0 ? true : false;
430     }
431     bool cmpGT(void *data1, void *data2) {
432         return strncmp((char *) data1, (char *) data2,
433                        b_type_size) > 0 ? true : false;
434     }
435     bool cmpGE(void *data1, void *data2) {
436         return strncmp((char *) data1, (char *) data2,
437                        b_type_size) >= 0 ? true : false;
438     }
439 };
440
442 class TypeDate:public BasicType {
443   public:
447   TypeDate(TypeCode typecode = DATE_TC, int64_t typesize = sizeof(time_t)):BasicType(typecode,
448               typesize)
449     {
450     }
454     int copy(void *dest, void *data) {
455         *(time_t *) dest = *(time_t *) data;
456         return b_type_size;
457     }
458     int formatTxt(void *dest, void *data) {
459         struct tm *tt = localtime((time_t *) data);
460         return strftime((char *) dest, 32, "%Y-%m-%d", tt);
461     }
462     int formatBin(void *dest, void *data) {
463         struct tm tt = { 0 };
464         strptime((char *) data, "%Y-%m-%d", &tt);
465         time_t et = mktime(&tt);
466         *(time_t *) dest = et;
467         return b_type_size;
468     }
469     bool cmpLT(void *data1, void *data2) {
470         return *(time_t *) data1 < *(time_t *) data2;
471     }
472     bool cmpLE(void *data1, void *data2) {
473         return *(time_t *) data1 <= *(time_t *) data2;
474     }
475     bool cmpEQ(void *data1, void *data2) {
476         return *(time_t *) data1 == *(time_t *) data2;
477     }
478     bool cmpGT(void *data1, void *data2) {
479         return *(time_t *) data1 > *(time_t *) data2;
480     }
481     bool cmpGE(void *data1, void *data2) {
482         return *(time_t *) data1 >= *(time_t *) data2;
483     }
484 };
485
487 class TypeTime:public BasicType {
488   public:
492     TypeTime(TypeCode typecode = TIME_TC, int64_t typesize = sizeof(time_t)):BasicType(typecode,
493               typesize)
494     {
495     }
499     int copy(void *dest, void *data) {
500         *(time_t *) dest = *(time_t *) data;
501         return b_type_size;
502     }
503     int formatTxt(void *dest, void *data) {
504         struct tm *tt = localtime((time_t *) data);
505         return strftime((char *) dest, 32, "%H:%M:%S", tt);
506     }
507     int formatBin(void *dest, void *data) {
508         struct tm tt = { 0 };
509         strptime((char *) data, "%H:%M:%S", &tt);
510         time_t et = mktime(&tt);
```

```
511         *(time_t *) dest = et;
512         return b_type_size;
513     }
514     bool cmpLT(void *data1, void *data2) {
515         return *(time_t *) data1 < *(time_t *) data2;
516     }
517     bool cmpLE(void *data1, void *data2) {
518         return *(time_t *) data1 <= *(time_t *) data2;
519     }
520     bool cmpEQ(void *data1, void *data2) {
521         return *(time_t *) data1 == *(time_t *) data2;
522     }
523     bool cmpGT(void *data1, void *data2) {
524         return *(time_t *) data1 > *(time_t *) data2;
525     }
526     bool cmpGE(void *data1, void *data2) {
527         return *(time_t *) data1 >= *(time_t *) data2;
528     }
529 };
530
532 class TypeDateTime:public BasicType {
533   public:
537     TypeDateTime(TypeCode typecode = DATETIME_TC, int64_t typesize = sizeof(time_t)):BasicType(typecode,
538                 typesize)
539     {
540     }
544     int copy(void *dest, void *data) {
545         *(time_t *) dest = *(time_t *) data;
546         return b_type_size;
547     }
548     int formatTxt(void *dest, void *data) {
549         struct tm *tt = localtime((time_t *) data);
550         return strftime((char *) dest, 32, "%Y-%m-%d %H:%M:%S", tt);
551     }
552     int formatBin(void *dest, void *data) {
553         struct tm tt = { 0 };
554         strptime((char *) data, "%Y-%m-%d %H:%M:%S", &tt);
555         time_t et = mktime(&tt);
556         *(time_t *) dest = et;
557         return b_type_size;
558     }
559     bool cmpLT(void *data1, void *data2) {
560         return *(time_t *) data1 < *(time_t *) data2;
561     }
562     bool cmpLE(void *data1, void *data2) {
563         return *(time_t *) data1 <= *(time_t *) data2;
564     }
565     bool cmpEQ(void *data1, void *data2) {
566         return *(time_t *) data1 == *(time_t *) data2;
567     }
568     bool cmpGT(void *data1, void *data2) {
569         return *(time_t *) data1 > *(time_t *) data2;
570     }
571     bool cmpGE(void *data1, void *data2) {
572         return *(time_t *) data1 >= *(time_t *) data2;
573     }
574 };
575 #endif
```

## 5.7 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"
```

### 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

### 5.7.1 Detailed Description

**Author**

Shimin Chen chensm@ict.ac.cn

**Version**

0.1

### 5.7.2 Description

This file provides the error handling and logging utility.

### 5.7.3 Macro Definition Documentation

#### 5.7.3.1 EL_TOTAL_FILES

```
#define EL_TOTAL_FILES (sizeof(EL_src_file_name)/sizeof(char*))
```

### 5.7.4 Variable Documentation

#### 5.7.4.1 EL_src_file_name

```
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
}
```

#### 5.7.4.2 thread_el

```
ErrorLog _Thread_local* thread_el = NULL
```

## 5.8 system/errorlog.d File Reference

## 5.9 system/errorlog.h File Reference

```
#include <pthread.h>
#include <stdio.h>
#include <string>
#include <unordered_map>
#include <time.h>
```

### 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

### 5.9.1 Detailed Description

**Author**

Shimin Chen chensm@ict.ac.cn

**Version**

0.1

### 5.9.2 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, ...); EL_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();

### 5.9.3 Macro Definition Documentation

#### 5.9.3.1 _Thread_local

```
#define _Thread_local __thread
```

#### 5.9.3.2 EL_ASSERT

```
#define EL_ASSERT(
              t )
```

#### 5.9.3.3 EL_BAD_FILEID

```
#define EL_BAD_FILEID (99999999)
```

#### 5.9.3.4 EL_DEBUG

```
#define EL_DEBUG 1
```

#### 5.9.3.5 EL_ERRCODE

```
#define EL_ERRCODE( ) (thread_el->getErrorCode())
```

#### 5.9.3.6 EL_ERRMSG

```
#define EL_ERRMSG( ) (thread_el->getErrorMsg())
```

#### 5.9.3.7 EL_ERROR

```
#define EL_ERROR 4
```

#### 5.9.3.8 EL_ERROR_CODE

```
#define EL_ERROR_CODE(
              fileid,
              lineno ) ((fileid)*100000 + (lineno))
```

#### 5.9.3.9 EL_GET_FILEID

```
#define EL_GET_FILEID(
              errcode ) ((errcode)/100000)
```

#### 5.9.3.10 EL_GET_FILENAME

```
#define EL_GET_FILENAME(
              errcode )  (ErrorLog::id2Name(EL_GET_FILEID(errcode)))
```

#### 5.9.3.11 EL_GET_LINENO

```
#define EL_GET_LINENO(
              errcode ) ((errcode)%100000)
```

#### 5.9.3.12 EL_INFO

```
#define EL_INFO 2
```

#### 5.9.3.13 EL_LEVEL_COMPILE

```
#define EL_LEVEL_COMPILE EL_INFO
```

#### 5.9.3.14 EL_LOG_DEBUG

```
#define EL_LOG_DEBUG(
              ...  )
```

#### 5.9.3.15 EL_LOG_ERROR

```
#define EL_LOG_ERROR(
               ... )   thread_el->log(EL_ERROR,__FILE__,__LINE__,__VA_ARGS__)
```

#### 5.9.3.16 EL_LOG_INFO

```
#define EL_LOG_INFO(
               ... )
```

**Value:**

```
do{ if (EL_INFO>=ErrorLog::el_level) \
       thread_el->log(EL_INFO,__FILE__,__LINE__,__VA_ARGS__);}while(0)
```

#### 5.9.3.17 EL_LOG_SERIOUS

```
#define EL_LOG_SERIOUS(
               ... )   thread_el->log(EL_SERIOUS,__FILE__,__LINE__,__VA_ARGS__)
```

#### 5.9.3.18 EL_LOG_WARN

```
#define EL_LOG_WARN(
               ... )
```

**Value:**

```
do{ if (EL_WARN>=ErrorLog::el_level) \
       thread_el->log(EL_WARN,__FILE__,__LINE__,__VA_ARGS__);}while(0)
```

#### 5.9.3.19 EL_OK

```
#define EL_OK (0)
```

#### 5.9.3.20 EL_RESET

```
#define EL_RESET( ) (thread_el->reset())
```

#### 5.9.3.21 EL_SERIOUS

```
#define EL_SERIOUS 5
```

#### 5.9.3.22 EL_WARN

```
#define EL_WARN 3
```

### 5.9.4 Variable Documentation

#### 5.9.4.1 EL_src_file_name

```
const char* EL_src_file_name[]  [extern]
```

#### 5.9.4.2 thread_el

```
_Thread_local ErrorLog* thread_el  [extern]
```

## 5.10 errorlog.h

Go to the documentation of this file.

```
1
78 #ifndef _ERRORLOG_H
79 #define _ERRORLOG_H
80
81 #include <pthread.h>
82 #include <stdio.h>
83 #include <string>
84 #include <unordered_map>
85 #include <time.h>
86
87 /* ------------------------------------------------------------------------ */
88 /* Error Logging Macros                                                     */
89 /* ------------------------------------------------------------------------ */
90
91 #define EL_DEBUG          1
92 #define EL_INFO           2
93 #define EL_WARN           3
94 #define EL_ERROR          4
95 #define EL_SERIOUS        5
96
97 // errors < this level will not be compiled into the executable
98 // this can be from EL_DEBUG to EL_ERROR.
99 // Note that we will always report any EL_ERROR and EL_SERIOUS messages.
100
101 #ifdef DEBUG
102 #define EL_LEVEL_COMPILE  EL_DEBUG
103 #else
104 #define EL_LEVEL_COMPILE  EL_INFO
105 #endif
106
107 #if EL_DEBUG >= EL_LEVEL_COMPILE
108 #define EL_LOG_DEBUG(...)  \
```

```
109 do{ if (EL_DEBUG>=ErrorLog::el_level) \
110         thread_el->log(EL_DEBUG,__FILE__,__LINE__,__VA_ARGS__);}while(0)
111 #else
112 #define EL_LOG_DEBUG(...)
113 #endif
114
115 #if EL_INFO >= EL_LEVEL_COMPILE
116 #define EL_LOG_INFO(...)  \
117 do{ if (EL_INFO>=ErrorLog::el_level) \
118         thread_el->log(EL_INFO,__FILE__,__LINE__,__VA_ARGS__);}while(0)
119 #else
120 #define EL_LOG_INFO(...)
121 #endif
122
123 #if EL_WARN >= EL_LEVEL_COMPILE
124 #define EL_LOG_WARN(...)  \
125 do{ if (EL_WARN>=ErrorLog::el_level) \
126         thread_el->log(EL_WARN,__FILE__,__LINE__,__VA_ARGS__);}while(0)
127 #else
128 #define EL_LOG_WARN(...)
129 #endif
130
131 #define EL_LOG_ERROR(...)  \
132         thread_el->log(EL_ERROR,__FILE__,__LINE__,__VA_ARGS__)
133
134 #define EL_LOG_SERIOUS(...)  \
135         thread_el->log(EL_SERIOUS,__FILE__,__LINE__,__VA_ARGS__)
136
137 #define EL_RESET()      (thread_el->reset())
138
139 #define EL_ERRCODE()    (thread_el->getErrorCode())
140
141 #define EL_ERRMSG()     (thread_el->getErrorMsg())
142
143 #if EL_DEBUG >= EL_LEVEL_COMPILE
144 #define EL_ASSERT(exp)    \
145 do{ if (!(exp)) EL_LOG_DEBUG("Assertion failed: %s\n", #exp);} while(0)
146 #else
147 #define EL_ASSERT(t)
148 #endif
149
150 /* ---------------------------------------------------------------------- */
151 /* Error Code                                                             */
152 /* ---------------------------------------------------------------------- */
153
154 // Error code consists of 3 digits of file id and 5 digits of line number
155 // line number cannot
156 #define EL_ERROR_CODE(fileid, lineno)     ((fileid)*100000 + (lineno))
157 #define EL_GET_FILEID(errcode)            ((errcode)/100000)
158 #define EL_GET_LINENO(errcode)            ((errcode)%100000)
159
160 #define EL_OK                             (0)
161 #define EL_BAD_FILEID                     (99999999)
162
163 #define EL_GET_FILENAME(errcode) \
164         (ErrorLog::id2Name(EL_GET_FILEID(errcode)))
165
166 // source file names
167 extern const char *EL_src_file_name[];
168
169 /* ---------------------------------------------------------------------- */
170 /* class ErrorLog                                                         */
171 /* ---------------------------------------------------------------------- */
172 class ErrorLog {
173
174     // ---
175     // class static
176     // ---
177
178   public:
179     static int el_level;
180     static const char *el_level_name[EL_SERIOUS + 1];
182
183   private:
184     static pthread_mutex_t el_lock;
185     static char *el_logfile;
186
187     static FILE *el_fp;
188     static std::unordered_map < std::string, int >*el_name_2_id;
190   public:
197     static void init(int level, const char *logfile);
198
205     static void setLevel(int level);
206
210     static void flushLog(void) {
211         if (el_fp)
212             fflush(el_fp);
```

```
213     } static void closeLog(void) {
217         if (el_fp)
218             fclose(el_fp);
219         el_fp = NULL;
220     }
221
230     static int name2Id(const char *src_name);
231
240     static const char *id2Name(int src_id);
241
242     // ---
243     // instance
244     // ---
245
246   private:
247     char *el_thread_name;
248
249     int el_err_code;
250
251     char *el_msg_buf;
252     int el_msg_cap;
253     char *el_msg_cur;
254
255     char *el_demangle_buf;
256     size_t el_demangle_len;
257     void *el_bt_buffer[256];
258
259
260     time_t el_tloc;
261     struct tm el_tm;
262
277     int getFuncNameGCC(char *bt_symbol);
278
279   public:
287     ErrorLog(const char *thread_name, int msg_cap = 256 * 1024);
288
292     ~ErrorLog();
293
297     void reset();
298
308     void log(int level, const char *src_name, const int lineno, ...);
309
313     int getErrorCode(void) {
314         return el_err_code;
315     }
316
320     const char *getErrorMsg(void) {
321         return el_msg_buf;
322     }
323
324 };                              // ErrorLog
325
326 #ifndef _Thread_local
327 #define _Thread_local __thread
328 #endif
329
330 extern _Thread_local ErrorLog *thread_el;
331 #endif
```

## 5.11 system/executor.cc File Reference

```
#include "executor.h"
```

### Functions

- int64_t allocColBuf (std::vector< int64_t > &col_id, int64_t &size_want, char *&buf_to_alloc)
- int64_t easyAlloc (int64_t size_want, char *&buf_to_alloc)
- int64_t getTupleSize (std::vector< int64_t > &col_id)

