CoGaDB: A Column-oriented GPU-accelerated DBMS

Reference Manual

Version 0.3

Otto-von-Guericke-University Magdeburg
School of Computer Science
Department of Technical and Business Information Systems
Database and Information Systems Group

Authors:

Sebastian Breß

Robin Haberkorn

Steven Ladewig

Contents

Chapter 1

Documentation

1.1 Introduction

CoGaDB is a prototype of a column-oriented GPU-accelerated database management system developed at the University of Magdeburg. Its purpose is to investigate advanced coprocessing techniques for effective GPUs utilization during database query processing. It utilizes our hybrid query processing engine (HyPE) for the physical optimization process.

CoGaDB's main purpose is to investigate a GPU-aware database architecture to achieve optimal performance of DBMS on hybrid CPU/GPU platforms. We are currently working on a architecture proposal and try to benefit from past experiences of hybrid CPU/GPU DBMS. Therefore, CoGaDB provides an extensible architecture to enable researchers an easy integration of their GPU-accelerated operators, coprocessing techniques and query optimization heuristics. Note that CoGaDB assumes that the complete database can be kept in main memory, because GPU-acceleration is not beneficial for workloads where disc I/O is the dominating factor.

1.2 Features

Currently, CoGaDB implements the following features:

- Written mainly in C++ and Cuda C
- Column-oriented in-memory database management system
- SQL Interface
- CPU and GPU operators for selection, sort, join, and simple aggregations using optimized parallel algorithms from the libraries Intel® TBB (CPU) and Thrust (GPU) and CPU only operators for projections and other management operators
- Uses HyPE, our hybrid query processing engine, for physical optimization and query processing [1, 2]

- Capable of data compression:
  - Run Length Encoding
  - Bit Vector Encoding
  - Dictionary Compression
  - Delta Coding
- NEW: SIMD Scan
- NEW: Supports filtering of strings on the GPU
- NEW: Support for primary key and foreign key integrity constraints

1.3 Supported Platforms

- Runs (currently) only on Linux: Ubuntu 12.04 (32 and 64 Bit)

1.4 Detailed Documentation

Here a list for more detailed documentation:

- **Installation** (p. 13)
- **Tutorial** (p. 15)
- **Architecture** (p. 21)
- **Concepts** (p. 25)
- **FAQ** (p. 27)

Chapter 2

Installation

2.1 Installation on Ubuntu

Just type the following command line:

```
sudo apt-get install gcc g++ make cmake doxygen doxygen-gui graphviz libboost
    -all-dev libtbb-dev libreadline6 libreadline6-dev bison
```

Alternatively, you use our installation script:

```
./setup-ubuntu.sh
```

2.2 Installation

Currently only Linux is officially supported. For Installation, download and unpack the release package. Afterwards, you have to install the necessary tools and libraries that CoGaDB uses:

- `Boost`: boost_filesystem, boost_system, boost_thread, boost_program_options
- `TBB`
- `Readline`
- `NVIDIA®CUDA®Toolkit`
- `HyPE Library`
- `Bison`

We included a setup script for ubuntu users: **setup-ubuntu.sh** (Note that CUDA has to be installed separately).

2.2.1 Building cogadb in a Terminal

Open a terminal and navigate to the directory were you unpacked CoGaDB.

Compile cogadb

```
cd gpudbms/
mkdir build
cd build
cmake ../
make
```

To run CoGaDB, issue

```
 cd build
 ./cogadb/bin/cogadbd
```

To generate and view the documentation, you can use the following commands:

```
make cogadb-doc
${BROWSER} cogadb/doc/documentation/html/index.htm
```

Chapter 3

Tutorial

In this section, we provide a short getting started guide for using CoGaDB. Furthermore, we list the available commands of CoGaDB's command line interface. Finally, we present a short demo of the SQL Interface to show its current capabilities.

3.1 Getting Started

At first, we have to create a directory, where CoGaDB can store its database:

```
set path_to_database=/home/DATA/coga_databases/ssb_sf1
```

Then, we have to create a database and import data. This can be done in two ways: using the sql interface (create table, insert into), or using a utility command. CoGaDB supports utility commands for importing databases from two common OLAP benchmarks: the TPC-H and the Star Schema Benchmark. Note that you have to generate the $*$.tbl files using the dbgen tool. Assuming we have generated a database for the star schema benchmark of scale factor one and stored the resulting $*$.tbl files in /home/DATA/benchmarks/star_schema_benchmark/SF1/, we can import the data with the following command:

```
create_ssb_database /home/DATA/benchmarks/star_schema_benchmark/SF1/
```

For the TPC-H benchmark, the command is create_tpch_database.

Now CoGaDB imports the data and stores them in the database. Depending on the scale factor, this can take a while. After the import finishes, we can start working with the database. Since CoGaDB is an in-memory database, we first have to load the database in the main memory:

```
loaddatabase
```

Then, we can start issuing queries. We can either use SQL or build in aliases for stored queries. We provide stored queries for all queries of the star schema benchmark. The

template command is ssbXY, which executes SSB-Query X.Y (X has to be a number between 1 and 4; Y has to be a number between 1 and 3 except when X is 3, in this case 4 is valid for Y as well).

Sometimes, when no NVIDIA GPU is available, we need to restrict CoGaDB to use only the CPU. We can configure this by issuing the following command:

```
setdevice cpu
```

If we want to allow CoGaDB to use both processing devices, we can replace cpu with any. It is also possible to force the usage of the GPU for all processing tasks by specifying gpu. However, this is NOT recommended, because for most complex queries, CoGaDB will not be able to perform all processing tasks on GPU only.