### 5.11.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.11.2 DESCRIPTION

definition of executor

### 5.11.3 Function Documentation

#### 5.11.3.1 allocColBuf()

```
int64_t allocColBuf (
            std::vector< int64_t > & col_id,
            int64_t & size_want,
            char *& buf_to_alloc )  [inline]
```

Set record buffer for an input column

**Parameters**

| | |
|---|---|
| *col_id* | reference of input column ID |
| *size_want* | size of buffer, used for return |
| *buf_to_alloc* | buffer to alloc, used for return |

**Return values**

| | |
|---|---|
| *>0* | actual allocated buffer size |
| *<0* | failure |

#### 5.11.3.2 easyAlloc()

```
int64_t easyAlloc (
            int64_t size_want,
            char *& buf_to_alloc )  [inline]
```

**Author**

zyl & dhk

#### 5.11.3.3 getTupleSize()

```
int64_t getTupleSize (
            std::vector< int64_t > & col_id )  [inline]
```

Get size of a tuple according to its column ID

**Parameters**

| | |
|---|---|
| *col_id* | reference of column ID |

**Return values**

| | |
|---|---|
| *size* | of this tuple |

## 5.12 system/executor.d File Reference

## 5.13 system/executor.h File Reference

```
#include "catalog.h"
#include "mymemory.h"
#include <map>
#include <algorithm>
#include <float.h>
```

### Classes

- struct AggreCondition
- struct Condition
- struct Conditions
- class Executor
- class Filter
- class GrAggRecord
- struct GroupbyAggre::group_by_hash
- struct GroupbyAggre::group_by_key
- class GroupbyAggre
- class HashJoin
- class IndexJoin
- class IndexScan

- class Join
- class Operator
- class Orderby
- class Project
- struct RequestColumn
- struct RequestTable
- class ResultTable
- class Scan
- class SelectQuery

## Enumerations

- enum AggregateMethod {
  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

- int64_t easyAlloc (int64_t size_want, char ∗&buf_to_alloc)

### 5.13.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.13.2 DESCRIPTION

definition of executor

### 5.13.3 Enumeration Type Documentation

#### 5.13.3.1 AggregateMethod

`enum AggregateMethod`

aggregate 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 | |

#### 5.13.3.2 CompareMethod

```
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 | |

### 5.13.4 Function Documentation

#### 5.13.4.1 easyAlloc()

```
int64_t easyAlloc (
            int64_t size_want,
            char *& buf_to_alloc )  [inline]
```

Wrapped function of alloc

**Parameters**

| | |
|---|---|
| *size_want* | size wanted to alloc |
| *buf_to_alloc* | reference of new buffer addr |

**Return values**

| | |
|---|---|
| $>0$ | actual allocated size |
| $<0$ | failure |

**Author**

zyl & dhk

## 5.14 executor.h

Go to the documentation of this file.

```
1
12 #ifndef _EXECUTOR_H
13 #define _EXECUTOR_H
14
15 #include "catalog.h"
16 #include "mymemory.h"
17
18 #include <map>
19 #include <algorithm>
20 #include <float.h>
21
23 enum AggregateMethod {
24     NONE_AM = 0,
25     COUNT,
26     SUM,
27     AVG,
28     MAX,
29     MIN,
30     MAX_AM
31 };
32
34 enum CompareMethod {
35     NONE_CM = 0,
36     LT,
37     LE,
38     EQ,
39     NE,
40     GT,
41     GE,
42     LINK,
43     MAX_CM
44 };
45
47 struct RequestColumn {
48     char name[128];
49     AggregateMethod aggregate_method;
50 };
51
53 struct RequestTable {
54     char name[128];
55 };
56
58 struct Condition {
59     RequestColumn column;
60     CompareMethod compare;
61     char value[128];
62 };
63
65 struct Conditions {
66     int condition_num;
67     Condition condition[4];
68 };
69
71 struct AggreCondition {
72     int column_rank;
73     AggregateMethod method;
74 };
75
77 class SelectQuery {
78   public:
79     int64_t database_id;
80     int select_number;
81     RequestColumn select_column[4];
82     int from_number;
```

```
83      RequestTable from_table[4];
84      Conditions where;
85      int groupby_number;
86      RequestColumn groupby[4];
87      Conditions having;
88      int orderby_number;
89      RequestColumn orderby[4];
90  };  // class SelectQuery
91
93  class ResultTable {
94    public:
95      int column_number;
96      BasicType **column_type;
97      char *buffer;
98      int64_t buffer_size;
99      int row_length;
100      int row_number;
101      int row_capicity;
102      int *offset;
103      int offset_size;
104
113      int init(BasicType *col_types[],int col_num,int64_t capicity = 1024);
122      char* getRC(int row, int column);
131      int writeRC(int row, int column, void *data);
136      int print(void);
140      int dump(FILE *fp);
144      int shut(void);
145
152      int append(char* src);
153  };  // class ResultTable
154
155  class Operator {
156    public:
157      char * buffer_from_father;
159          public:
163      Operator() {};
167      virtual      ~Operator() {};
168
174      inline void setBuffer(char * buffer_allocated) {
175        buffer_from_father = buffer_allocated;
176      }
180      inline char * getBuffer() {
181        return buffer_from_father;
182      }
183
189      virtual bool  open    () {};
195      virtual bool  getNext () {};
201      virtual bool  close   () {};
202  };
203
205  class Executor {
206    private:
207      SelectQuery *current_query;
208    public:
209      Operator* root;
210    public:
216      Operator* planner(SelectQuery *query);
217
224      int findCol(char* table_name, char* column_name);
225
233      int exec(SelectQuery *query, ResultTable *result);
234
241      int close();
242
248      int64_t getRank(std::vector < int64_t > &vec, int64_t id);
249  };
250
258  inline int64_t easyAlloc(int64_t size_want, char * & buf_to_alloc);
259
264  class Scan : public Operator {
265    private:
266      Table * scan_table;
267      int64_t total_record;
268      int64_t next_record;
269    public:
273      Scan() {};
277      ~Scan() {};
278
284      void setTable (Table * table) {
285        scan_table = table;
286      }
292      bool  open    ();
298      bool  getNext ();
304      bool  close   ();
305  };
306
310  class IndexScan : public Operator {
```

```
311   private:
312     Table * from;
313     Index * index;
314     IndexType i_type;
315     void * info_ptr;
316     void * current_key;
317     bool key_end;
319   public:
325     void setTabIdx(Table * table, Index * index) {
326       this -> from = table;
327       this -> index = index;
328     }
332     IndexScan() {};
338     IndexScan(Table * table, Index * index) {
339       setTabIdx(table, index);
340     }
344     ~IndexScan() {};
350     bool open();
356     void updateKey(void * search_key);
362     bool getNext();
368     bool close();
369 };
370
375 class Filter : public Operator {
376   private:
377     Operator * child;
378     char * buf_for_child;
379     int64_t child_buf_size;
380     std::vector < int64_t > input_cid;
381     char * filt_pos;
382     int64_t filt_off;
383     CompareMethod cmp_mtd;
384     BasicType * filt_type;
385     bool (*cmp_func) (void *a, void *b, BasicType * data_type);
386     int64_t in_tuple_size;
387     char value[128];
389   private:
393     bool (*cmp_table [MAX_CM])(void * data1, void * data2, BasicType * data_type);
403     static inline bool cmpLT(void *data1, void *data2, BasicType * data_type){
404       return data_type -> cmpLT(data1, data2);
405     }
414     static inline bool cmpLE(void *data1, void *data2, BasicType * data_type){
415       return data_type -> cmpLE(data1, data2);
416     }
425     static inline bool cmpEQ(void *data1, void *data2, BasicType * data_type){
426       return data_type -> cmpEQ(data1, data2);
427     }
436     static inline bool cmpNE(void *data1, void *data2, BasicType * data_type){
437       return !(data_type -> cmpEQ(data1, data2));
438     }
447     static inline bool cmpGT(void *data1, void *data2, BasicType * data_type){
448       return data_type -> cmpGT(data1, data2);
449     }
458     static inline bool cmpGE(void *data1, void *data2, BasicType * data_type){
459       return data_type -> cmpGE(data1, data2);
460     }
467     void setFiltCond(int64_t filt_rank, CompareMethod cmp_mtd, char * value);
471     void initCmpFunc() {
472       cmp_table[NONE_CM] = NULL;
473       cmp_table[LT     ] = &cmpLT;
474       cmp_table[LE     ] = &cmpLE;
475       cmp_table[EQ     ] = &cmpEQ;
476       cmp_table[NE     ] = &cmpNE;
477       cmp_table[GT     ] = &cmpGT;
478       cmp_table[GE     ] = &cmpGE;
479       cmp_table[LINK   ] = NULL;
480     }
481
482   public:
487     void setChild(Operator * child) {
488       this -> child = child;
489     }
498     bool setColumn(int64_t c_id [], int64_t num_column, int64_t filt_rank, CompareMethod cmp_mtd, char *
      value) {
499       if (cmp_mtd == NONE_CM || cmp_mtd == LINK || cmp_mtd == MAX_CM) {
500         return false;
501       }
502
503       for (int64_t i = 0; i < num_column; i ++){
504         input_cid.push_back(c_id[i]);
505       }
506       setFiltCond(filt_rank, cmp_mtd, value);
507
508       return true;
509     }
519     bool setColumn(std::vector < int64_t > input_cid, int64_t filt_rank, CompareMethod cmp_mtd, char *
      value) {
```

```
520       if (cmp_mtd == NONE_CM || cmp_mtd == LINK || cmp_mtd == MAX_CM) {
521         return false;
522       }
523
524       this -> input_cid = input_cid;
525       setFiltCond(filt_rank, cmp_mtd, value);
526
527       return true;
528     }
529
533     Filter() {
534       initCmpFunc();
535     };
539     ~Filter() {};
548     Filter(Operator * child, std::vector < int64_t > input_cid, int64_t filt_rank, CompareMethod
      cmp_mtd, char * value) {
549       initCmpFunc();
550       setChild(child);
551       setColumn(input_cid, filt_rank, cmp_mtd, value);
552     }
562     Filter(Operator * child, int64_t c_id [], int64_t num_column, int64_t filt_rank, CompareMethod
      cmp_mtd, char * value) {
563       initCmpFunc();
564       setChild(child);
565       setColumn(c_id, num_column, filt_rank, cmp_mtd, value);
566     }
567
573     bool open();
579     bool  getNext ();
585     bool  close   ();
586
587 };
588
590 class Join : public Operator {
591   private:
592     Operator * left;
593     Operator * right;
595     std::vector < int64_t > left_cid;
596     std::vector < int64_t > right_cid;
598     int64_t left_rank;
599     int64_t right_rank;
601   public:
605     Join() {}
609     virtual ~Join() {}
614     void setLeftOp(Operator * lchild) {
615       left = lchild;
616     }
621     void setRightOp(Operator * rchild) {
622       right = rchild;
623     }
628     Operator * & getLeftOp() {
629       return left;
630     }
635     Operator * & getRightOp() {
636       return right;
637     }
645     void setJoinCol(std::vector < int64_t > left_cid, std::vector < int64_t > right_cid, int64_t
      left_rank, int64_t right_rank) {
646       this -> left_cid = left_cid;
647       this -> right_cid = right_cid;
648       this -> left_rank = left_rank;
649       this -> right_rank = right_rank;
650     }
655     std::vector <int64_t> & getLeftCol() {
656       return left_cid;
657     }
662     std::vector <int64_t> & getRightCol() {
663       return right_cid;
664     }
669     int64_t getLeftRank() {
670       return left_rank;
671     }
676     int64_t getRightRank() {
677       return right_rank;
678     }
684     virtual bool open() = 0;
690     virtual bool getNext() = 0;
696     virtual bool close() = 0;
697 };
698
703 class IndexJoin : public Join {
704   private:
705     int64_t left_tuple_size;
706     int64_t right_tuple_size;
708     char * left_buf;
709     char * right_buf;
710     int64_t left_buf_size;
```

```
711     int64_t right_buf_size;
715     void * current_key;
716     bool right_has_next;
718   public:
722     IndexJoin() {}
726     ~IndexJoin() {}
727
735     IndexJoin(std::vector < int64_t > left_cid, std::vector < int64_t > right_cid, int64_t left_rank,
      int64_t right_rank) {
736       setJoinCol(left_cid, right_cid, left_rank, right_rank);
737     }
738
744     bool open();
750     bool getNext();
756     bool close();
757 };
758
764 class HashJoin : public Join {
765   private:
766     char * left_buf;
767     char * right_buf;
769     std::vector <char *> middle_buf_array;
770     int64_t middle_buf_size;
772     int64_t left_tuple_size;
773     int64_t right_tuple_size;
774     int64_t right_buf_size;
776     bool right_has_next;
778     char * right_key_pos;
779     int64_t left_key_off;
780     BasicType * right_key_type;
781     BasicType * left_key_type;
783     char txt_buf [128];
784     std::multimap <std::string, char *> hash_index;
786     // Note: this should be arranged by lexicographic order
787     std::multimap <std::string, char *> ::iterator last_iter;
788     std::multimap <std::string, char *> ::iterator upper_iter;
790   public:
794     HashJoin() {}
798     ~HashJoin() {}
799     /*setIndex(std::vector < int64_t > left_cid, int64_t left_rank) {
800       // Create a Hash Index
801       //hash_index = new HashIndex()
802     }*/
810     HashJoin(std::vector < int64_t > left_cid, std::vector < int64_t > right_cid, int64_t left_rank,
      int64_t right_rank) {
811       setJoinCol(left_cid, right_cid, left_rank, right_rank);
812     }
818     bool open();
824     bool getNext();
830     bool close();
831 };
832
837 class Project : public Operator {
838   private:
839     Operator * child;
840     char * buf_for_child;
842     int64_t self_buf_size;
844     std::vector < int64_t > input_cid;
845     std::vector < int64_t > output_cid;
846     std::vector < int64_t > out_to_in;
848     int64_t in_tuple_size;
849     int64_t in_buf_size;
851     std::vector < BasicType * > input_type;
852     std::vector < int64_t > input_off;
853     std::vector < char * > input_pos;
855     char * output_type;
856     int64_t output_type_size;
857     int64_t output_type_buf_size;
858     bool topid;
860   public:
864     Project() {};
868     ~Project() {
869       g_memory.free(output_type, output_type_buf_size);
870     };
875     void setChild(Operator * child) {
876       this -> child = child;
877     }
882     int64_t getColnum() {
883       return output_cid.size();
884     }
889     BasicType ** getSchema() {
890       return (BasicType **)output_type;
891     }
892
898     void setProjCol(std::vector <int64_t> in_cid, std::vector <int64_t> out_cid) {
899       input_cid = in_cid;
900       output_cid = out_cid;
```