Now, we can launch queries:

```
CoGaDB>exec select sum(lo_extendedprice*lo_discount) as revenue from lineorder,
       dates where lo_orderdate = d_datekey and d_weeknuminyear = 6 and d_year = 1994
      and lo_discount between 5 and 7 and lo_quantity between 26 and 35;
+-------------+
| REVENUE     |
+=============+
| 2.49945e+10 |
+-------------+
1 rows

Execution Time: 155.28039 ms
```

3.2 Scripting Language

CoGaDB offers a set of commands not included in SQL to ease development and debugging:

Command	Description
loaddatabase	loads complete database in main memory
unittests	performs a self check of CoGaDB
printschema	prints the schema of the active database
showgpucache	prints status information of the GPU column cache
simple_ssb_queries	simple demonstrator for queries on SSB Benchmark data set
set <variablename>=<variablevalue>	assign the value <variablevalue> to the variable <variablename>
print <variable>	print value of variable
create_tpch_database <path to ∗.tbl files>	import tables of TPC-H benchmark in CoGaDB
create_ssb_database <path to ∗.tbl files>	import tables of star schema benchmark in CoGaDB
exec <SQL statement>="">	Execute SQL statements
explain <SQL>	Display query plan generated from SQL expression
explain_unoptimized <SQL>	As above, but does not apply logical optimizer before showing the plan
hypestatus	Prints all operations and corresponding algorithms registered in HyPE for CoGaDB's operators
integrityconstraints	Prints integrity constraints configured for current database
toggleQC	Toggle the state of Query Chopping activation. Per default QC is off.
ssbXY	Execute SSB-Query X.Y (X has to be a number between 1 and 4; Y has to be a number between 1 and 3 except when X is 3, in this case 4 is valid for Y as well)
setdevice <DEVICE>	Sets the default device, which is used for execution. Possible values are 'cpu', 'gpu' or 'any' to use either the CPU or the GPU or both concurrently.
setparallelizationmode <PARALLELIZATION mode>="">	Sets the default parallelization mode for sub-plans generated during Two Phase Physical Optimization (TOPPO) in the second phase (currently only for complex selections). Valid values are 'serial' and 'parallel'

Command	Description
about	shows credits
version	shows version of CoGaDB
quit	exits CoGaDB

CoGaDB has the following build in variables:

Variable	Description
path_to_database	absolute or relative path to directory where the database is stored
print_query_plan	print the generated query plans for a SQL query (true,false)

3.3 SQL Interface

CoGaDB supports a subset of the SQL-92 standard. We provide a short demo in the following to show the current capabilities of the SQL Interface. Note that we shortened the output of the following listings to the relevant information: query, result and execution time.

Lets first create a table:

```
CoGaDB>exec create table Test ( id int, val varchar);
TEST:
+----+-----+
| ID | VAL |
+====+=====+
+----+-----+
0 rows

Execution Time: 1.45447 ms
```

Now we can insert data:

```
CoGaDB>exec insert into Test values (0,'Car');
TEST:
+----+-----+
| ID | VAL |
+====+=====+
| 0  | Car |
+----+-----+
1 rows

Execution Time: 0.71237 ms
CoGaDB>exec insert into Test values (1,'Truck');
TEST:
+----+-------+
| ID | VAL   |
+====+=======+
| 0  | Car   |
| 1  | Truck |
+----+-------+
2 rows

Execution Time: 0.32729 ms
CoGaDB>exec insert into Test values (2,'Boat');
TEST:
+----+-------+
| ID | VAL   |
+====+=======+
| 0  | Car   |
| 1  | Truck |
```

```
| 2  | Boat  |
+----+-------+
3 rows

Execution Time: 0.36719 ms
```

Finally, we can query our table:

```
CoGaDB>exec select * from Test;
+----+-------+
| ID | VAL   |
+====+=======+
| 0  | Car   |
| 1  | Truck |
| 2  | Boat  |
+----+-------+
3 rows

Execution Time: 2.87929 ms
```

Now we show a more complex query typical for OLAP workloads. We execute query 2.3 from the Star Schema Benchmark:

```
CoGaDB>exec select sum(lo_revenue), d_year, p_brand from lineorder, dates, part
      , supplier where lo_orderdate = d_datekey and lo_partkey = p_partkey and
      lo_suppkey = s_suppkey and p_brand= 'MFGR#2239' and s_region = 'EUROPE' group by d_year
      , p_brand order by d_year, p_brand;
+--------+-----------+-------------+
| D_YEAR | P_BRAND   | LO_REVENUE  |
+========+===========+=============+
| 1992   | MFGR#2239 | 7.32066e+08 |
| 1993   | MFGR#2239 | 6.65355e+08 |
| 1994   | MFGR#2239 | 7.33858e+08 |
| 1995   | MFGR#2239 | 6.22905e+08 |
| 1996   | MFGR#2239 | 6.28615e+08 |
| 1997   | MFGR#2239 | 7.84213e+08 |
| 1998   | MFGR#2239 | 4.09671e+08 |
+--------+-----------+-------------+
7 rows
```

Chapter 4

Architecture

4.1 Overview

We now provide an overview of CoGaDB's architecture in a top down direction. As most DBMSs, CoGaDB possesses an SQL interface that can be used to launch queries. The SQL Interface constructs an abstract syntax tree, which is then converted to a logical query plan. Then, CoGaDB's logical optimizer applies a set of optimizer rules to the logical query plan to make it more efficient (e.g., it pushes down selections and resolves cross products and implicit join conditions to joins).

CoGaDB uses as physical optimizer our Hybrid Query Processing Engine (HyPE) [1, 2]. The logical plan is passed to HyPE, which has three components: a hybrid CPU/GPU optimizer, a processing device allocator and algorithm selector, and an estimation component, which estimates the execution time of an operator on a certain processing device (e.g., the CPU or the GPU). The hybrid query optimizer creates a physical query plan from a logical query plan using the algorithm selector and the cost estimator. Then, the query is executed by HyPE's execution engine. Internally, CoGaDB has to register its operators to HyPE and has to implement an adapter interface, which maps HyPE's abstract operator class to a set of functions calling the actual operators. For more information about the physical optimization phase, the interested reader is referred to the respective research papers [1, 2, 3].

Depending on the chosen processing device, data needs to be copied to the GPU. This is handled by the GPU buffer manager, which caches input columns on the GPU. If a similar query is run (which is typical for interactive data analysis), the data is already available on the GPU, which significantly accelerates query processing.

The complete query processor is build on a column-oriented, in-memory storage. In case the database does not fit into the main memory, the virtual memory manager of the operating system manages the database buffer. Similar to other main memory optimized DBMSs (e.g., MonetDB), CoGaDB processes a query operator wise. Therefore, CoGaDB executes a complete operator, which consumes its input and materializes its output. Then, the next operator is applied to the previous operators output, until all operators of a query were executed. This processing model allows for efficient caching on the CPU and for coalesced memory accesses on the GPU, which is the key for peak

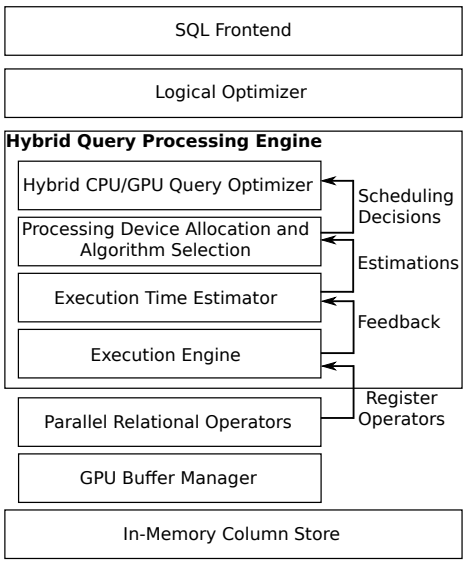

Figure 4.1: The architecture of CoGaDB

performance. Note that the storage is read only during query processing, because we do not yet support transactions. However, data can be updated offline, when no queries are processed, which fits the typical warehousing process, where data is loaded in a bulk into the database and is then analyzed.