```
901
902       output_type_size = sizeof(BasicType *) * output_cid.size();
903       //printf("set1");
904
905       output_type_buf_size = easyAlloc(output_type_size, output_type);
906       int64_t offset = 0;
907       //printf("set2");
908
909
910       for (int64_t i = 0; i < out_cid.size(); i ++) {
911         // Add output's index from input
912         auto iter = std::find(input_cid.begin(), input_cid.end(), out_cid[i]);
913         out_to_in.push_back(std::distance(input_cid.begin(), iter));
914
915         //printf("set(%d)\n", i);
916         BasicType ** out_type = (BasicType **)(output_type + offset);
917         Column * column = (Column *)(g_catalog.getObjById(output_cid[i]));
918         *out_type = column -> getDataType();
919         offset += sizeof(BasicType *);
920       }
921     }
927     Project(std::vector <int64_t> in_cid, std::vector <int64_t> out_cid) {
928       setProjCol(in_cid, out_cid);
929     }
935     bool open();
941     bool getNext();
947     bool close();
953     bool top();
954 };
955
960 class GrAggRecord {
961   public:
962     char * middle_record;
963     std::vector <int64_t> sum;
964     std::vector <int64_t> count;
971     GrAggRecord(char * middle_record, int64_t num_aggr) {
972       this -> middle_record = middle_record;
973       sum.resize(num_aggr);
974       count.resize(num_aggr);
975     }
976 };
977
982 class GroupbyAggre : public Operator {
983   private:
984     Operator* child;
985     std::vector <int64_t> in_cid;
986     std::vector <int64_t> groupby_rank;
987     std::vector <int64_t> out_cid;
988     std::vector <AggreCondition> conditions;
990     char * buf_for_child;
991     int64_t child_buf_size;
992     int64_t child_tuple_size;
994     int64_t middle_tuple_size;
995     int64_t middle_buf_size;
996     std::vector < char * > middle_buf_array;
998     typedef std::vector < BasicType * > group_by_type_t;
999     group_by_type_t group_by_type;
1000     std::vector < char * > group_by_pos;
1001     std::vector < int64_t > group_by_size;
1002     std::vector < BasicType * > aggr_type;
1003     std::vector < char * > aggr_pos;
1005     // Specify key type
1006     typedef struct group_by_key {
1007       group_by_type_t type_array;
1008       std::vector <char *> value_array;
1009
1010       bool operator == (const group_by_key & k) const {
1011         int64_t count = 0;
1012         for (int64_t i = 0; i < value_array.size(); i ++) {
1013           if (k.type_array[i] -> cmpEQ(value_array[i], k.value_array[i])) {
1014             count ++;
1015           }
1016         }
1017         return count == value_array.size();
1018       }
1019     } group_by_key_t;
1020     // Specify hash function
1021     typedef struct group_by_hash {
1022       size_t operator() (const group_by_key_t &key) const {
1023         std::hash <std::string> hash_func;
1024         size_t hash_val = 0;
1025         char cmp_buffer[128];
1026         for (int64_t i = 0; i < key.value_array.size(); i ++) {
1027           key.type_array[i] -> formatTxt(cmp_buffer, key.value_array[i]);
1028           hash_val ^= hash_func(std::string(cmp_buffer)) + 0xCafeBabe + (hash_val « 3) + (hash_val »
     1);
1029         }
```

```
1030         return hash_val;
1031       }
1032     } group_by_hash_t;
1033     std::unordered_map <group_by_key_t, GrAggRecord *, group_by_hash_t> hash_group;
1034     std::unordered_map <group_by_key_t, GrAggRecord *> ::iterator next_iter;
1042     static void sumInt8(void * sum, void * count, void * x) {
1043       *(int8_t *)sum += *(int8_t *)x;
1044     }
1051     static void sumInt16(void * sum, void * count, void * x) {
1052       *(int16_t *)sum += *(int16_t *)x;
1053     }
1060     static void sumInt32(void * sum, void * count, void * x) {
1061       *(int32_t *)sum += *(int32_t *)x;
1062     }
1069     static void sumInt64(void * sum, void * count, void * x) {
1070       *(int64_t *)sum += *(int64_t *)x;
1071     }
1078     static void sumFloat32(void * sum, void * count, void * x) {
1079       *(float *)sum += *(float *)x;
1080     }
1087     static void sumFloat64(void * sum, void * count, void * x) {
1088       *(double *)sum += *(double *)x;
1089     }
1091     void (*sum_table [MAXTYPE_TC])(void * sum, void * count, void * x);
1092
1099     static void avgInt8(void * sum, void * count, void * x) {
1100       sumInt8(sum, count, x);
1101       *(int64_t *)count += 1;
1102     }
1109     static void avgInt16(void * sum, void * count, void * x) {
1110       sumInt16(sum, count, x);
1111       *(int64_t *)count += 1;
1112     }
1119     static void avgInt32(void * sum, void * count, void * x) {
1120       sumInt32(sum, count, x);
1121       *(int64_t *)count += 1;
1122     }
1129     static void avgInt64(void * sum, void * count, void * x) {
1130       sumInt64(sum, count, x);
1131       *(int64_t *)count += 1;
1132     }
1139     static void avgFloat32(void * sum, void * count, void * x) {
1140       sumFloat32(sum, count, x);
1141       *(int64_t *)count += 1;
1142     }
1149     static void avgFloat64(void * sum, void * count, void * x) {
1150       sumFloat64(sum, count, x);
1151       *(int64_t *)count += 1;
1152     }
1154     void (*avg_table [MAXTYPE_TC])(void * sum, void * count, void * x);
1161     static void maxInt8(void * sum, void * count, void * x) {
1162       *(int8_t *)sum = std::max(*(int8_t *)sum, *(int8_t *)x);
1163     }
1170     static void maxInt16(void * sum, void * count, void * x) {
1171       *(int16_t *)sum = std::max(*(int16_t *)sum, *(int16_t *)x);
1172     }
1179     static void maxInt32(void * sum, void * count, void * x) {
1180       *(int32_t *)sum = std::max(*(int32_t *)sum, *(int32_t *)x);
1181     }
1188     static void maxInt64(void * sum, void * count, void * x) {
1189       *(int64_t *)sum = std::max(*(int64_t *)sum, *(int64_t *)x);
1190     }
1197     static void maxFloat32(void * sum, void * count, void * x) {
1198       *(float *)sum = std::max(*(float *)sum, *(float *)x);
1199     }
1206     static void maxFloat64(void * sum, void * count, void * x) {
1207       *(double *)sum = std::max(*(double *)sum, *(double *)x);
1208     }
1210     void (*max_table [MAXTYPE_TC])(void * sum, void * count, void * x);
1217     static void minInt8(void * sum, void * count, void * x) {
1218       *(int8_t *)sum = std::min(*(int8_t *)sum, *(int8_t *)x);
1219     }
1226     static void minInt16(void * sum, void * count, void * x) {
1227       *(int16_t *)sum = std::min(*(int16_t *)sum, *(int16_t *)x);
1228     }
1235     static void minInt32(void * sum, void * count, void * x) {
1236       *(int32_t *)sum = std::min(*(int32_t *)sum, *(int32_t *)x);
1237     }
1244     static void minInt64(void * sum, void * count, void * x) {
1245       *(int64_t *)sum = std::min(*(int64_t *)sum, *(int64_t *)x);
1246     }
1253     static void minFloat32(void * sum, void * count, void * x) {
1254       *(float *)sum = std::min(*(float *)sum, *(float *)x);
1255     }
1262     static void minFloat64(void * sum, void * count, void * x) {
1263       *(double *)sum = std::min(*(double *)sum, *(double *)x);
1264     }
```

```
1266      void (*min_table [MAXTYPE_TC])(void * sum, void * count, void * x);
1273      static void count(void * sum, void * count, void * x) {
1274        *(int64_t *)count += 1;
1275      }
1276
1277      void (*aggr_method [4])(void * sum, void * count, void * x);
1284      static void initSum(void * sum, void * count) {
1285        *(int64_t *)sum = 0;
1286      }
1292      static void initCount(void * sum, void * count) {
1293        *(int64_t *)count = 0;
1294      }
1300      static void initAvg(void * sum, void * count) {
1301        initSum(sum, count);
1302        *(int64_t *)sum = 0;
1303      }
1309      static void initInt8Min(void * sum, void * count) {
1310        *(int64_t *)sum = INT8_MAX;
1311      }
1317      static void initInt16Min(void * sum, void * count) {
1318        *(int64_t *)sum = INT16_MAX;
1319      }
1325      static void initInt32Min(void * sum, void * count) {
1326        *(int64_t *)sum = INT32_MAX;
1327      }
1333      static void initInt64Min(void * sum, void * count) {
1334        *(int64_t *)sum = INT64_MAX;
1335      }
1341      static void initFloat32Min(void * sum, void * count) {
1342        *(float *)sum = FLT_MAX;
1343      }
1349      static void initFloat64Min(void * sum, void * count) {
1350        *(double *)sum = DBL_MAX;
1351      }
1353      void (*init_min_table [MAXTYPE_TC])(void * sum, void * count);
1359      static void initInt8Max(void * sum, void * count) {
1360        *(int64_t *)sum = INT8_MIN;
1361      }
1367      static void initInt16Max(void * sum, void * count) {
1368        *(int64_t *)sum = INT16_MIN;
1369      }
1375      static void initInt32Max(void * sum, void * count) {
1376        *(int64_t *)sum = INT32_MIN;
1377      }
1383      static void initInt64Max(void * sum, void * count) {
1384        *(int64_t *)sum = INT64_MIN;
1385      }
1391      static void initFloat32Max(void * sum, void * count) {
1392        *(float *)sum = FLT_MIN;
1393      }
1399      static void initFloat64Max(void * sum, void * count) {
1400        *(double *)sum = DBL_MIN;
1401      }
1403      void (*init_max_table [MAXTYPE_TC])(void * sum, void * count);
1404
1405      void (*init_method [4])(void * sum, void * count);
1413      static void finalInt8Sum(void * sum, void * count, void * result) {
1414        *(int8_t *)result = *(int8_t *)sum;
1415      }
1422      static void finalInt16Sum(void * sum, void * count, void * result) {
1423        *(int16_t *)result = *(int16_t *)sum;
1424      }
1430      static void finalInt32Sum(void * sum, void * count, void * result) {
1431        *(int32_t *)result = *(int32_t *)sum;
1432      }
1439      static void finalInt64Sum(void * sum, void * count, void * result) {
1440        *(int64_t *)result = *(int64_t *)sum;
1441      }
1447      static void finalFloat32Sum(void * sum, void * count, void * result) {
1448        *(float *)result = *(float *)sum;
1449      }
1456      static void finalFloat64Sum(void * sum, void * count, void * result) {
1457        *(double *)result = *(double *)sum;
1458      }
1460      void (*final_sum_table [MAXTYPE_TC])(void * sum, void * count, void * result);
1467      static void finalCount(void * sum, void * count, void * result) {
1468        *(int64_t *)result = *(int64_t *)count;
1469      }
1476      static void finalIntAvg(void * sum, void * count, void * result) {
1477        int64_t c = *(int64_t *)count;
1478        int64_t s = *(int64_t *)sum;
1479        *(double *)result = (double)s / (double)c;
1480      }
1487      static void finalFloat32Avg(void * sum, void * count, void * result) {
1488        int64_t c = *(int64_t *)count;
1489        float s = *(float *)sum;
1490        *(double *)result = (double)s / (double)c;
```

```
1491     }
1498     static void finalFloat64Avg(void * sum, void * count, void * result) {
1499       int64_t c = *(int64_t *)count;
1500       double s = *(double *)sum;
1501       *(double *)result = (double)s / (double)c;
1502     }
1504     void (*final_avg_table [MAXTYPE_TC])(void * sum, void * count, void * result);
1505
1506     void (*final_method [4])(void * sum, void * count, void * result);
1508   public:
1513     GroupbyAggre() {
1514       sum_table[INT8_TC] = &sumInt8;
1515       sum_table[INT16_TC] = &sumInt16;
1516       sum_table[INT32_TC] = &sumInt32;
1517       sum_table[INT64_TC] = &sumInt64;
1518       sum_table[FLOAT32_TC] = &sumFloat32;
1519       sum_table[FLOAT64_TC] = &sumFloat64;
1520
1521       avg_table[INT8_TC] = &avgInt8;
1522       avg_table[INT16_TC] = &avgInt16;
1523       avg_table[INT32_TC] = &avgInt32;
1524       avg_table[FLOAT32_TC] = &avgFloat32;
1525       avg_table[FLOAT64_TC] = &avgFloat64;
1526
1527       max_table[INT8_TC] = &maxInt8;
1528       max_table[INT16_TC] = &maxInt16;
1529       max_table[INT32_TC] = &maxInt32;
1530       max_table[INT64_TC] = &maxInt64;
1531       max_table[FLOAT32_TC] = &maxFloat32;
1532       max_table[FLOAT64_TC] = &maxFloat64;
1533
1534       min_table[INT8_TC] = &minInt8;
1535       min_table[INT16_TC] = &minInt16;
1536       min_table[INT32_TC] = &minInt32;
1537       min_table[INT64_TC] = &minInt64;
1538       min_table[FLOAT32_TC] = &minFloat32;
1539       min_table[FLOAT64_TC] = &minFloat64;
1540
1541       init_min_table[INT8_TC] = &initInt8Min;
1542       init_min_table[INT16_TC] = &initInt16Min;
1543       init_min_table[INT32_TC] = &initInt32Min;
1544       init_min_table[FLOAT32_TC] = &initFloat32Min;
1545       init_min_table[FLOAT64_TC] = &initFloat64Min;
1546
1547       init_max_table[INT8_TC] = &initInt8Max;
1548       init_max_table[INT16_TC] = &initInt16Max;
1549       init_max_table[INT32_TC] = &initInt32Max;
1550       init_max_table[FLOAT32_TC] = &initFloat32Max;
1551       init_max_table[FLOAT64_TC] = &initFloat64Max;
1552
1553       final_sum_table[INT8_TC] = &finalInt8Sum;
1554       final_sum_table[INT16_TC] = &finalInt16Sum;
1555       final_sum_table[INT32_TC] = &finalInt32Sum;
1556       final_sum_table[INT64_TC] = &finalInt64Sum;
1557       final_sum_table[FLOAT32_TC] = &finalFloat32Sum;
1558       final_sum_table[FLOAT64_TC] = &finalFloat64Sum;
1559
1560       final_avg_table[INT8_TC] = &finalIntAvg;
1561       final_avg_table[INT16_TC] = &finalIntAvg;
1562       final_avg_table[INT32_TC] = &finalIntAvg;
1563       final_avg_table[INT64_TC] = &finalIntAvg;
1564       final_avg_table[FLOAT32_TC] = &finalFloat32Avg;
1565       final_avg_table[FLOAT64_TC] = &finalFloat64Avg;
1566     };
1570     ~GroupbyAggre() {};
1571
1579     void set (std::vector<int64_t> input_colid, std::vector<int64_t> groupby_rank,
       std::vector<AggreCondition> conditions, std::vector<int64_t> output_colid) {
1580       this -> in_cid = input_colid;
1581       this -> groupby_rank = groupby_rank;
1582       this -> conditions = conditions;
1583       this -> out_cid = output_colid;
1584
1585     }
1590     void setChild(Operator * child) {
1591       this -> child = child;
1592     }
1598     bool  open    ();
1604     bool  getNext ();
1610     bool  close   ();
1611 };
1612
1613 class Orderby : public Operator {
1614   private:
1615     Operator* child;
1616     std::vector<int64_t> colid;
1617     std::vector<int> colrank;
```

```
1618     std::vector<int> coloff;
1619     std::vector<BasicType*> coltype;
1620     int orderby_num;
1621     int64_t tuple_size;
1622     int64_t middle_buf_size;
1623     char* child_buffer;
1624     std::vector<char*> middle_buf_array;
1625     int arrayid;
1626     int64_t self_buf_size;
1627   public:
1631     Orderby() {};
1632
1636     ~Orderby() {};
1637
1643     void set (std::vector<int64_t> input_colid, std::vector<int> orderby_rank) {
1644       colid = input_colid;
1645       colrank = orderby_rank;
1646       orderby_num = colrank.size();
1647     }
1652     void setChild(Operator * child) {
1653       this -> child = child;
1654     }
1660     bool  open    ();
1666     bool  getNext ();
1672     bool  close   ();
1673 };
1674
1675
1676
1677 #endif
```

# 5.15 system/gcc_pf_p3.h File Reference

## Macros

- #define pfld(mem_var) prefetcht0(mem_var)
- #define pfldnta(mem_var) prefetchnta(mem_var)
- #define pfst(mem_var) prefetcht0(mem_var)
- #define pfstnta(mem_var) prefetchnta(mem_var)
- #define prefetchnta(mem_var) __asm__ __volatile__ ("prefetchnta %0": :"m"(mem_var))
- #define prefetcht0(mem_var) __asm__ __volatile__ ("prefetcht0 %0": :"m"(mem_var))
- #define prefetcht1(mem_var) __asm__ __volatile__ ("prefetcht1 %0": :"m"(mem_var))
- #define ptouch(mem_var) __asm__ __volatile__ ("movl %0, %%eax": :"m"(mem_var):"%eax")

## 5.15.1 Macro Definition Documentation

### 5.15.1.1 pfld

```
#define pfld(
              mem_var ) prefetcht0(mem_var)
```

### 5.15.1.2 pfldnta

```
#define pfldnta(
              mem_var ) prefetchnta(mem_var)
```

#### 5.15.1.3 pfst

```
#define pfst(
             mem_var ) prefetcht0(mem_var)
```

#### 5.15.1.4 pfstnta

```
#define pfstnta(
             mem_var ) prefetchnta(mem_var)
```

#### 5.15.1.5 prefetchnta

```
#define prefetchnta(
             mem_var )  __asm__ __volatile__ ("prefetchnta %0":  :"m"(mem_var))
```

#### 5.15.1.6 prefetcht0

```
#define prefetcht0(
             mem_var )  __asm__ __volatile__ ("prefetcht0 %0":  :"m"(mem_var))
```

#### 5.15.1.7 prefetcht1

```
#define prefetcht1(
             mem_var )  __asm__ __volatile__ ("prefetcht1 %0":  :"m"(mem_var))
```

#### 5.15.1.8 ptouch

```
#define ptouch(
             mem_var )  __asm__ __volatile__ ("movl %0, %%eax":  :"m"(mem_var):"%eax")
```

## 5.16 gcc_pf_p3.h

Go to the documentation of this file.

```
1 /* File Name: gcc_pf_p3.h
2  * Author:    Shimin Chen
3  */
4
5 #ifndef _GCC_PREFETCH_P3
6 #define _GCC_PREFETCH_P3
7
8 #define prefetcht0(mem_var)      \
9         __asm__ __volatile__ ("prefetcht0 %0": :"m"(mem_var))
10 #define prefetcht1(mem_var)      \
11         __asm__ __volatile__ ("prefetcht1 %0": :"m"(mem_var))
12 //#define prefetcht2(mem_var)
13 //       __asm__ __volatile__ ("prefetcht2 %0": :"m"(mem_var))
14 #define prefetchnta(mem_var) \
15         __asm__ __volatile__ ("prefetchnta %0": :"m"(mem_var))
16
17 /*
18    T0 (temporal data):
19        prefetch data into all levels of the cache hierarchy.
20        Pentium III processor 1st- or 2nd-level cache.
21        Pentium 4 and Intel Xeon processors 2nd-level cache.
22
23    T1 (temporal data with respect to first level cache)
24        prefetch data into level 2 cache and higher.
25        Pentium III processor 2nd-level cache.
26        Pentium 4 and Intel Xeon processors 2nd-level cache.
27
28    T2 (temporal data with respect to second level cache)
29        prefetch data into level 2 cache and higher.
30        Pentium III processor 2nd-level cache.
31        Pentium 4 and Intel Xeon processors 2nd-level cache.
32
33    NTA (non-temporal data with respect to all cache levels)
34         prefetch data into non-temporal cache structure and
35         into a location close to the processor, minimizing cache pollution.
36         Pentium III processor 1st-level cache
37         Pentium 4 and Intel Xeon processors 2nd-level cache
38  */
39
40 #define pfld(mem_var)      prefetcht0(mem_var)
41 #define pfst(mem_var)      prefetcht0(mem_var)
42 #define pfldnta(mem_var)   prefetchnta(mem_var)
43 #define pfstnta(mem_var)   prefetchnta(mem_var)
44
45 #define ptouch(mem_var) \
46         __asm__ __volatile__ ("movl %0, %%eax": :"m"(mem_var):"%eax")
47
48 #endif /* _GCC_PREFETCH_P3 */
```

## 5.17 system/global.cc File Reference

```
#include "global.h"
```

### Functions

- int global_init ()
- int global_shut ()

### 5.17.1 Function Documentation

#### 5.17.1.1 global_init()

```
int global_init ( )
```

init memory and catalog.

#### 5.17.1.2 global_shut()

```
int global_shut ( )
```

shut down catalog and memory.

## 5.18 system/global.d File Reference

## 5.19 system/global.h File Reference

```
#include "memory.h"
#include "catalog.h"
```

### Macros

- #define BNODE_POINTERS_NUM (16)
- #define GLOBAL_MEMORY_MINIMUM (1L<< 3)
- #define GLOBAL_MEMORY_SIZE (1L<<31)

### Functions

- int global_init ()
- int global_shut ()

### Variables

- Catalog g_catalog
- Memory g_memory

### 5.19.1 Macro Definition Documentation

#### 5.19.1.1 BNODE_POINTERS_NUM

```
#define BNODE_POINTERS_NUM (16)
```

#### 5.19.1.2 GLOBAL_MEMORY_MINIMUM

```
#define GLOBAL_MEMORY_MINIMUM (1L<< 3)
```

#### 5.19.1.3 GLOBAL_MEMORY_SIZE

```
#define GLOBAL_MEMORY_SIZE (1L<<31)
```

### 5.19.2 Function Documentation

#### 5.19.2.1 global_init()

```
int global_init ( )
```

init memory and catalog.

#### 5.19.2.2 global_shut()

```
int global_shut ( )
```

shut down catalog and memory.

### 5.19.3 Variable Documentation

#### 5.19.3.1 g_catalog

```
Catalog g_catalog  [extern]
```

#### 5.19.3.2 g_memory

```
Memory g_memory  [extern]
```

## 5.20 global.h

Go to the documentation of this file.

```
1 #include "memory.h"
2 #include "catalog.h"
3
4 #define GLOBAL_MEMORY_SIZE    (1L«31)
5 #define GLOBAL_MEMORY_MINIMUM (1L« 3)
6
7 #define BNODE_POINTERS_NUM    (16)      //  shoule be 2*m, it's a default value and strongly advised
8
9 extern Memory g_memory;
10 extern Catalog g_catalog;
11
15 int global_init();
16
20 int global_shut();
```

## 5.21 system/hashindex.cc File Reference

```
#include "hashindex.h"
```

### 5.21.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.21.2 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

@basic 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.

## 5.22 system/hashindex.d File Reference

## 5.23 system/hashindex.h File Reference

```
#include "schema.h"
#include "hashtable.h"
```

## Classes

- class HashIndex
- struct HashInfo

## Macros

- #define HASHINFO_CAPICITY (8)

### 5.23.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.23.2 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

@basic 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.

### 5.23.3 Macro Definition Documentation

#### 5.23.3.1 HASHINFO_CAPICITY

```
#define HASHINFO_CAPICITY (8)
```

## 5.24 hashindex.h

Go to the documentation of this file.

```
1
22 #ifndef _HASHINDEX_H
23 #define _HASHINDEX_H
24
25 #include "schema.h"
26 #include "hashtable.h"
27
28 // support INT and CHARN, id type data
29 #define HASHINFO_CAPICITY (8)
30
32 struct HashInfo {
33     char *result[HASHINFO_CAPICITY];
34     int     rnum;
35     int     ppos;
36     int     last;
37     int64_t hash;
38 };
39
41 class HashIndex:public Index {
42   private:
43     HashTable *ih_hashtable;
44     int64_t ih_cell_capbits;
45     int64_t *ih_hash_bits;
46     BasicType **ih_datatype;
47     int64_t ih_column_num;
48     int64_t ih_column_cap;
49     int64_t *ih_table_offset;
51   public:
58     HashIndex(int64_t h_id, const char *i_name, Key & i_key)
59         :Index(h_id,i_name,HASHINDEX,i_key)
60     {
61     }
62     // init process bool init(void);
67     bool init (void);
72     void setCellCap(int64_t cell_capbits) {  // cell szie is power of cell_capbits by 2
73         this->ih_cell_capbits = cell_capbits;
74     }
80     bool addIndexDTpye(BasicType * i_dt,int64_t offset);
86     bool finish(void);
91     bool shut(void);
92
93     //  data operator
101     bool insert(void *i_data, void *p_in);
109     bool insert(void *i_data[], void *p_in);
118     bool set_ls(void *i_data1, void *i_data2, void *info);
127     bool set_ls(void *i_data1[], void *i_data2[], void *info);
136     bool lookup(void *i_data, void *info, void *&result);
145     bool lookup(void *i_data[], void *info, void *&result);
152     bool del(void *i_data);
159     bool del(void *i_data[]);
163     void print(void);
164
165   private:
171     int64_t tranToInt64(void *i_data);
177     int64_t tranToInt64(void *i_data[]);
184     int64_t hash(char *str, int64_t maxlen);
192     bool cmpEQ(void *i_data[], void *result);
200     bool cmpEQ(void *i_data, void *result);
201 };
202 #endif
```

## 5.25 system/hashtable.cc File Reference

```
#include "hashtable.h"
```

### Macros

- #define ESTIMATE_ERROR (1024)

### 5.25.1 Macro Definition Documentation

#### 5.25.1.1 ESTIMATE_ERROR

```
#define ESTIMATE_ERROR (1024)
```

## 5.26 system/hashtable.d File Reference

## 5.27 system/hashtable.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <unordered_map>
#include "mymemory.h"
```

### 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

### 5.27.1 Macro Definition Documentation

#### 5.27.1.1 hc_capacity

```
#define hc_capacity hc_union.num_2_or_more.capacity
```

#### 5.27.1.2 hc_ent

```
#define hc_ent hc_union.ent
```

#### 5.27.1.3 hc_ents

```
#define hc_ents hc_union.num_2_or_more.ents
```

## 5.28 hashtable.h

Go to the documentation of this file.

```
1
15 #ifndef _HASH_TABLE_H
16 #define _HASH_TABLE_H
17 /* ---------------------------------------------------------------------- */
18
19 #include <stdio.h>
20 #include <stdint.h>
21 #include <stdlib.h>
22 #include <string.h>
23 #include <unordered_map>
24 #include "mymemory.h"
25
26 /* ---------------------------------------------------------------------- */
28 class Hashcode_Ptr {
29   public:
30     int64_t hash_code;
31     char *tuple;
32 };  // class Hashcode_Ptr
33
35 class HashCell {
36   public:
37     int hc_num;
38     union {
39         Hashcode_Ptr ent;
40         struct {                // not hit,seach a array,not a link-list,decrease cache miss
41             int capacity;
42             Hashcode_Ptr *ents;
43         } num_2_or_more;
44     } hc_union;
45 };  // class HashCell
46
47 #define hc_ent          hc_union.ent
48 #define hc_capacity     hc_union.num_2_or_more.capacity
49 #define hc_ents         hc_union.num_2_or_more.ents
50
52 class HashTable {
53   public:
54     int estimated_num_distinct_keys;
55     double estimated_duplicates_per_key;
56     int initial_array_size;
57     char *begin;
58     Hashcode_Ptr *free_header[16];
59                                         // allocate the array
60                                         // free_header[0]: all of initial_array_size
61                                         // free_header[1]: 2*initial_array_size
62                                         // ...
63                                         // free_header[k]: 2^k * initial_array_size
64     Hashcode_Ptr *avail;
65     Hashcode_Ptr *end;
66     HashCell *table;
67     int table_size;
68     int more_allocated;
69   private:
70     std::unordered_map<void*, int> pointer2size;
71     void *allocate(int size);
72     void  free(void *mem);
73     int size_to_slot(int array_size);
75   public:
82     HashTable(int estimatedNumDistinctKeys, double estimatedDupPerKey,
83               int num_partitions);
87     ~HashTable();
```

```
88
96      bool add(int64_t hashCode, char *tup);
104      bool del(int64_t hashCode, char *tup);
105
106      // return num matched
107      // <0: means capacity has been reached, there could be more
108      // -ret is the last position probed
117      int probe(int64_t hashCode, char *match[], int capacity);
127      int probe_contd(int64_t hashCode, int last, char *match[],
128                      int capacity);
132      void utilization();
136      void show();
137
138 };                              //HashTable
139
140 /* ---------------------------------------------------------------------------- */
141 #endif                          /* _HASH_TABLE_H */
```

## 5.29 system/mymemory.cc File Reference

```
#include "mymemory.h"
```

### Variables

- Memory g_memory

### 5.29.1 Detailed Description

@autrhor liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.29.2 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

### 5.29.3 Variable Documentation

#### 5.29.3.1 g_memory

```
Memory g_memory
```

## 5.30 system/mymemory.d File Reference

## 5.31 system/mymemory.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <vector>
#include <sys/mman.h>
```

### Classes

- class Memory

### Macros

- #define MEMORY_OK 0
- #define NON_TABLE_MEMORY_ADDR ((void∗)0x600000000000)
- #define TABLE_MEMORY_ALLOC_INC (1L<<28)
- #define TABLE_MEMORY_ALLOC_MAX (1L<<34)
- #define TABLE_MEMORY_INIT_ADDR ((void∗)0x700000000000)
- #define TABLE_MEMORY_MAX_ADDR ((void∗)0x7f0000000000)

### Variables

- Memory g_memory

### 5.31.1 Detailed Description

@autrhor liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.31.2 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

### 5.31.3 Macro Definition Documentation

#### 5.31.3.1 MEMORY_OK

```
#define MEMORY_OK 0
```

#### 5.31.3.2 NON_TABLE_MEMORY_ADDR

```
#define NON_TABLE_MEMORY_ADDR ((void*)0x600000000000)
```

#### 5.31.3.3 TABLE_MEMORY_ALLOC_INC

```
#define TABLE_MEMORY_ALLOC_INC (1L<<28)
```

#### 5.31.3.4 TABLE_MEMORY_ALLOC_MAX

```
#define TABLE_MEMORY_ALLOC_MAX (1L<<34)
```

#### 5.31.3.5 TABLE_MEMORY_INIT_ADDR

```
#define TABLE_MEMORY_INIT_ADDR ((void*)0x700000000000)
```

#### 5.31.3.6 TABLE_MEMORY_MAX_ADDR

```
#define TABLE_MEMORY_MAX_ADDR ((void*)0x7f0000000000)
```

### 5.31.4 Variable Documentation

#### 5.31.4.1 g_memory

```
Memory g_memory  [extern]
```

## 5.32 mymemory.h

Go to the documentation of this file.