CoGaDB's architecture is summarized in Figure 4.1.

4.2 CoGaDB's Query Interfaces

CoGaDB has a modular design, meaning it has a layered architecture, where an upper layer is implemented using the preceding layer. The most advanced way to launch queries in CoGaDB is to use the SQL interface. CoGaDB's SQL parser creates a logical query plan, consisting of operators from the Logical Operator based API. Using the logical query plan (Logical Operator based API), HyPE creates a physical query plan (which is executable) by choosing for each operator in the logical query plan a suitable physical operator from the Physical Operator based API. The Physical Operator based API provides a special interface, so HyPE can be used as execution engine. Internally, the physical operators have to execute a certain operation using the algorithm HyPE selected. This algorithms are performed in the Function based API, which contains functions that are capable of processing complex operations (e.g., selections with arbitrary combined filter predicates or groupbys with multiple aggregation functions). The complex functions are implemented using the Internal API, which consists of highly optimized primitives using libraries such as TBB or Thrust.

4.2.1 SQL Interface

- Pass queries in SQL-92 via an interactive shell
- Automatic plan generation, optimization and operator scheduling
- Utility commands not included in SQL
- Recommended API for creating queries in CoGaDB

4.2.2 Logical Operator based API

- Build queries via API
- Uses HyPE as execution engine, scheduling and query optimization is done automatically (implements the mapping layer for HyPE)
- All available operators can be found in the namespace CoGaDB::query_processing::logical_operator

4.2.3 Physical Operator based API

- Build queries via API
- Uses HyPE as execution engine, however, scheduling and query optimization is done manually (bypasses the mapping layer for HyPE)
- All available operators can be found in the namespace CoGaDB::query_processing::physical_operator

4.2.4 Function based API

- Build queries via API
- HyPE is not used for execution, so a manual execution of operators (including scheduling and query optimization) is neccessary, for hand-tuned queries (not recommended)
- Calls to CoGaDB's actual database operators
- All available functions can be found in the class **CoGaDB::BaseTable** (p. 37)

4.2.5 Internal API

- Internal functions that implement CoGaDB's actual database operators
- Work on single columns either on CPU or GPU
- Each operator has a well defined task it is optimized for (e.g., filter a column, sort a column, join two columns) and returns a list of tuple identifiers (TIDs), which are positionlists

- Functions are distributed in different modules, they can be found in the class **CoGaDB::ColumnBaseTyped** (p. 49) and CoGaDB::gpu::GPU_Operators.

4.3 Operators

We differentiate between to types of operators. Processing operators perfrom computations on the actual data and can be executed on the CPU or the GPU. Management Operators decompose complex Operations (e.g., filtering a table according to multiple and arbitrary complex selections or sorting a table after multiple columns).

4.3.1 Processing Operators

- Selection,
- Sort
- Join
- Groupby (ColumnAlgebra, AggregationFunctions)

4.3.2 Management Operators

- Projection (in a column stores it is just skipping some columns while keeping others)

Chapter 5

Concepts

In this section, we describe important concepts and design decisions in CoGaDB. We start with one of the most important building blocks of the query processor, the LookupTables. Then, we discuss the design and capabilities of CoGaDB'S optimizer, divided in the logical and physical optimizer.

5.1 Lookup Tables

A Lookup Table is a view on one or multiple tables. They are the bridge between the table-based operators and the internal column-based operators.

Internally, each operator returns the result as a list of TIDs. A LookupTable is basically a list of a pointer to a table, a pointer to a TID list, indicating which tuples of the underlying table belong to the Lookup Table, and a attribute list, specifying which columns of the table are included in the LookupTable. Therefore, LookupTables are a cheap mechanism to store intermediate results. Furthermore, they behave as they were "normal" tables, with the exception that LookupTables cannot be updated. Columns of LookupTables are LookupArrays, which consist of a pointer to a materialized column from a materialized table and a pointer to a TID list. To keep track of which LookupArray indexes a column from which table, we use a helper data structure called LookupColumn. A LookupColumn describes which part of one materialized table is part of a LookupTable, which can be the result of an arbitrary sequence of operators, including binary operators such as joins.

5.2 Logical Optimization

CoGaDB implements a simple logical optimizer. It basically implements two of the most basic optimizations: push down selections and resolve cross products by merging them with join conditions to natural joins. To achieve this, CoGaDB has currently four optimizer rules:

1. Break complex selection expressions in conjunctive normal form in a sequence

of selections consisting of at most one disjunction

2. Push down the simplefied selections as far as possible. (Either to a SCAN operator, or to a binary operator, where not all conditions in the disjunction fit completely on one subtree, which is typically the case for join conditions.)

3. Now the join conditions were pushed down far enough so they are directly over their respective CROSS_JOIN operators. Therefore, the optimizer removes the join condition, expressed by the selection, and the cross product and replaces them with a semantically equivalent JOIN operator. This process is repeated until all CROSS_JOINS are resolved.

4. In the final step, the optimizer combines succeeding selections (each only one disjunction) to complex selections in conjunctive normal form. This allows for certain optimizations in case two phase physical optimization is used.

5.3 Physical Optimization

The core of CoGaDB's physical optimization is the HyPE Library, which is our Hybrid Query Processing Engine. It allocates for each operator in a query plan a processing device and decides on the most suitable algorithm on the selected processing device. Thus, HyPE takes care of the complete physical optimization in CoGaDB.

Chapter 6

FAQ

- What is CoGaDB?
  - CoGaDB is a Column-oriented GPU-accelerated DBMS. Its purpose is to be an evaluation platform for researchers who would like to test their own GPU co-processing techniques, query optimization strategies and GPU algorithms.
- Under which License is CoGaDB distributed?
  - CoGaDB is released under the GPL v3 License. Therefore, you can download and extend it as you like as long as you obey the terms of the license.
- Can I join the project?
  - Sure, we are always looking for new project members, which help us to extend and improve CoGaDB. You should have basic knowledge in C++ and database implementation techniques. If you are interested in joining the project, contact the development team via `Sebastian Breß`.
- I have a technical problem, can I get support?
  - We offer non-commercial support for CoGaDB. In case of questions, suggestions or bug reports, feel free to contact the development team via – `Sebastian Breß`.