```
1
16 #ifndef _MYMEMORY_H
17 #define _MYMEMORY_H
18
19 #include <stdint.h>
20 #include <stdio.h>
21 #include <vector>
22 #include <sys/mman.h>
23
24 #define MEMORY_OK 0
25
26 #define NON_TABLE_MEMORY_ADDR  ((void*)0x600000000000)
27 #define TABLE_MEMORY_INIT_ADDR ((void*)0x700000000000)
28 #define TABLE_MEMORY_MAX_ADDR  ((void*)0x7f0000000000)
29 #define TABLE_MEMORY_ALLOC_MAX (1L«34)  //  maximum   16  GB per table
30 #define TABLE_MEMORY_ALLOC_INC (1L«28)  //  fixed     256 MB per increment
31
32 class Memory {
33   private:
34     char *m_head;
35     char *m_curr;
36     char *m_tail;
37     char **m_array_list;
38     int64_t m_total;
39     int64_t m_mins;
40     char *m_table_addr;
42   public:
50     int init(int64_t total, int64_t mins);
58     int64_t alloc(char *&p, int64_t size);
66     int64_t free(char *p, int64_t size);
70     int shut(void);
74     int print(void);
80     int allocTableAddr (char *&p);
81   private:
85     unsigned int slot(int64_t size);
92     int64_t alloc_default(char *&p, int64_t size);
93 };  // class Memory
94
95 extern Memory g_memory;
96 #endif
```

## 5.33 system/nodepref.h File Reference

```
#include "gcc_pf_p3.h"
```

### Macros

- #define AREA_LINE_NUM ((NODE_LINE_NUM+1)/2)
- #define BNODE_SIZE (CACHE_LINE_SIZE * NODE_LINE_NUM)
- #define CACHE_LINE_SIZE L3_CACHE_LINE
- #define ITEM_SIZE ( 8)
- #define L3_CACHE_LINE (64)
- #define LEAF_PREF NODE_PREF
- #define LEAF_PREF_ST NODE_PREF_ST
- #define NODE_LINE_NUM ( 4)

### 5.33.1 Macro Definition Documentation

#### 5.33.1.1 AREA_LINE_NUM

```
#define AREA_LINE_NUM ((NODE_LINE_NUM+1)/2)
```

#### 5.33.1.2 BNODE_SIZE

```
#define BNODE_SIZE (CACHE_LINE_SIZE * NODE_LINE_NUM)
```

#### 5.33.1.3 CACHE_LINE_SIZE

```
#define CACHE_LINE_SIZE L3_CACHE_LINE
```

#### 5.33.1.4 ITEM_SIZE

```
#define ITEM_SIZE ( 8)
```

–X86_64 prefetch file–

#### 5.33.1.5 L3_CACHE_LINE

```
#define L3_CACHE_LINE (64)
```

#### 5.33.1.6 LEAF_PREF

```
#define LEAF_PREF NODE_PREF
```

#### 5.33.1.7 LEAF_PREF_ST

```
#define LEAF_PREF_ST NODE_PREF_ST
```

#### 5.33.1.8 NODE_LINE_NUM

```
#define NODE_LINE_NUM ( 4)
```

## 5.34 nodepref.h

Go to the documentation of this file.

```
1
15 #ifndef _PBTREE_NODEPREF_H
16 #define _PBTREE_NODEPREF_H
17 /* ------------------------------------------------------------------------ */
19 #include "gcc_pf_p3.h"
20
21 #define ITEM_SIZE     ( 8)
22 #define NODE_LINE_NUM ( 4)
23 #define L3_CACHE_LINE (64)
24 // node size
25 #define CACHE_LINE_SIZE         L3_CACHE_LINE
26 #define BNODE_SIZE              (CACHE_LINE_SIZE * NODE_LINE_NUM)
27 #define AREA_LINE_NUM   ((NODE_LINE_NUM+1)/2)
28
29 static void inline NODE_PREF(register void *bbp)
30 {
31      pfld (* ((char *)bbp));
32 #    if NODE_LINE_NUM >= 2
33      pfld (* ((char *)bbp + CACHE_LINE_SIZE));
34 #    endif
35 #    if NODE_LINE_NUM >= 3
36      pfld (* ((char *)bbp + CACHE_LINE_SIZE*2));
37 #    endif
38 #    if NODE_LINE_NUM >= 4
39      pfld (* ((char *)bbp + CACHE_LINE_SIZE*3));
40 #    endif
41 #    if NODE_LINE_NUM >= 5
42      pfld (* ((char *)bbp + CACHE_LINE_SIZE*4));
43 #    endif
44 #    if NODE_LINE_NUM >= 6
45      pfld (* ((char *)bbp + CACHE_LINE_SIZE*5));
46 #    endif
47 #    if NODE_LINE_NUM >= 7
48      pfld (* ((char *)bbp + CACHE_LINE_SIZE*6));
49 #    endif
50 #    if NODE_LINE_NUM >= 8
51      pfld (* ((char *)bbp + CACHE_LINE_SIZE*7));
52 #    endif
53 #    if NODE_LINE_NUM >= 9
54      pfld (* ((char *)bbp + CACHE_LINE_SIZE*8));
55 #    endif
56 #    if NODE_LINE_NUM >= 10
57      pfld (* ((char *)bbp + CACHE_LINE_SIZE*9));
58 #    endif
59 #    if NODE_LINE_NUM >= 11
60      pfld (* ((char *)bbp + CACHE_LINE_SIZE*10));
61 #    endif
62 #    if NODE_LINE_NUM >= 12
63      pfld (* ((char *)bbp + CACHE_LINE_SIZE*11));
64 #    endif
65 #    if NODE_LINE_NUM >= 13
66      pfld (* ((char *)bbp + CACHE_LINE_SIZE*12));
67 #    endif
68 #    if NODE_LINE_NUM >= 14
69      pfld (* ((char *)bbp + CACHE_LINE_SIZE*13));
70 #    endif
71 #    if NODE_LINE_NUM >= 15
72      pfld (* ((char *)bbp + CACHE_LINE_SIZE*14));
73 #    endif
74 #    if NODE_LINE_NUM >= 16
75      pfld (* ((char *)bbp + CACHE_LINE_SIZE*15));
76 #    endif
77 #    if NODE_LINE_NUM >= 17
78      pfld (* ((char *)bbp + CACHE_LINE_SIZE*16));
79 #    endif
80 #    if NODE_LINE_NUM >= 18
81      pfld (* ((char *)bbp + CACHE_LINE_SIZE*17));
82 #    endif
83 #    if NODE_LINE_NUM >= 19
84      pfld (* ((char *)bbp + CACHE_LINE_SIZE*18));
85 #    endif
86 #    if NODE_LINE_NUM >= 20
87      pfld (* ((char *)bbp + CACHE_LINE_SIZE*19));
88 #    endif
89 #    if NODE_LINE_NUM >= 21
90      pfld (* ((char *)bbp + CACHE_LINE_SIZE*20));
91 #    endif
92 #    if NODE_LINE_NUM >= 22
93      pfld (* ((char *)bbp + CACHE_LINE_SIZE*21));
94 #    endif
95 #    if NODE_LINE_NUM >= 23
96      pfld (* ((char *)bbp + CACHE_LINE_SIZE*22));
```

```
97 #    endif
98 #    if NODE_LINE_NUM >= 24
99      pfld (* ((char *)bbp + CACHE_LINE_SIZE*23));
100 #    endif
101 #    if NODE_LINE_NUM >= 25
102      pfld (* ((char *)bbp + CACHE_LINE_SIZE*24));
103 #    endif
104 #    if NODE_LINE_NUM >= 26
105      pfld (* ((char *)bbp + CACHE_LINE_SIZE*25));
106 #    endif
107 #    if NODE_LINE_NUM >= 27
108      pfld (* ((char *)bbp + CACHE_LINE_SIZE*26));
109 #    endif
110 #    if NODE_LINE_NUM >= 28
111      pfld (* ((char *)bbp + CACHE_LINE_SIZE*27));
112 #    endif
113 #    if NODE_LINE_NUM >= 29
114      pfld (* ((char *)bbp + CACHE_LINE_SIZE*28));
115 #    endif
116 #    if NODE_LINE_NUM >= 30
117      pfld (* ((char *)bbp + CACHE_LINE_SIZE*29));
118 #    endif
119 #    if NODE_LINE_NUM >= 31
120      pfld (* ((char *)bbp + CACHE_LINE_SIZE*30));
121 #    endif
122 #    if NODE_LINE_NUM >= 32
123      pfld (* ((char *)bbp + CACHE_LINE_SIZE*31));
124 #    endif
125 #    if NODE_LINE_NUM >= 33
126 #    error "NODE_LINE_NUM must be <= 32!"
127 #    endif
128 }
129
130 static void inline NODE_PREF_ST(register void *bbp)
131 {
132      pfst (* ((char *)bbp));
133 #    if NODE_LINE_NUM >= 2
134      pfst (* ((char *)bbp + CACHE_LINE_SIZE));
135 #    endif
136 #    if NODE_LINE_NUM >= 3
137      pfst (* ((char *)bbp + CACHE_LINE_SIZE*2));
138 #    endif
139 #    if NODE_LINE_NUM >= 4
140      pfst (* ((char *)bbp + CACHE_LINE_SIZE*3));
141 #    endif
142 #    if NODE_LINE_NUM >= 5
143      pfst (* ((char *)bbp + CACHE_LINE_SIZE*4));
144 #    endif
145 #    if NODE_LINE_NUM >= 6
146      pfst (* ((char *)bbp + CACHE_LINE_SIZE*5));
147 #    endif
148 #    if NODE_LINE_NUM >= 7
149      pfst (* ((char *)bbp + CACHE_LINE_SIZE*6));
150 #    endif
151 #    if NODE_LINE_NUM >= 8
152      pfst (* ((char *)bbp + CACHE_LINE_SIZE*7));
153 #    endif
154 #    if NODE_LINE_NUM >= 9
155      pfst (* ((char *)bbp + CACHE_LINE_SIZE*8));
156 #    endif
157 #    if NODE_LINE_NUM >= 10
158      pfst (* ((char *)bbp + CACHE_LINE_SIZE*9));
159 #    endif
160 #    if NODE_LINE_NUM >= 11
161      pfst (* ((char *)bbp + CACHE_LINE_SIZE*10));
162 #    endif
163 #    if NODE_LINE_NUM >= 12
164      pfst (* ((char *)bbp + CACHE_LINE_SIZE*11));
165 #    endif
166 #    if NODE_LINE_NUM >= 13
167      pfst (* ((char *)bbp + CACHE_LINE_SIZE*12));
168 #    endif
169 #    if NODE_LINE_NUM >= 14
170      pfst (* ((char *)bbp + CACHE_LINE_SIZE*13));
171 #    endif
172 #    if NODE_LINE_NUM >= 15
173      pfst (* ((char *)bbp + CACHE_LINE_SIZE*14));
174 #    endif
175 #    if NODE_LINE_NUM >= 16
176      pfst (* ((char *)bbp + CACHE_LINE_SIZE*15));
177 #    endif
178 #    if NODE_LINE_NUM >= 17
179      pfst (* ((char *)bbp + CACHE_LINE_SIZE*16));
180 #    endif
181 #    if NODE_LINE_NUM >= 18
182      pfst (* ((char *)bbp + CACHE_LINE_SIZE*17));
183 #    endif
```

```
184 #     if NODE_LINE_NUM >= 19
185       pfst (* ((char *)bbp + CACHE_LINE_SIZE*18));
186 #     endif
187 #     if NODE_LINE_NUM >= 20
188       pfst (* ((char *)bbp + CACHE_LINE_SIZE*19));
189 #     endif
190 #     if NODE_LINE_NUM >= 21
191       pfst (* ((char *)bbp + CACHE_LINE_SIZE*20));
192 #     endif
193 #     if NODE_LINE_NUM >= 22
194       pfst (* ((char *)bbp + CACHE_LINE_SIZE*21));
195 #     endif
196 #     if NODE_LINE_NUM >= 23
197       pfst (* ((char *)bbp + CACHE_LINE_SIZE*22));
198 #     endif
199 #     if NODE_LINE_NUM >= 24
200       pfst (* ((char *)bbp + CACHE_LINE_SIZE*23));
201 #     endif
202 #     if NODE_LINE_NUM >= 25
203       pfst (* ((char *)bbp + CACHE_LINE_SIZE*24));
204 #     endif
205 #     if NODE_LINE_NUM >= 26
206       pfst (* ((char *)bbp + CACHE_LINE_SIZE*25));
207 #     endif
208 #     if NODE_LINE_NUM >= 27
209       pfst (* ((char *)bbp + CACHE_LINE_SIZE*26));
210 #     endif
211 #     if NODE_LINE_NUM >= 28
212       pfst (* ((char *)bbp + CACHE_LINE_SIZE*27));
213 #     endif
214 #     if NODE_LINE_NUM >= 29
215       pfst (* ((char *)bbp + CACHE_LINE_SIZE*28));
216 #     endif
217 #     if NODE_LINE_NUM >= 30
218       pfst (* ((char *)bbp + CACHE_LINE_SIZE*29));
219 #     endif
220 #     if NODE_LINE_NUM >= 31
221       pfst (* ((char *)bbp + CACHE_LINE_SIZE*30));
222 #     endif
223 #     if NODE_LINE_NUM >= 32
224       pfst (* ((char *)bbp + CACHE_LINE_SIZE*31));
225 #     endif
226 #     if NODE_LINE_NUM >= 33
227 #     error "NODE_LINE_NUM must be <= 32!"
228 #     endif
229 }
230
231 #define LEAF_PREF           NODE_PREF
232 #define LEAF_PREF_ST        NODE_PREF_ST
233
234 // prefetch AREA_LINE_NUM cache lines starting from &(area[i])
235 static void inline AREA_PREF(void * area[], int i, int total)
236 {
237    if ((i) + AREA_LINE_NUM*CACHE_LINE_SIZE/ITEM_SIZE <= (total)) {
238       register char *p = (char *)(&(area[i]));
239       pfst (* p);
240 #     if AREA_LINE_NUM >= 2
241       pfst (* (p + CACHE_LINE_SIZE));
242 #     endif
243 #     if AREA_LINE_NUM >= 3
244       pfst (* (p + CACHE_LINE_SIZE*2));
245 #     endif
246 #     if AREA_LINE_NUM >= 4
247       pfst (* (p + CACHE_LINE_SIZE*3));
248 #     endif
249 #     if AREA_LINE_NUM >= 5
250       pfst (* (p + CACHE_LINE_SIZE*4));
251 #     endif
252 #     if AREA_LINE_NUM >= 6
253       pfst (* (p + CACHE_LINE_SIZE*5));
254 #     endif
255 #     if AREA_LINE_NUM >= 7
256       pfst (* (p + CACHE_LINE_SIZE*6));
257 #     endif
258 #     if AREA_LINE_NUM >= 8
259       pfst (* (p + CACHE_LINE_SIZE*7));
260 #     endif
261 #     if AREA_LINE_NUM >= 9
262       pfst (* (p + CACHE_LINE_SIZE*8));
263 #     endif
264 #     if AREA_LINE_NUM >= 10
265       pfst (* (p + CACHE_LINE_SIZE*9));
266 #     endif
267 #     if AREA_LINE_NUM >= 11
268       pfst (* (p + CACHE_LINE_SIZE*10));
269 #     endif
270 #     if AREA_LINE_NUM >= 12
```