Bibliography

[1] S. Breß. Why it is time for a hype: A hybrid query processing engine for efficient gpu coprocessing in dbms. *The VLDB PhD workshop, PVLDB*, 6(12):1398–1403, 2013.

[2] S. Breß, F. Beier, H. Rauhe, K.-U. Sattler, E. Schallehn, and G. Saake. Efficient co-processor utilization in database query processing. *Information Systems*, 38(8):1084–1096, 2013.

[3] S. Breß, I. Geist, E. Schallehn, M. Mory, and G. Saake. A framework for cost based optimization of hybrid cpu/gpu query plans in database systems. *Control and Cybernetics*, 41(4):715–742, 2012.

Chapter 7

Class Index

7.1 Class Hierarchy

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

Chapter 8

Class Index

8.1 Class List

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

Chapter 9

Class Documentation

9.1 CoGaDB::BaseTable Class Reference

Public Types

- typedef shared_pointer_namespace::shared_ptr < **BaseTable** > **TablePtr**

Public Member Functions

- **BaseTable** (const std::string &name, const TableSchema &schema)
- const std::string & **getName** () const throw ()
- void **setName** (const std::string &) throw ()
- const TableSchema **getSchema** () const throw ()
- std::string **toString** ()
- virtual void **print** ()=0
- virtual bool **store** ()=0
- virtual bool **load** ()=0
- virtual bool **loadDatafromFile** (std::string filepath)=0
- virtual const TablePtr **materialize** () const =0
- virtual bool **addColumn** (ColumnPtr)=0
- virtual unsigned int **getNumberofRows** () const throw ()
- unsigned int **getSizeinBytes** () const throw ()
- void **printSchema** () const
- virtual bool **isMaterialized** () const =0 throw ()
- virtual const Tuple **fetchTuple** (const TID &id) const =0
- virtual bool **insert** (const Tuple &t)=0
- virtual bool **update** (const std::string &attribute_name, const boost::any &value)=0
- virtual bool **remove** (const std::string &attribute_name, const boost::any &value)=0
- bool **setPrimaryKeyConstraint** (const std::string &column_name)

- bool **hasPrimaryKeyConstraint** (const std::string &column_name) const throw ()
- bool **hasForeignKeyConstraint** (const std::string &column_name) const throw ()
- bool **setForeignKeyConstraint** (const std::string &column_name, const ForeignKeyConstraint &prim_foreign_key_reference)
- const ForeignKeyConstraint * **getForeignKeyConstraint** (const std::string &column_name)
- virtual const ColumnPtr **getColumnbyName** (const std::string &column_name) const =0 throw ()
- bool **renameColumns** (const RenameList &rename_list)

Static Public Member Functions

- static const TablePtr **createResultTable** (TablePtr table, PositionListPtr tids, MaterializationStatus mat_stat, const std::string &operation_name)
- static const TablePtr **selection** (TablePtr table, const std::string &column_name, const boost::any &value_for_comparison, const ValueComparator &comp, MaterializationStatus mat_stat=MATERIALIZE, ParallelizationMode comp_mode=SERIAL, const ComputeDevice comp_dev=CPU)
- static const TablePtr **selection** (TablePtr table, const KNF_Selection_Expression &, MaterializationStatus mat_stat=MATERIALIZE, ParallelizationMode comp_mode=SERIAL)
- static const TablePtr **selection** (TablePtr table, const Disjunction &disjunction, MaterializationStatus mat_stat=MATERIALIZE, ParallelizationMode comp_mode=SERIAL, hype::DeviceConstraint dev_constr=hype::DeviceConstraint(hype::ANY_DEVICE))
- static const TablePtr **projection** (TablePtr table, const std::list< std::string > &columns_to_select, MaterializationStatus mat_stat=MATERIALIZE, const ComputeDevice comp_dev=CPU)
- static const TablePtr **join** (TablePtr table1, const std::string &join_column_table1, TablePtr table2, const std::string &join_column_table2, JoinAlgorithm join_alg=SORT_MERGE_JOIN, MaterializationStatus mat_stat=MATERIALIZE, const ComputeDevice comp_dev=CPU)
- static const TablePtr **crossjoin** (TablePtr table1, TablePtr table2, MaterializationStatus mat_stat=MATERIALIZE)
- static const TablePtr **sort** (TablePtr table, const std::string &column_name, SortOrder order=ASCENDING, MaterializationStatus mat_stat=MATERIALIZE, ComputeDevice comp_dev=CPU)
- static const TablePtr **sort** (TablePtr table, const std::list< std::string > &column_names, SortOrder order=ASCENDING, MaterializationStatus mat_stat=MATERIALIZE, ComputeDevice comp_dev=CPU)
- static const TablePtr **groupby** (TablePtr table, const std::string &grouping_column, const std::string &aggregation_column, const std::string &result_column_name, AggregationMethod agg_meth=SUM, ComputeDevice comp_dev=CPU)
- static const TablePtr **groupby** (TablePtr table, const std::list< std::string > &grouping_columns, std::list< std::pair< std::string, AggregationMethod > > aggregation_functions, ComputeDevice comp_dev=CPU)

- static TablePtr **ColumnConstantOperation** (TablePtr tab, const std::string &col_name, const boost::any &value, const std::string &result_col_name, **ColumnAlgebraOperation** operation, const ComputeDevice comp_dev=CPU)

  adds a new column named result_col_name to the table, which is the result of col_name <operation> value
- static TablePtr **ColumnAlgebraOperation** (TablePtr tab, const std::string &col1_name, const std::string &col2_name, const std::string &result_col_name, **ColumnAlgebraOperation** operation, const ComputeDevice comp_dev=CPU)

  adds a new column named result_col_name to the table, which is the result of col1_name <operation> col2_name
- static TablePtr **AddConstantValueColumnOperation** (TablePtr tab, const std::string &col_name, AttributeType type, const boost::any &value, const ComputeDevice comp_dev=CPU)

  fills a Column #rows times with value and append to table

Protected Member Functions

- virtual const std::vector < ColumnPtr > & **getColumns** () const =0

Protected Attributes

- std::string **name_**
- TableSchema **schema_**

Friends

- class **LookupColumn**

9.1.1 Member Function Documentation

9.1.1.1 virtual bool CoGaDB::BaseTable::store ( ) [pure virtual]

tries to store **BaseTable** (p. 37) in database

9.1.1.2 virtual bool CoGaDB::BaseTable::load ( ) [pure virtual]

tries to load **BaseTable** (p. 37) form database

9.2 CoGaDB::query_processing::physical_operator::column_scan_operator Class Reference

Public Types

- typedef column_processing::cpu::TypedOperatorPtr **TypedOperatorPtr**
- typedef column_processing::cpu::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **column_scan_operator** (const hype::SchedulingDecision &sched_dec, const std::string &table_name, const std::string &column_name)
- **column_scan_operator** (const hype::SchedulingDecision &sched_dec, TablePtr table_ptr, const std::string &column_name)
- virtual bool **execute** ()
- virtual ∼**column_scan_operator** ()
- **column_scan_operator** (const hype::SchedulingDecision &sched_dec, const std::string &table_name, const std::string &column_name)
- virtual bool **execute** ()
- virtual ∼**column_scan_operator** ()