```
271       pfst (* (p + CACHE_LINE_SIZE*11));
272 #     endif
273 #     if AREA_LINE_NUM >= 13
274       pfst (* (p + CACHE_LINE_SIZE*12));
275 #     endif
276 #     if AREA_LINE_NUM >= 14
277       pfst (* (p + CACHE_LINE_SIZE*13));
278 #     endif
279 #     if AREA_LINE_NUM >= 15
280       pfst (* (p + CACHE_LINE_SIZE*14));
281 #     endif
282 #     if AREA_LINE_NUM >= 16
283       pfst (* (p + CACHE_LINE_SIZE*15));
284 #     endif
285 #     if AREA_LINE_NUM >= 17
286 #     error "AREA_LINE_NUM must be <= 16!"
287 #     endif
288     }
289 }
290
291 /* ---------------------------------------------------------------------- */
292 #endif /* _PBTREE_NODEPREF_H */
```

## 5.35 system/pbtree.cc File Reference

```
#include "pbtree.h"
```

### Macros

- #define LEFT_KEY_NUM ((LEAF_KEY_NUM+1)/2)
- #define LEFT_KEY_NUM ((NON_LEAF_KEY_NUM)/2)
- #define RIGHT_KEY_NUM ((LEAF_KEY_NUM+1) - LEFT_KEY_NUM)
- #define RIGHT_KEY_NUM ((NON_LEAF_KEY_NUM) - LEFT_KEY_NUM)

### 5.35.1 Detailed Description

**Author**

Shimin Chen shimin.chen@gmail.com

**Version**

1.0

### 5.35.2 LICENSE

TBD

### 5.35.3 DESCRIPTION

The pbtree class implements prefetching B+-tree (without jump pointer arrays). with NO_PREFETCH, this becomes an B+-tree implementation.

### 5.35.4 Macro Definition Documentation

#### 5.35.4.1 LEFT_KEY_NUM [1/2]

```
#define LEFT_KEY_NUM ((LEAF_KEY_NUM+1)/2)
```

#### 5.35.4.2 LEFT_KEY_NUM [2/2]

```
#define LEFT_KEY_NUM ((NON_LEAF_KEY_NUM)/2)
```

#### 5.35.4.3 RIGHT_KEY_NUM [1/2]

```
#define RIGHT_KEY_NUM ((LEAF_KEY_NUM+1) - LEFT_KEY_NUM)
```

#### 5.35.4.4 RIGHT_KEY_NUM [2/2]

```
#define RIGHT_KEY_NUM ((NON_LEAF_KEY_NUM) - LEFT_KEY_NUM)
```

## 5.36 system/pbtree.d File Reference

## 5.37 system/pbtree.h File Reference

```
#include <assert.h>
#include <stdlib.h>
#include "mymemory.h"
#include "nodepref.h"
#include <stdio.h>
```

### Classes

- class bnode
- class pbtree
- class Pbtree
- class Pointer8B

## Macros

- #define BKEY_NUM (BNODE_SIZE/(KEY_SIZE+POINTER8B_SIZE) - 1)
- #define bleaf bnode
- #define bnext(ptr) (((bleaf *)(ptr))->ch(0))
- #define bnum(ptr) (((bleaf *)(ptr))->k(0))
- #define KEY_SIZE 8
- #define LEAF_KEY_NUM BKEY_NUM
- #define MAX_KEY ((key_type)(0x7fffffffffffffff))
- #define MIN_KEY ((key_type)(0x8000000000000000))
- #define NON_LEAF_KEY_NUM BKEY_NUM
- #define POINTER8B_SIZE 8
- #define POINTER_SIZE 8

## Typedefs

- typedef long long key_type

### 5.37.1 Detailed Description

**Author**

Shimin Chen shimin.chen@gmail.com

**Version**

1.0

### 5.37.2 LICENSE

TBD

### 5.37.3 DESCRIPTION

The pbtree class implements prefetching B+-tree (without jump pointer arrays). with NO_PREFETCH, this becomes an B+-tree implementation.

### 5.37.4 Macro Definition Documentation

#### 5.37.4.1 BKEY_NUM

```
#define BKEY_NUM (BNODE_SIZE/(KEY_SIZE+POINTER8B_SIZE) - 1)
```

#### 5.37.4.2 bleaf

```
#define bleaf bnode
```

#### 5.37.4.3 bnext

```
#define bnext(
             ptr ) (((bleaf *)(ptr))->ch(0))
```

#### 5.37.4.4 bnum

```
#define bnum(
             ptr ) (((bleaf *)(ptr))->k(0))
```

#### 5.37.4.5 KEY_SIZE

```
#define KEY_SIZE 8
```

#### 5.37.4.6 LEAF_KEY_NUM

```
#define LEAF_KEY_NUM BKEY_NUM
```

#### 5.37.4.7 MAX_KEY

```
#define MAX_KEY ((key_type)(0x7fffffffffffffff))
```

#### 5.37.4.8 MIN_KEY

```
#define MIN_KEY ((key_type)(0x8000000000000000))
```

#### 5.37.4.9 NON_LEAF_KEY_NUM

```
#define NON_LEAF_KEY_NUM BKEY_NUM
```

#### 5.37.4.10 POINTER8B_SIZE

```
#define POINTER8B_SIZE 8
```

#### 5.37.4.11 POINTER_SIZE

```
#define POINTER_SIZE 8
```

### 5.37.5 Typedef Documentation

#### 5.37.5.1 key_type

```
typedef long long key_type
```

## 5.38 pbtree.h

Go to the documentation of this file.

```
1
16 /* B+ tree:
17
18    non_leaf nodes:
19     - structure:
20        k[0] k[1] k[2] ... k[N-1]
21         ch[0] ch[1] ch[2] ... ch[N-1]
22
23     - property
24        k[0] is the actual number of keys in the node
25        k[1] < k[2] < ... < k[N-1]
26        ch[] are pointers to subtrees.
27
28        can hold at most (N-1) keys and N pointers.
29
30     - implementation specific:
31       assert(k[0]>=1)
32
33    leaf nodes:
34     - structure:
35         k[0]    k[1]  k[2] ... k[N-1]
36         ch[0]   ch[1] ch[2]... ch[N-1]
37
38     - property:
39         (k[i], ch[i]) 1<=i<=N-1, are index entries
40        k[0] is the actual number of entries in the node.
41        ch[0] is leaf sibling pointer.
42
43        can hold at most (N-1) index entries.
44
45     - implementation specific:
46       assert(k[0]>=1)
```

```
47  */
48
49 #ifndef _PBTREE_PBTREE_H
50 #define _PBTREE_PBTREE_H
51
52 #include <assert.h>
53 #include <stdlib.h>
54 #include "mymemory.h"
55
56 #include "nodepref.h"
57 /* ---------------------------------------------------------------------- */
58 // #include "tree.h" based on x86_64 prefetch
59 //---------from tree.h-----------------------------------------------------
60 #include <stdio.h>
61
62    typedef long long key_type;
63    #define KEY_SIZE             8
64    #define POINTER_SIZE         8
65    #define POINTER8B_SIZE       8
66    #define MAX_KEY              ((key_type)(0x7fffffffffffffff))
67    #define MIN_KEY              ((key_type)(0x8000000000000000))
68
69 /* ---------------------------------------------------------------------- */
70
71 // BKEY_NUM = N-1
72 #define BKEY_NUM                (BNODE_SIZE/(KEY_SIZE+POINTER8B_SIZE) - 1)
73 #define NON_LEAF_KEY_NUM        BKEY_NUM
74 #define LEAF_KEY_NUM            BKEY_NUM
75
76 /* ---------------------------------------------------------------------- */
77 class Pointer8B {
78  public:
79    // unsigned int num;
80    unsigned long long value;
81
82  public:
83    Pointer8B & operator= (const void * ptr)
84    {
85        value= (unsigned long long)ptr;
86        return *this;
87    }
88
89    Pointer8B & operator= (const Pointer8B & p)
90    {
91        value= p.value;
92        return *this;
93    }
94
95    operator void*() { return (void *)value; }
96    operator char*() { return (char *)value; }
97    operator struct bnode *() { return (struct bnode *)value;}
98    operator struct bleaf *() { return (struct bleaf *)value;}
99    operator unsigned long long () { return value;}
100
101    void print(void) {
102        if (value & (1UL<<63)) {
103            void *p = (void*)(value & ((1UL<<63)-1));
104            int num = *(int*) p;
105            void** vptrs = (void**)((char*)p+2*sizeof(int));
106            printf ("[");
107            for (int ii=0; ii< num-1; ii++)
108                printf("%p,", vptrs[ii]);
109            printf ("%p]\n", vptrs[num-1]);
110        }
111        else {
112            printf("[%p]\n", (void*)value);
113        }
114    }
115
116 }; // Pointer8B
117
118 #ifdef UNIFIED_IDX_ENTRY
119 typedef struct IdxEntry{
120   key_type   k;
121   Pointer8B  ch;
122 } IdxEntry;
123
124 class bnode {
125  public:
126   IdxEntry   ent[BKEY_NUM + 1];
127  public:
128   key_type  & k(int idx)  { return ent[idx].k; }
129   Pointer8B & ch(int idx) { return ent[idx].ch; }
130   char * chEndAddr(int idx)
131   {return (char *)&(ent[idx].ch)+sizeof(Pointer8B)-1;}
132 }; // bnode
133 #else
```

```
134 class bnode {
135  public:
136   key_type   key[BKEY_NUM + 1];
137   Pointer8B  child[BKEY_NUM + 1];
138  public:
139   key_type  & k(int idx)  { return key[idx]; }
140   Pointer8B & ch(int idx) { return child[idx]; }
141   char * chEndAddr(int idx)
142   {return (char *)&(child[idx])+sizeof(Pointer8B)-1;}
143 }; // bnode
144 #endif
145
146 #define bleaf  bnode
147 #define bnum(ptr)       (((bleaf *)(ptr))->k(0))
148 #define bnext(ptr)      (((bleaf *)(ptr))->ch(0))
149
150 /* ---------------------------------------------------------------------- */
151 class pbtree {
152  private: // tree nodes definition
153
154    bnode * get_node ()
155    {
156     char *p = NULL;
157         int64_t sz = g_memory.alloc (p, BNODE_SIZE);
158     return sz == BNODE_SIZE? (bnode*)p: NULL;
159    }
160
161    void delete_node (void *p)
162    {
163         g_memory.free((char*)p, BNODE_SIZE);
164    }
165
166  public:  // root and level
167
168    bnode *  tree_root;
169    int      root_level;               // leaf: level 0, parent of leaf: level 1
170
171  private:
172    /* don't need to be prefetched!!! */
173    bnode * parray[32];
174    int     ppos[32];
175
176  public:
177
178    int init (void);
179    int shut (void);
180    // given a search key, perform the search operation
181    // return the leaf node pointer and the position with leaf node
182    void * lookup (key_type key, int *pos);
183
184    void * get_recptr (void *p, int pos)
185    {
186         if (p == NULL) return NULL;
187         return ((bleaf *)p)->ch(pos);
188    }
189
190    key_type get_key (void *p, int pos)
191    {
192     return ((bleaf *)p)->k(pos);
193    }
194
195    void** get_recptr_sp (void *p, int pos)
196    {
197         if (p == NULL) return NULL;
198         return (void**)&(((bleaf *)p)->ch(pos));
199    }
200
201    // insert (key, ptr)
202    void insert (key_type key, void *ptr);
203
204    // delete key
205    void del (key_type key);
206
207    // range scan
208    // Input:  *p is the starting leaf node
209    //         *spos is the starting position
210    //         endkey is the end key
211    //         *num is the buffer size of area[]
212    // Operation: store record pointers into area[]
213    // Output:
214    //    return 0: endkey is not reached
215    //              should call again with (*p, *spos)
216    //              *num is the number of pointers retrieved into area[]
217    //    return non-0: endkey is reached
218    //              *num is the number of pointers retrieved into area[]
219    int scan (void **p, int *spos, key_type startkey, key_type endkey,
220              void * area[], int *num);
```

```
221
222  private:
223    void print (bnode *p, int level);
224    void check (bnode *p, int level, key_type *start, key_type *end, void **ptr);
225
226  public:
227    void print ()
228    {
229         printf("Printing pbtree\n");
230             print (tree_root, root_level);
231    }
232
233    void check (key_type *start, key_type *end)
234    {
235         assert (sizeof(bnode)==BNODE_SIZE);
236         assert (sizeof(bleaf)==BNODE_SIZE);
237         void * ptr = NULL;
238         check (tree_root, root_level, start, end, &ptr);
239    }
240
241    int level () {return root_level;}
242
243  public:
244    void save (char *filename)
245    {
246       FILE *fp = fopen (filename, "w");
247       if (fp == NULL) {
248         perror (filename); exit (1);
249       }
250       fprintf (fp, "tree_root=%p\n", tree_root);
251       fprintf (fp, "root_level=%d\n\n", root_level);
252       fclose (fp);
253    }
254
255    void load (char *filename)
256    {
257       FILE *fp = fopen (filename, "r");
258       if (fp == NULL) {
259         perror (filename); exit (1);
260       }
261       if ((fscanf (fp, "tree_root=%p\n", &tree_root) != 1) ||
262           (fscanf (fp, "root_level=%d\n", &root_level) != 1)) {
263
264         fprintf (stderr, "%s format error!\n", filename);
265         exit (1);
266       }
267       fclose (fp);
268    }
269
270 }; // pbtree
271
272 /* ---------------------------------------------------------------------- */
273 class Pbtree {   // support duplicated elements
274   private:
275     pbtree p_pbtree;
276     char*  p_free_header[16]; // minimum size is 32 Byte
277
278   public:
279     void  init   (void);
280     bool  insert (key_type key, void* ptr);
281     bool  del    (key_type key, void* ptr);
282     void* lookup (key_type);
283     int   get_recptr (void* p, void* match[], int capicity, int &pos);
284
285     void* lookup_s (key_type key, int *pos);
286     int   scan (void **p, int *spos, key_type startkey, key_type endkey,
287                 void * area[], int *num);
288
289     void  shut   (void);
290     void  print  (void);
291
292   public:
293     bool  allocate (char *&p,int leve) {
294         if (p_free_header[leve]) {
295             p = p_free_header[leve];
296             p_free_header[leve] = *(char**)p_free_header[leve];
297             *(int*)(p) = 0;
298             *(int*)(p+sizeof(int)) = leve2cap (leve);
299             return true;
300         }
301         else {
302             int64_t size = leve2size (leve);
303             int64_t sz = g_memory.alloc (p, size);
304             if (sz != size) {
305                 printf ("[Pbtree][ERROR][allocate]: g_memory is not enough! 296\n");
306                 return false;
307             }
```

```
308             *(int*)(p) = 0;
309             *(int*)(p+sizeof(int)) = leve2cap (leve);
310             return true;
311         }
312         return false;
313     }
314     bool  free (char *p, int leve) {
315         *(char**)(p) = p_free_header[leve];
316         p_free_header[leve] = p;
317         return true;
318     }
319     int cap2leve(int cap) {
320         int t = ((cap+1)»2);
321         int leve = 0;
322         while (t>1) {
323             leve ++;
324             t = t»1;
325         }
326         return leve;
327     }
328     int leve2cap(int leve) {
329         return (1«(leve+2))-1;
330     }
331     int leve2size (int leve) {
332         return 1«(leve+5);
333     }
334     int size2leve (int size) {
335         int leve = 0;
336         int sz = size;
337         while (sz > (1«5)) {
338             leve ++;
339             sz = sz » 1;
340         }
341         return leve;
342     }
343 };
344
345 /* ---------------------------------------------------------------------- */
346 #endif /* _PBTREE_H */
```

## 5.39 system/pbtreeindex.cc File Reference

```
#include "pbtreeindex.h"
```

## 5.40 system/pbtreeindex.d File Reference

## 5.41 system/pbtreeindex.h File Reference

```
#include "schema.h"
#include "pbtree.h"
```

### Classes

- class PbtreeIndex
- struct PbtreeInfo

### Macros

- #define PBTREEINFO_CAPICITY (16)

### 5.41.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.41.2 DESCRIPTION

pbtree index, here we support ID data type(INT8/16/32/64/DATE/TIME/DATETIME) 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

@basic 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 BptreeInfo with proper value, the first param is valid, leave the second to be NULL, BptreeInfo can help you iterately get values you need. (2) call lookup to get the value iterately. (3) scan you can scan to iterately get the value you need

### 5.41.3 Macro Definition Documentation

#### 5.41.3.1 PBTREEINFO_CAPICITY

```
#define PBTREEINFO_CAPICITY (16)
```

## 5.42 pbtreeindex.h

Go to the documentation of this file.