9.3 CoGaDB::ColumnBase Class Reference

This class represents a generic column, is the base class for all column classes and allows a uniform handling of columns.

Inherited by **CoGaDB::ColumnBaseTyped**< **T** >.

Public Types

- typedef shared_pointer_namespace::shared_ptr < **ColumnBase** > **ColumnPtr**

  *defines a smart pointer to a **ColumnBase** (p. 40) Object*

Public Member Functions

- **ColumnBase** (const std::string &name, AttributeType db_type)
- virtual bool **insert** (const boost::any &new_Value)=0

  appends a value new_Value to end of column
- virtual bool **update** (TID tid, const boost::any &new_Value)=0

  updates the value on position tid with a value new_Value
- virtual bool **update** (PositionListPtr tids, const boost::any &new_value)=0

  updates the values specified by the position list with a value new_Value
- virtual bool **remove** (TID tid)=0

  deletes the value on position tid
- virtual bool **remove** (PositionListPtr tid)=0

  deletes the values defined in the position list
- virtual const boost::any **get** (TID tid)=0

generic function for fetching a value form a column (slow)

- virtual std::string **getStringValue** (TID tid)=0

  generic function for fetching a string representation for a value form a column

- virtual void **print** () const =0 throw ()

  prints the content of a column

- virtual size_t **size** () const =0 throw ()

  returns the number of values (rows) in a column

- virtual unsigned int **getSizeinBytes** () const =0 throw ()

  returns the size in bytes the column consumes in main memory

- virtual const **ColumnPtr copy** () const =0

  virtual copy constructor

- virtual const **ColumnPtr gather** (PositionListPtr tid_list)=0

  creates a new column by fetching all values identified by the tid_list

- virtual const **ColumnPtr materialize** ()=0 throw ()

  materializes a column to a normal uncompressed column with dense values

- virtual const PositionListPtr **sort** (SortOrder order=ASCENDING)=0

  sorts a column w.r.t. a SortOrder

- virtual const PositionListPtr **selection** (const boost::any &value_for_comparison, const ValueComparator comp)=0

  filters the values of a column according to a filter condition consisting of a comparison value and a ValueComparator (=,<,>)

- virtual const PositionListPtr **selection** (**ColumnPtr**, const ValueComparator comp)=0

  filters the values of a column according to a filter condition consisting of a comparison column and a ValueComparator (=,<,>). This implements the comparison of two values from two columns.

- virtual const PositionListPtr **parallel_selection** (const boost::any &value_for_comparison, const ValueComparator comp, unsigned int number_of_threads)=0

  filters the values of a column in parallel according to a filter condition consisting of a comparison value and a ValueComparator (=,<,>)

- virtual const PositionListPairPtr **hash_join** (**ColumnPtr** join_column)=0

  joins two columns using the hash join algorithm

- virtual const PositionListPairPtr **parallel_hash_join** (**ColumnPtr** join_column, unsigned int number_of_threads)=0

  joins two columns using the hash join algorithm with a parallel pruning phase

- virtual const PositionListPairPtr **sort_merge_join** (**ColumnPtr** join_column)=0

  joins two columns using the sort merge join algorithm

- virtual const PositionListPairPtr **nested_loop_join** (**ColumnPtr** join_column)=0

  joins two columns using the nested loop join algorithm

- virtual bool **add** (const boost::any &new_Value)=0

  adds constant to column

- virtual bool **add** (**ColumnPtr** column)=0

  vector addition of two columns

- virtual bool **minus** (const boost::any &new_Value)=0
  substracts constant from column
- virtual bool **minus** (**ColumnPtr** column)=0
  vector substraction of two columns
- virtual bool **multiply** (const boost::any &new_Value)=0
  multiply constant with column
- virtual bool **multiply** (**ColumnPtr** column)=0
  multiply two columns A and B
- virtual bool **division** (const boost::any &new_Value)=0
  devide values in column by a constant
- virtual bool **division** (**ColumnPtr** column)=0
  devide column A with column B
- virtual bool **store** (const std::string &path)=0
  store a column on the disc
- virtual bool **load** (const std::string &path)=0
  load column from disc
- virtual bool **isMaterialized** () const =0 throw ()
  use this method to determine whether the column is materialized or a Lookup Column
- virtual bool **isCompressed** () const =0 throw ()
  use this method to determine whether the column is materialized or a Lookup Column
- virtual const std::type_info & **type** () const =0 throw ()
  returns type information of internal values
- AttributeType **getType** () const throw ()
  returns database type of column (as defined in "SQL" statement)
- const std::string **getName** () const throw ()
  returns attribute name of column
- void **setName** (const std::string &value) throw ()
  sets the attribute name of column
- virtual bool **is_equal** (**ColumnPtr** column)=0
  test this column and column for equality
- virtual int **compareValuesAtIndexes** (TID id1, TID id2)=0
  compares the values of this column on position id1 with value at position id2
- virtual bool **setPrimaryKeyConstraint** ()=0
- virtual bool **hasPrimaryKeyConstraint** () const =0 throw ()
- virtual bool **hasForeignKeyConstraint** () const =0 throw ()
- virtual bool **setForeignKeyConstraint** (const ForeignKeyConstraint &prim_-foreign_key_reference)=0
- virtual const ForeignKeyConstraint & **getForeignKeyConstraint** ()=0

Protected Attributes

- std::string **name_**
  attribute name of the column
- AttributeType **db_type_**
  database type of the column

9.3.1 Detailed Description

This class is indentended to be a base class, so it has a virtual destruktor and pure virtual methods, which need to be implemented in a derived class.