```
1
23 #ifndef _PBTREEINDEX_H
24 #define _PBTREEINDEX_H
25
26 #include "schema.h"
27 #include "pbtree.h"
28
29 // support INT, id type data
30 #define PBTREEINFO_CAPICITY (16)
31
33 struct PbtreeInfo {
34     key_type left;
35     key_type right;
36     void *l_ptr;
37     void *s_ptr;
38     int   s_pos;
39     int   s_num;
40     int   s_end;
41     void *area[PBTREEINFO_CAPICITY];
42     void *result[PBTREEINFO_CAPICITY];
43     int   cr_area;
44     int   cr_resu;
```

```
45     int   pos_resu;
46     int   le_resu;
47 };
48
49 class PbtreeIndex : public Index {
50   private:
51     Pbtree     pi_pbtree;
52     BasicType *pi_datatype;
54   public:
61     PbtreeIndex (int64_t pi_id, const char *i_name, Key &i_key)
62         :Index (pi_id, i_name, BPTREEINDEX, i_key)
63     {
64     }
69     bool init (void);
75     bool setIndexDTpye(BasicType * i_dt);
80     bool shut(void);
81
82     //  data operator
90     bool insert(void *i_data, void *p_in);
99     bool set_ls(void *i_data1, void *i_data2, void *info);
108     bool lookup(void *i_data, void *info, void *&result);
116     bool scan (void *info, void *&result);
124     bool del(void *i_data, void *p_del);
128     void print(void);
129
130 };
131
132 #endif
```

## 5.43 system/rowtable.cc File Reference

```
#include "rowtable.h"
```

### 5.43.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.43.2 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

## 5.44 system/rowtable.d File Reference

## 5.45 system/rowtable.h File Reference

```
#include "mymemory.h"
#include "schema.h"
```

## Classes

- class MStorage
- class RowTable
- class RPattern

## Variables

- Memory g_memory

### 5.45.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.45.2 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

### 5.45.3 Variable Documentation

#### 5.45.3.1 g_memory

```
Memory g_memory  [extern]
```

# 5.46 rowtable.h

Go to the documentation of this file.

```
1
19 #ifndef _ROWTABLE_H
20 #define _ROWTABLE_H
21
22 #include "mymemory.h"
23 #include "schema.h"
24
25 extern Memory g_memory;
26
28 class RPattern {
29   private:
30     int64_t rp_colnum;
31     int64_t *rp_offset;
32     BasicType **rp_dtype;
33     char *rp_memory;
34     int64_t rp_mem_sz;
35     int64_t rp_current;
36     int64_t rp_row_sz;
37     char par[128 - 3 * sizeof(void *) - 4 * sizeof(int64_t)];
39   public:
46     bool init(int64_t col_num) {
47         rp_row_sz = 0L;
48         rp_current = 0L;
49         rp_colnum = col_num;
50         int64_t alloc_size =
51             rp_colnum * (sizeof(BasicType *) + sizeof(int64_t));
52         for (rp_mem_sz = 8L; rp_mem_sz < alloc_size;
53              rp_mem_sz = rp_mem_sz << 1);
54          alloc_size = g_memory.alloc(rp_memory, rp_mem_sz);
55         if (alloc_size != rp_mem_sz) {
56             printf("[RPattern][ERROR][init]: alloc memory error! -1\n");
57             return false;
58         }
59         rp_offset = (int64_t *) rp_memory;
60         rp_dtype = (BasicType **) (rp_memory + rp_mem_sz / 2);
61         return true;
62     }
67     bool addColumn(BasicType * col_type) {
68         if (rp_current >= rp_colnum)
69             return false;
70         rp_dtype[rp_current] = col_type;
71         rp_offset[rp_current] = rp_row_sz;
72         rp_row_sz += col_type->getTypeSize();
73         rp_current++;
74         return true;
75     }
82     int64_t getColumnOffset(int64_t col_rank) {
83         return col_rank < rp_colnum ? rp_offset[col_rank] : -1;
84     }
91     BasicType *getColumnType(int64_t col_rank) {
92         return col_rank < rp_colnum ? rp_dtype[col_rank] : NULL;
93     }
97     void reset(void) {
98         rp_row_sz = 0L;
99         rp_current = 0L;
100     }
104     void shut(void) {
105         g_memory.free(rp_memory, rp_mem_sz);
106     }
111     int64_t getRowSize(void) {
112         return rp_row_sz;
113     }
120     int64_t print(char *r_ptr) {
121         int64_t sz = 0;
122         for (int64_t ii = 0; ii < rp_colnum - 1; ii++) {
123             char buf[1024];
124             sz += rp_dtype[ii]->formatTxt(buf, r_ptr + rp_offset[ii]);
125             printf("%s\t", buf);
126         }
127         printf("%c", *(r_ptr + rp_row_sz - 1));  // label of validation
128         return sz;
129     }
130
131 };  // class RPattern
132
134 class MStorage {
135
136   private:
137     int64_t  ms_record_size;
138     int64_t  ms_record_num;
139     int64_t  ms_record_max;
140     char    *ms_memory;
```

```
141     char    *ms_memory_cur;
142     int64_t  ms_memory_size;
143     char     pad[128-4*sizeof(int64_t)-sizeof(void *)];
144   public:
151     bool init(int64_t record_size) {
152         ms_record_num  = 0L;
153         ms_memory_size = 0L;
154         ms_record_max  = 0L;
155         ms_record_size = record_size;
156         if(g_memory.allocTableAddr (ms_memory)) {
157             printf ("[Mstorage][ERROR][init]: allocTableAddr error!\n");
158             return false;
159         }
160         ms_memory_cur = ms_memory;
161         return true;
162     }
169     int64_t allocRow(char *&pointer) {
170         if (ms_record_num >= ms_record_max) {
171             if (!expand()) {
172                 printf ("[Mstorage][ERROR][allowRow]: expend error! -172\n");
173                 return -1;
174             }
175         }
176         pointer = ms_memory_cur;
177         ms_memory_cur += ms_record_size;
178         return ms_record_num++;
179     }
186     char *getRow(int64_t record_rank) {
187         if (record_rank >= ms_record_num) {
188             printf("[MStorage][ERROR][getRow]: record_rank exceed! -4\n");
189             return NULL;
190         }
191         return ms_memory + record_rank*ms_record_size;
192     }
196     void shut(void) {
197         for (char *mm= ms_memory; mm< ms_memory_cur; mm+= TABLE_MEMORY_ALLOC_INC) {
198             if (munmap (mm, TABLE_MEMORY_ALLOC_INC)) {
199                 printf("[MStorage][ERROR][shut]: munmap non-table error!-50\n");
200             }
201         }
202     }
206     int64_t getRecordNum(void) {
207         return ms_record_num;
208     }
209   private:
215     bool expand(void) {
216         if (ms_memory_size > TABLE_MEMORY_ALLOC_MAX) {
217             printf("[Mstorage][ERROR][expend]: exceed maximux mmap fail! -218\n");
218             return false;
219         }
220         void* addr = mmap (ms_memory+ms_memory_size, TABLE_MEMORY_ALLOC_INC, PROT_READ|PROT_WRITE,
    MAP_PRIVATE|MAP_ANONYMOUS|MAP_FIXED, -1, 0);
221         if (addr == MAP_FAILED) {
222             printf("[Mstorage][ERROR][expend]: mmap fail! \n");
223             return false;
224         }
225         ms_memory_size += TABLE_MEMORY_ALLOC_INC;
226         ms_record_max = ms_memory_size / ms_record_size;
227         return true;
228     }
229 };  // class MStorage
230
232 class RowTable:public Table {
233   private:
234     RPattern r_pattern;
235     MStorage r_storage;
236   public:
242     RowTable(int64_t r_id, const char *r_name)
243         :Table(r_id, r_name,ROWTABLE) {
244     }
245
246     // schema operating method, if you call finish, you must not call init and add Column
247
251     bool init(void);
255     bool finish(void);
259     bool shut(void);
260
261     // data   operating method
262
    //-----------------------------------------------------------------------------------------------------------------------------
263     // select
264     // get data by record_rank, mainly for OLAP to scan
273     bool selectCol(int64_t record_rank, int64_t column_rank, char *dest);
283     bool selectCols(int64_t record_rank, int64_t column_total,
284                     int64_t * column_ranks, char *dest);
292     bool select(int64_t record_rank, char *dest);
293
```

```
294     // if you know the pointer of row by index, for OLTP to copy out the data to dest
303     bool selectCol(char *row_pointer, int64_t column_rank, char *dest);
313     bool selectCols(char *row_pointer, int64_t column_total,
314                     int64_t * column_ranks, char *dest);
322     bool select(char *row_pointer, char *dest);
323
324     // update
333     bool updateCol(char *row_pointer, int64_t column_rank, char *source);
342     bool updateCol(int64_t record_rank, int64_t column_rank, char *source);
352     bool updateCols(int64_t record_rank, int64_t column_total,
353                     int64_t * column_ranks, char *source);
363     bool updateCols(char *row_pointer, int64_t column_total,
364                     int64_t * column_ranks, char *source);
374     bool updateCols(int64_t record_rank, int64_t column_total,
375                     int64_t * column_ranks, char *source[]);
385     bool updateCols(char *row_pointer, int64_t column_total,
386                     int64_t * column_ranks, char *source[]);
387
388     // del
389
396     bool del(int64_t record_rank);
403     bool del(char *row_pointer);
404
405     // insert
406
413     bool insert(char *source);
420     bool insert(char *columns[]);
421
      //-----------------------------------------------------------------------------------------------------------------------
422
426     bool printData(void);
430     bool loadData(const char *filename);
434     RPattern & getRPattern(void) {
435         return r_pattern;
436     }
440     MStorage & getMStorage(void) {
441         return r_storage;
442     }
446     int64_t getRecordNum(void) {
447         return r_storage.getRecordNum();
448     }
455     void *getRecordPtr(int64_t row_rank) {
456         char *ptr = NULL;
457         return access(row_rank, ptr) ? ptr : NULL;
458     }
459
460   private:
468     bool access(int64_t record_rank, char *&pointer);
477     bool accessCol(int64_t record_rank, int64_t column_rank,
478                    char *&pointer);
485     bool isValid(char *record_ptr) {
486         return *(record_ptr + r_pattern.getRowSize() - 1) ==
487             'Y' ? true : false;
488     }
495     bool invalid(char *record_ptr) {
496         if (*(record_ptr + r_pattern.getRowSize() - 1) == 'N')
497             return false;
498         *(record_ptr + r_pattern.getRowSize() - 1) = 'N';
499         return true;
500     }
501 };  // class RowTable
502
503 #endif
```

## 5.47 system/runaimdb.cc File Reference

```
#include "global.h"
#include "executor.h"
#include <stdio.h>
#include <stdlib.h>
#include <vector>
```

### 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 [ ]

### 5.47.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.47.2 DESCRIPTION

the main entrance of AIMDB

### 5.47.3 Function Documentation

#### 5.47.3.1 load_data()

```
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 '\t', a row ends with '
' (2) no empty row

#### 5.47.3.2 load_schema()

```
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 '\t', a row ends with '
' (2) claim Database,Table,column,index in order (3) no empty row

#### 5.47.3.3 main()

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

#### 5.47.3.4 test()

```
int test (
            void  )
```

#### 5.47.3.5 testOne()

```
int testOne (
             int which )
```

### 5.47.4 Variable Documentation

#### 5.47.4.1 print_flag

```
int print_flag = false
```

#### 5.47.4.2 querys

```
SelectQuery querys[22]
```

----------—tpch test----------—

#### 5.47.4.3 table_name

```
const char* table_name[]
```

**Initial value:**

```
= {
    "part",
    "supplier",
    "partsupp",
    "customer",
    "nation",
    "lineitem",
    "region",
    "orders"
}
```

## 5.48 system/runaimdb.d File Reference

## 5.49 system/schema.h File Reference

```
#include <string.h>
#include <vector>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include "datatype.h"
```

## 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 }

### 5.49.1 Detailed Description

**Author**

liugang( liugang@ict.ac.cn)

**Version**

0.1

### 5.49.2 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 ()

## 5.49.3 Macro Definition Documentation

### 5.49.3.1 OBJ_NAME_MAX

```
#define OBJ_NAME_MAX (128)
```

## 5.49.4 Enumeration Type Documentation

### 5.49.4.1 ColumnType

```
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 | |

### 5.49.4.2 IndexType

```
enum IndexType
```

an enum for Index.

**Enumerator**

| | |
|---|---|
| INVID_I | |

**Enumerator**

| | |
|---|---|
| HASHINDEX | hash index |
| BPTREEINDEX | bptree index |
| ARTTREEINDEX | art tree index |
| MAXTYPE_I | |

#### 5.49.4.3 ObjectType

`enum ObjectType`

an enum for ObjectType label.

**Enumerator**

| | |
|---|---|
| INVID_O | |
| DATABASE | database |
| TABLE | table |
| COLUMN | column |
| INDEX | index |
| MAXTYPE_O | |

#### 5.49.4.4 TableType

`enum TableType`

an enum for Table.