Author

Sebastian Breß

Version

0.2

Date

2013

Copyright

GNU LESSER GENERAL PUBLIC LICENSE - Version 3, `http://www.gnu.org/licenses/lgpl-3.0.txt`

9.3.2 Member Function Documentation

9.3.2.1 virtual bool CoGaDB::ColumnBase::insert (const boost::any & *new_Value*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 53), and **CoGaDB::LookupArray< T >** (p. 80).

9.3.2.2 virtual bool CoGaDB::ColumnBase::update (TID *tid,* const boost::any & *new_Value*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 54), and **CoGaDB::LookupArray< T >** (p. 81).

9.3.2.3 virtual bool CoGaDB::ColumnBase::update (PositionListPtr *tids,* const boost::any & *new_value*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 54), and **CoGaDB::LookupArray< T >** (p. 81).

9.3.2.4 virtual bool CoGaDB::ColumnBase::remove (TID *tid*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 54), and **CoGaDB::LookupArray< T >** (p. 81).

9.3.2.5 virtual bool CoGaDB::ColumnBase::remove (PositionListPtr *tid*) [pure virtual]

assumes tid list is sorted ascending

Returns

true for sucess and false in case an error occured

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 54), and **CoGaDB::LookupArray< T >** (p. 81).

9.3.2.6 virtual const boost::any CoGaDB::ColumnBase::get (TID *tid*) [pure virtual]

check whether the object is valid (e.g., when a tid is not valid, then the returned object is invalid as well)

Returns

object of type boost::any containing the value on position tid

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 54), and **CoGaDB::LookupArray< T >** (p. 81).

9.3.2.7 virtual std::string CoGaDB::ColumnBase::getStringValue (TID *tid*) [pure virtual]

Returns

string representing the value on position tid

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 55).

9.3.2.8 virtual const ColumnPtr CoGaDB::ColumnBase::copy ( ) const [pure virtual]

Returns

a ColumnPtr to an exakt copy of the current column

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 55), and **CoGaDB::LookupArray< T >** (p. 82).

9.3.2.9 virtual const ColumnPtr CoGaDB::ColumnBase::gather (PositionListPtr *tid_list*) [pure virtual]

Returns

a ColumnPtr that contains only values from the tid_list

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 55), and **CoGaDB::LookupArray< T >** (p. 82).

9.3.2.10 virtual const ColumnPtr CoGaDB::ColumnBase::materialize ( ) throw () [pure virtual]

Returns

a ColumnPtr to an materialized column

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 59), and **CoGaDB::LookupArray< T >** (p. 83).

9.3.2.11 virtual const PositionListPtr CoGaDB::ColumnBase::sort (SortOrder *order* = ASCENDING) [pure virtual]

Returns

PositionListPtr to a PositionList, which represents the result

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 55).

9.3.2.12 virtual const PositionListPtr CoGaDB::ColumnBase::selection (const boost::any & *value_for_comparison,* const ValueComparator *comp*) [pure virtual]

Returns

PositionListPtr to a PositionList, which represents the result

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 56).

9.3.2.13 virtual const PositionListPtr CoGaDB::ColumnBase::selection (ColumnPtr , const ValueComparator *comp*) [pure virtual]

Returns

PositionListPtr to a PositionList, which represents the result

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 56).

9.3.2.14 virtual const PositionListPtr CoGaDB::ColumnBase::parallel_selection (const boost::any & *value_for_comparison,* const ValueComparator *comp,* unsigned int *number_of_threads*) [pure virtual]

the additional parameter specifies the number of threads that may be used to perform the operation

Returns

PositionListPtr to a PositionList, which represents the result

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 56).

9.3.2.15 virtual const PositionListPairPtr CoGaDB::ColumnBase::hash_join (ColumnPtr *join_column*) [pure virtual]

Returns

PositionListPairPtr to a PositionListPair, which represents the result

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 56).

9.3.2.16 virtual const PositionListPairPtr CoGaDB::ColumnBase::parallel_hash_join (ColumnPtr *join_column,* unsigned int *number_of_threads*) [pure virtual]

Returns

PositionListPairPtr to a PositionListPair, which represents the result

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 56).

9.3.2.17 virtual const PositionListPairPtr CoGaDB::ColumnBase::sort_merge_join (ColumnPtr *join_column*) [pure virtual]

Returns

PositionListPairPtr to a PositionListPair, which represents the result

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 57).

9.3.2.18 virtual const PositionListPairPtr CoGaDB::ColumnBase::nested_loop_join (ColumnPtr *join_column*) [pure virtual]

Returns

PositionListPairPtr to a PositionListPair, which represents the result

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 57).

9.3.2.19 virtual bool CoGaDB::ColumnBase::add (const boost::any & *new_Value*) [pure virtual]

for all indeces i holds the following property: B[i]=A[i]+new_Value

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 60), and **CoGaDB::ColumnBaseTyped**< **T** > (p. 57).

9.3.2.20 virtual bool CoGaDB::ColumnBase::add (ColumnPtr *column*) [pure virtual]

for all indeces i holds the following property: C[i]=A[i]+B[i]

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 60), and **CoGaDB::ColumnBaseTyped**< **T** > (p. 57).

9.3.2.21 virtual bool CoGaDB::ColumnBase::minus (const boost::any & *new_Value*) [pure virtual]

for all indeces i holds the following property: B[i]=A[i]-new_Value

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 60), and **CoGaDB::ColumnBaseTyped**< **T** > (p. 57).

9.3.2.22 virtual bool CoGaDB::ColumnBase::minus (ColumnPtr *column*) [pure virtual]

for all indeces i holds the following property: C[i]=A[i]-B[i]

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 60), and **CoGaDB::ColumnBaseTyped**< **T** > (p. 57).

9.3.2.23 virtual bool CoGaDB::ColumnBase::multiply (const boost::any & *new_Value*) [pure virtual]

for all indeces i holds the following property: B[i]=A[i]∗new_Value

Implemented in **CoGaDB::ColumnBaseTyped**< **T** > (p. 60), and **CoGaDB::ColumnBaseTyped**< **T** > (p. 58).

9.3.2.24 virtual bool CoGaDB::ColumnBase::multiply (ColumnPtr *column*) [pure virtual]

for all indeces i holds the following property: C[i]=A[i]∗B[i]

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 60), and **CoGaDB::ColumnBaseTyped< T >** (p. 58).

9.3.2.25 virtual bool CoGaDB::ColumnBase::division (const boost::any & *new_Value*) [pure virtual]

for all indeces i holds the following property: B[i]=A[i]/new_Value

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 61), and **CoGaDB::ColumnBaseTyped< T >** (p. 58).

9.3.2.26 virtual bool CoGaDB::ColumnBase::division (ColumnPtr *column*) [pure virtual]

for all indeces i holds the following property: C[i]=A[i]/B[i]

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 61), and **CoGaDB::ColumnBaseTyped< T >** (p. 58).

9.3.2.27 virtual bool CoGaDB::ColumnBase::store (const std::string & *path*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 58), and **CoGaDB::LookupArray< T >** (p. 82).

9.3.2.28 virtual bool CoGaDB::ColumnBase::load (const std::string & *path*) [pure virtual]

calling load on a column that is not empty yields undefined behaviour

Returns

true for sucess and false in case an error occured

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 58), and **CoGaDB::LookupArray< T >** (p. 82).

9.3.2.29 virtual bool CoGaDB::ColumnBase::isMaterialized ( ) const throw () [pure virtual]

Returns

true in case the column is storing the plain values (without compression) and false in case the column is a LookupColumn.

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 59), and **CoGaDB::LookupArray< T >** (p. 82).

9.3.2.30 virtual bool CoGaDB::ColumnBase::isCompressed ( ) const throw () [pure virtual]

Returns

true in case the column is storing the compressed values and false otherwise.

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 59), and **CoGaDB::LookupArray< T >** (p. 83).

9.3.2.31 const std::string CoGaDB::ColumnBase::getName ( ) const throw ()

Returns

attribute name of column

9.3.2.32 virtual bool CoGaDB::ColumnBase::is_equal (ColumnPtr *column*) [pure virtual]

Returns

returns true if columns are equal and false otherwise

Implemented in **CoGaDB::ColumnBaseTyped< T >** (p. 59).

9.4 CoGaDB::ColumnBaseTyped< T > Class Template Reference

This class represents a column with type T, is the base class for all typed column classes and allows a uniform handling of columns of a certain type T.

Inherits **CoGaDB::ColumnBase**.

Inherited by **CoGaDB::LookupArray< T >**.

Public Types

- typedef boost::unordered_multimap< T, TID, boost::hash< T > , std::equal_to< T > > **HashTable**

Public Member Functions

- **ColumnBaseTyped** (const std::string &name, AttributeType db_type)
- virtual bool **insert** (const boost::any &new_Value)=0

  appends a value new_Value to end of column
- virtual bool **insert** (const T &new_Value)=0
- virtual bool **update** (TID tid, const boost::any &new_value)=0

  updates the value on position tid with a value new_Value
- virtual bool **update** (PositionListPtr tid, const boost::any &new_value)=0

  updates the values specified by the position list with a value new_Value
- virtual bool **remove** (TID tid)=0

  deletes the value on position tid
- virtual bool **remove** (PositionListPtr tid)=0

  deletes the values defined in the position list
- virtual bool **clearContent** ()=0
- virtual const boost::any **get** (TID tid)=0

  generic function for fetching a value form a column (slow)
- virtual std::string **getStringValue** (TID tid)

  generic function for fetching a string representation for a value form a column
- virtual void **print** () const =0 throw ()

  prints the content of a column
- virtual size_t **size** () const =0 throw ()

  returns the number of values (rows) in a column
- virtual unsigned int **getSizeinBytes** () const =0 throw ()

  returns the size in bytes the column consumes in main memory
- virtual const **ColumnPtr copy** () const =0

  virtual copy constructor
- virtual const **ColumnPtr gather** (PositionListPtr tid_list)=0

  creates a new column by fetching all values identified by the tid_list
- virtual const PositionListPtr **sort** (SortOrder order)

  sorts a column w.r.t. a SortOrder
- virtual const PositionListPtr **selection** (const boost::any &value_for_comparison, const ValueComparator comp)

  filters the values of a column according to a filter condition consisting of a comparison value and a ValueComparator (=,<,>)
- virtual const PositionListPtr **selection** (**ColumnPtr**, const ValueComparator comp)

  filters the values of a column according to a filter condition consisting of a comparison column and a ValueComparator (=,<,>). This implements the comparison of two values from two columns.
- virtual const PositionListPtr **parallel_selection** (const boost::any &value_for_-comparison, const ValueComparator comp, unsigned int number_of_threads)

  filters the values of a column in parallel according to a filter condition consisting of a comparison value and a ValueComparator (=,<,>)

- virtual const PositionListPtr **lock_free_parallel_selection** (const boost::any &value_for_comparison, const ValueComparator comp, unsigned int number_of-_threads)
- virtual const PositionListPairPtr **hash_join** (**ColumnPtr** join_column)

  joins two columns using the hash join algorithm
- virtual const PositionListPairPtr **parallel_hash_join** (**ColumnPtr** join_column, unsigned int number_of_threads)

  joins two columns using the hash join algorithm with a parallel pruning phase
- virtual const PositionListPairPtr **sort_merge_join** (**ColumnPtr** join_column)

  joins two columns using the sort merge join algorithm
- virtual const PositionListPairPtr **nested_loop_join** (**ColumnPtr** join_column)

  joins two columns using the nested loop join algorithm
- virtual bool **add** (const boost::any &new_Value)

  adds constant to column
- virtual bool **add** (**ColumnPtr** join_column)

  vector addition of two columns
- virtual bool **minus** (const boost::any &new_Value)

  substracts constant from column
- virtual bool **minus** (**ColumnPtr** join_column)

  vector substraction of two columns
- virtual bool **multiply** (const boost::any &new_Value)

  multiply constant with column
- virtual bool **multiply** (**ColumnPtr** join_column)

  multiply two columns A and B
- virtual bool **division** (const boost::any &new_Value)

  devide values in column by a constant
- virtual bool **division** (**ColumnPtr** join_column)

  devide column A with column B
- virtual bool **store** (const std::string &path)=0

  store a column on the disc
- virtual bool **load** (const std::string &path)=0

  load column from disc
- virtual bool **isMaterialized** () const =0 throw ()

  use this method to determine whether the column is materialized or a Lookup Column
- virtual bool **isCompressed** () const =0 throw ()

  use this method to determine whether the column is materialized or a Lookup Column
- virtual const **ColumnPtr materialize** ()=0 throw ()

  materializes a column to a normal uncompressed column with dense values
- virtual bool **is_equal** (**ColumnPtr** column)

  test this column and column for equality
- virtual int **compareValuesAtIndexes** (TID id1, TID id2)

  compares the values of this column on position id1 with value at position id2
- bool **setPrimaryKeyConstraint** ()
- bool **hasPrimaryKeyConstraint** () const throw ()