**Enumerator**

| | |
|---|---|
| INVID_T | |
| ROWTABLE | row table |
| COLTABLE | column table |
| MAXTYPE_T | |

## 5.50 schema.h

Go to the documentation of this file.

```
1
26 #ifndef _SCHEMA_H
27 #define _SCHEMA_H
28
29 #include <string.h>
30 #include <vector>
31 #include <stdint.h>
32 #include <stdio.h>
33 #include <stdlib.h>
34 #include "datatype.h"
35
37 enum ObjectType {
38     INVID_O = 0,
39     DATABASE,
40     TABLE,
41     COLUMN,
42     INDEX,
43     MAXTYPE_O
44 };
45
47 enum IndexType {
48     INVID_I = 0,
49     HASHINDEX,
50     BPTREEINDEX,
51     ARTTREEINDEX,
52     MAXTYPE_I
53 };
54
56 enum TableType {
57     INVID_T = 0,
58     ROWTABLE,
59     COLTABLE,
60     MAXTYPE_T
61 };
62
64 enum ColumnType {
65     INVID_C = 0,
66     INT8,
67     INT16,
68     INT32,
69     INT64,
70     FLOAT32,
71     FLOAT64,
72     CHARN,
73     DATE,
74     TIME,
75     DATETIME,
76     MAXTYPE_C
77 };
78
79 #define OBJ_NAME_MAX (128)
80
82 class Object {
83   private:
84     int64_t o_id;
85     ObjectType o_type;
86     char o_name[OBJ_NAME_MAX];
88   public:
95     Object(int64_t o_id, ObjectType o_type, const char *o_name) {
96         this->o_id = o_id;
97         this->o_type = o_type;
98         strncpy(this->o_name, o_name, OBJ_NAME_MAX - 1);
99         this->o_name[OBJ_NAME_MAX - 1] = '\0';
100     }
104     virtual bool shut (void) { return false; }
108     virtual void print(void) {
109         printf("Object- o_id: %ld o_type: %d, o_name: %s\n", o_id, o_type,
110                o_name);
111     }
115     int64_t getOid(void) {
116         return o_id;
117     }
121     ObjectType getOtype(void) {
122         return o_type;
123     }
127     char *getOname(void) {
128         return o_name;
129     }
133     bool changeName(char *o_name) {
134         if (strlen(o_name) >= OBJ_NAME_MAX - 1) {
135             printf
136                 ("[Object][ERROR][changeName]: o_name exceed length! -1\n");
```

```
137             return false;
138         }
139         strncpy(this->o_name, o_name, OBJ_NAME_MAX);
140         return true;
141     }
142 };  // class Object
143
145 class Column: public Object {
146   private:
147     ColumnType c_type;
148     int64_t c_size;
149     BasicType *c_datatype;
150     int64_t c_offset;
151   public:
159     Column(int64_t c_id, const char *c_name, ColumnType c_type, int64_t c_size = 0)
160         :Object(c_id, COLUMN,c_name)
161     {
162         this->c_type = c_type;
163         this->c_size = c_size;
164         this->c_datatype = NULL;
165         this->c_offset = 0;
166     }
170     virtual ~Column(void) {
171         delete c_datatype;
172     }
176     ColumnType getCType(void) {
177         return c_type;
178     }
182     int64_t getCSize(void) {
183         return c_size;
184     }
188     int64_t getCoffset(void) {
189         return c_offset;
190     }
194     int64_t setCoffset(int64_t offset) {
195         return this->c_offset = offset;
196     }
200     virtual bool init(void) {
201         switch (c_type) {
202         case INVID_C:
203             printf("[Column][ERROR][init]: invid type! -1\n");
204             return false;
205         case INT8:
206             c_datatype = new TypeInt8();
207             break;
208         case INT16:
209             c_datatype = new TypeInt16();
210             break;
211         case INT32:
212             c_datatype = new TypeInt32();
213             break;
214         case INT64:
215             c_datatype = new TypeInt64();
216             break;
217         case FLOAT32:
218             c_datatype = new TypeFloat32();
219             break;
220         case FLOAT64:
221             c_datatype = new TypeFloat64();
222             break;
223         case CHARN:
224             c_datatype = new TypeCharN(c_size);
225             break;
226         case DATE:
227             c_datatype = new TypeDate();
228             break;              // days from 1970-01-01 till current DATE
229         case TIME:
230             c_datatype = new TypeTime();
231             break;              // seconds from 00:00:00 till current TIME
232         case DATETIME:
233             c_datatype = new TypeDateTime();
234             break;              // seconds from 1970-01-01 00:00:00 till current DATETIME
235         case MAXTYPE_C:
236             printf("[Column][ERROR][init]: invid type! -1\n");
237             return false;
238         }
239         c_size     = c_datatype->getTypeSize();
240         return true;
241     }
245     virtual bool finish(void) {
246         return true;
247     }
251     virtual bool shut(void) {
252         delete c_datatype;
253         return true;
254     }
258     virtual void print(void) {
```

```
259         printf("Column- c_id: %ld c_type: %d c_size: %ld c_name: %s c_offset: %ld\n",
260                getOid(), c_type, c_size, getOname(), getCoffset());
261     }
265     BasicType *getDataType(void) {
266         return c_datatype;
267     }
268     // HACK, if you want to implement COLTABLE, then add code here, now is for ROWTABLE
269 };  // class Column
270
271 //  Note: the table provides no key definition (primary key & foreign key)
272
274 class Table:public Object {
275   private:
276     TableType t_type;
277     std::vector < int64_t > t_columns;
278     std::vector < int64_t > t_index;
280   public:
284     TableType getTtype(void) {
285         return t_type;
286     }
290     virtual ~Table(void) {}
295     std::vector < int64_t > &getColumns(void) {
296         return t_columns;
297     }
301     std::vector < int64_t > &getIndexs(void) {
302         return t_index;
303     }
310     int64_t getColumnRank(int64_t c_id) {
311         return getRank(t_columns, c_id);
312     }
319     int64_t getIndexRank(int64_t i_id) {
320         return getRank(t_index, i_id);
321     }
329     int64_t getRank(std::vector < int64_t > &vec, int64_t id) {
330         for (unsigned int ii = 0; ii < vec.size(); ii++)
331             if (vec[ii] == id)
332                 return ii;
333         return -1;
334     }
335   public:
342     Table(int64_t t_id, const char *t_name, TableType t_type):Object(t_id, TABLE,
343           t_name)
344     {
345         this->t_type = t_type;
346     }
350     virtual void print(void) {
351         printf("Table- t_id: %ld t_type: %d t_name: %s columns: {",
352                getOid(), getTtype(), getOname());
353         unsigned int ii = 0;
354         for (; ii < t_columns.size() - 1; ii++) {
355             printf("%ld,", t_columns[ii]);
356         }
357         printf("%ld} index: {", t_columns[ii]);
358         if (t_index.size() > 0) {
359             for (ii = 0; ii < t_index.size() - 1; ii++) {
360                 printf("%ld,", t_index[ii]);
361             }
362             printf("%ld", t_index[ii]);
363         }
364         printf("}\n");
365     }
366
367     // schema operating method, if you call finish, you must not call init and add Column
368
372     virtual bool init(void) {
373         return false;
374     }
378     virtual bool addColumn(int64_t column_id) {
379         t_columns.push_back(column_id);
380         return true;
381     }
385     virtual bool addIndex(int64_t index_id) {
386         t_index.push_back(index_id);
387         return true;
388     }
392     virtual bool finish(void) {
393         return false;
394     }                           // call actual storage system, to actual inilizer
398     virtual bool shut(void) {
399         return false;
400     }
401
402     // data   operating method
403
404     // select
405     // get data by record_rank, mainly for OLAP to sacn
414     virtual bool selectCol(int64_t record_rank, int64_t column_rank,
```

```
415                             char *dest) {
416         return false;
417     }
427     virtual bool selectCols(int64_t record_rank, int64_t column_total,
428                             int64_t * column_ranks, char *dest) {
429         return false;
430     }
438     virtual bool select(int64_t record_rank, char *dest) {
439         return false;
440     }
441     // if you know the pointer of row by index, for OLTP to copy out the data to dest
450     virtual bool selectCol(char *row_pointer, int64_t column_rank,
451                             char *dest) {
452         return false;
453     }
463     virtual bool selectCols(char *row_pointer, int64_t column_total,
464                             int64_t * column_ranks, char *dest) {
465         return false;
466     }
474     virtual bool select(char *row_pointer, char *dest) {
475         return false;
476     }
477
478     // update
487     virtual bool updateCol(int64_t record_rank, int64_t column_rank,
488                             char *source) {
489         return false;
490     }
499     virtual bool updateCol(char *row_pointer, int64_t column_rank,
500                             char *source) {
501         return false;
502     }
503     /* update several column data.
504      * @param  record_rank  the n th row in the table storage
505      * @param  column_total total number of columns to select
506      * @param  column_ranks array of column_rank, column_rank is the n th column in table pattern
507      * @param  source       buffer to store data to change for
508      * @retval true         success
509      * @retval false        failure
510      */
511     virtual bool updateCols(int64_t record_rank, int64_t column_total,
512                             int64_t * column_ranks, char *source) {
513         return false;
514     }
524     virtual bool updateCols(char *row_pointer, int64_t column_total,
525                             int64_t * column_ranks, char *source) {
526         return false;
527     }
537     virtual bool updateCols(int64_t record_rank, int64_t column_total,
538                             int64_t * column_ranks, char *source[]) {
539         return false;
540     }
550     virtual bool updateCols(char *row_pointer, int64_t column_total,
551                             int64_t * column_ranks, char *source[]) {
552         return false;
553     }
554
555     // del
562     virtual bool del(int64_t record_rank) {
563         return false;
564     }
571     virtual bool del(char *row_pointer) {
572         return false;
573     }
580     virtual bool del(char *columns[]) {
581         return false;
582     }                           // this isn't support because SQL condition
583
584     // insert
591     virtual bool insert(char *source) {
592         return false;
593     }                           // the insert data are arranged in a buffer(source), in the order of and
    fixed-length pattern
600     virtual bool insert(char *columns[]) {
601         return false;
602     }                           // the insert data are arranged in different space, and we know its
    pointer
606     virtual int64_t getRecordNum(void) {
607         return -1;
608     }
612     virtual void *getRecordPtr(int64_t row_rank) {
613         return NULL;
614     }
618     virtual bool loadData(const char *filename) {
619         return false;
620     }
624     virtual bool printData(void) {
```

```
625         return false;
626     }
627 };  // class Table
628
630 class Database:public Object {
631   private:
632     std::vector < int64_t > d_table;
634   public:
640     Database(int64_t d_id, const char *d_name)
641         :Object(d_id, DATABASE,d_name) {
642     }
646     virtual ~Database(void) {}
650     virtual bool init(void) {
651         return false;
652     }
659     virtual bool addTable(int64_t table_id) {
660         d_table.push_back(table_id);
661         return true;
662     }
666     virtual bool finish(void) {
667         return false;
668     }                           // call actual storage system, to actual inilizer
672     virtual bool shut(void) {
673         return false;
674     }
678     virtual void print(void) {
679         printf("Database- d_id: %ld d_name: %s table: {", getOid(),
680                getOname());
681         unsigned int ii = 0;
682         for (; ii < d_table.size() - 1; ii++) {
683             printf("%ld,", d_table[ii]);
684         }
685         printf("%ld}\n", d_table[ii]);
686     }
690     std::vector < int64_t > &getTables(void) {
691         return d_table;
692     }
693     // insert
701     virtual bool insert(int64_t table_id, char *source) {
702         return false;
703     }
711     virtual bool insert(int64_t table_id, char *columns[]) {
712         return false;
713     }
721     virtual bool loadData(int64_t table_id, const char *filename) {
722         return false;
723     }
724 };  // class Database
725
727 class Key {
728   private:
729     std::vector < int64_t > key;
730   public:
734     Key(void) {
735     }
740     void set(std::vector < int64_t > &in_key) {
741         for (unsigned int ii = 0; ii < in_key.size(); ii++)
742             key.push_back(in_key[ii]);
743     }
750     bool contain(int64_t col_id) {
751         for (unsigned int ii = 0; ii < key.size(); ii++) {
752             if (key[ii] == col_id)
753                 return true;
754         }
755         return false;
756     }
760     Key & operator=(const Key & p) {
761         key.clear();
762         for (unsigned int ii = 0; ii < p.key.size(); ii++)
763             key.push_back(p.key[ii]);
764         return *this;
765     }
769     void print(void) {
770         printf("{");
771         unsigned int ii = 0;
772         for (; ii < key.size() - 1; ii++)
773             printf("%ld,", key[ii]);
774         printf("%ld}", key[ii]);
775     }
779     std::vector < int64_t > &getKey(void) {
780         return key;
781     }
782 };  // class Key
783
785 class Index:public Object {
786   protected:
787     IndexType i_type;
```

```
788     Key i_key;
789     int64_t i_t_id;
791   public:
799     Index(int64_t i_id, const char *i_name, IndexType i_type,
800             Key & i_key):Object(i_id, INDEX, i_name) {
801         this->i_type = i_type;
802         this->i_key = i_key;
803         this->i_t_id = 0;
804     }
808     virtual ~Index(void) {}
812     virtual bool init(void) {
813         return false;
814     }
818     virtual bool finish(void) {
819         return false;
820     }
824     virtual bool shut(void) {
825         return false;
826     }
834     virtual bool insert(void *i_data, void *p_in) {
835         return false;
836     }
844     virtual bool insert(void *i_data[], void *p_in) {
845         return false;
846     }
853     virtual bool del(void *i_data) {
854         return false;
855     }
862     virtual bool del(void *i_data[]) {
863         return false;
864     }
872     virtual bool del(void *i_data, void *p_del) {
873         return false;
874     }
882     virtual bool del(void *i_data[], void *p_del) {
883         return false;
884     }
885     virtual bool update(void *i_data, void *p_in) {
886         return false;
887     }
888     virtual bool update(void *i_data[], void *p_in) {
889         return false;
890     }
891
892     // the following function can pull one by one
901     virtual bool set_ls(void *i_data1, void *i_data2, void *info) {
902         return false;
903     }
912     virtual bool set_ls(void *i_data1[], void *i_data2[], void *info) {
913         return false;
914     }
922     virtual bool lookup(void *i_data, void *&result) {
923         return false;
924     }
932     virtual bool lookup(void *i_data[], void *&result) {
933         return false;
934     }
943     virtual bool lookup(void *i_data, void *info, void *&result) {
944         return false;
945     }
954     virtual bool lookup(void *i_data[], void *info, void *&result) {
955         return false;
956     }
964     virtual bool scan (void *info, void *&result) {
965         return false;
966     }
974     virtual bool scan_1(void *i_left, void *info) {
975         return false;
976     }
984     virtual bool scan_1(void *i_left[], void *info) {
985         return false;
986     }
995     virtual bool scan_2(void *i_right, void *info, void *&result) {
996         return false;
997     }
1006     virtual bool scan_2(void *i_right[], void *info, void *&result) {
1007         return false;
1008     }
1014     virtual int64_t tranToInt64(void *i_data) {
1015         return -1;
1016     }
1022     virtual int64_t tranToInt64(void *i_data[]) {
1023         return -1;
1024     }
1028     virtual void print(void) {
1029         printf("Index- i_id: %ld type: %d i_name: %s columns: ", getOid(),
1030                 i_type, getOname());
```

```
1031         i_key.print();
1032         printf("\n");
1033     }
1037     IndexType getIType(void) {
1038         return i_type;
1039     }
1043     Key & getIKey(void) {
1044         return i_key;
1045     }
1049     virtual void setIndexTid (int64_t tid) {
1050         this->i_t_id = tid;
1051     }
1055     virtual int64_t getIndexTid (void) {
1056         return this->i_t_id;
1057     }
1058 };  // class Index
1059
1060 #endif
```

# Index