- bool **hasForeignKeyConstraint** () const throw ()
- bool **setForeignKeyConstraint** (const ForeignKeyConstraint &prim_foreign_key_reference)
- const ForeignKeyConstraint & **getForeignKeyConstraint** ()
- virtual const std::type_info & **type** () const throw ()

  returns type information of internal values
- virtual T & **operator[]** (const int index)=0

  defines operator[] for this class, which enables the user to thread all typed columns as arrays.
- bool **operator==** (**ColumnBaseTyped**< T > &column)
- template<>
  bool **add** (const boost::any &)

  adds constant to column
- template<>
  bool **add** (**ColumnPtr**)

  vector addition of two columns
- template<>
  bool **minus** (const boost::any &)

  substracts constant from column
- template<>
  bool **minus** (**ColumnPtr**)

  vector substraction of two columns
- template<>
  bool **multiply** (const boost::any &)

  multiply constant with column
- template<>
  bool **multiply** (**ColumnPtr**)

  multiply two columns A and B
- template<>
  bool **division** (const boost::any &)

  devide values in column by a constant
- template<>
  bool **division** (**ColumnPtr**)

  devide column A with column B

Static Public Member Functions

- static void **hash_join_pruning_thread** (**ColumnBaseTyped**< T > *join_column, HashTable *hashtable, unsigned int *join_tids_table1, unsigned int *join_tids_table2, unsigned int thread_id, unsigned int number_of_threads, unsigned int *result_size)
- static void **join_write_result_chunk_thread** (**ColumnBaseTyped**< T > *join_column, unsigned int *join_tids_table1, unsigned int *join_tids_table2, unsigned int *join_tids_result_table1, unsigned int *join_tids_result_table2, unsigned int thread_id, unsigned int number_of_threads, unsigned int begin_index_result, unsigned int end_index_result)

Protected Member Functions

- bool **checkUniqueness** ()
- bool **checkReferentialIntegrity** (**ColumnPtr** primary_key_col)

Protected Attributes

- bool **has_primary_key_constraint_**
- bool **has_foreign_key_constraint_**
- ForeignKeyConstraint **fk_constr_**

9.4.1 Detailed Description

template<class T>class CoGaDB::ColumnBaseTyped< T >

This class is indentended to be a base class, so it has a virtual destruktor and pure virtual methods, which need to be implemented in a derived class. Furthermore, it declares pure virtual methods to allow a generic handling of typed columns, e.g., operator[]. All algorithms can be applied to a typed column, because of this operator. This abstracts from a columns implementation detail, e.g., whether they are compressed or not.

Author

Sebastian Breß

Version

0.2

Date

2013

Copyright

GNU LESSER GENERAL PUBLIC LICENSE - Version 3, `http://www.gnu.org/licenses/lgpl-3.0.txt`

9.4.2 Member Function Documentation

9.4.2.1 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::insert (const boost::any & *new_Value*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBase** (p. 43).

Implemented in **CoGaDB::LookupArray< T >** (p. 80).

9.4.2.2 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::update (TID *tid,* const boost::any & *new_Value*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBase** (p. 43).

Implemented in **CoGaDB::LookupArray< T >** (p. 81).

9.4.2.3 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::update (PositionListPtr *tids,* const boost::any & *new_value*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBase** (p. 43).

Implemented in **CoGaDB::LookupArray< T >** (p. 81).

9.4.2.4 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::remove (TID *tid*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBase** (p. 44).

Implemented in **CoGaDB::LookupArray< T >** (p. 81).

9.4.2.5 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::remove (PositionListPtr *tid*) [pure virtual]

assumes tid list is sorted ascending

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBase** (p. 44).

Implemented in **CoGaDB::LookupArray< T >** (p. 81).

9.4.2.6 template<class T > virtual const boost::any CoGaDB::ColumnBaseTyped< T >::get (TID *tid*) [pure virtual]

check whether the object is valid (e.g., when a tid is not valid, then the returned object is invalid as well)

Returns

object of type boost::any containing the value on position tid

Implements **CoGaDB::ColumnBase** (p. 44).

Implemented in **CoGaDB::LookupArray< T >** (p. 81).

9.4.2.7 template<class T > std::string CoGaDB::ColumnBaseTyped< T >::getStringValue (TID *tid*) [virtual]

Returns

string representing the value on position tid

Implements **CoGaDB::ColumnBase** (p. 44).

9.4.2.8 template<class T > virtual const ColumnPtr CoGaDB::ColumnBaseTyped< T >::copy ( ) const [pure virtual]

Returns

a ColumnPtr to an exakt copy of the current column

Implements **CoGaDB::ColumnBase** (p. 45).

Implemented in **CoGaDB::LookupArray< T >** (p. 82).

9.4.2.9 template<class T > virtual const ColumnPtr CoGaDB::ColumnBaseTyped< T >::gather (PositionListPtr *tid_list*) [pure virtual]

Returns

a ColumnPtr that contains only values from the tid_list

Implements **CoGaDB::ColumnBase** (p. 45).

Implemented in **CoGaDB::LookupArray< T >** (p. 82).

9.4.2.10 template<class T > const PositionListPtr CoGaDB::ColumnBaseTyped< T >::sort (SortOrder *order*) [virtual]

Returns

PositionListPtr to a PositionList, which represents the result

Implements **CoGaDB::ColumnBase** (p. 45).

9.4.2.11 template<class T > const PositionListPtr CoGaDB::ColumnBaseTyped< T >::selection (const boost::any & *value_for_comparison,* const ValueComparator *comp*) [virtual]

Returns

PositionListPtr to a PositionList, which represents the result

Implements **CoGaDB::ColumnBase** (p. 45).

9.4.2.12 template<class T > const PositionListPtr CoGaDB::ColumnBaseTyped< T >::selection (ColumnPtr , const ValueComparator *comp*) [virtual]

Returns

PositionListPtr to a PositionList, which represents the result

Implements **CoGaDB::ColumnBase** (p. 46).

9.4.2.13 template<class T > virtual const PositionListPtr CoGaDB::ColumnBaseTyped< T >::parallel_selection (const boost::any & *value_for_comparison,* const ValueComparator *comp,* unsigned int *number_of_threads*) [virtual]

the additional parameter specifies the number of threads that may be used to perform the operation

Returns

PositionListPtr to a PositionList, which represents the result

Implements **CoGaDB::ColumnBase** (p. 46).

9.4.2.14 template<class T > const PositionListPairPtr CoGaDB::ColumnBaseTyped< T >::hash_join (ColumnPtr *join_column*) [virtual]

Returns

PositionListPairPtr to a PositionListPair, which represents the result

Implements **CoGaDB::ColumnBase** (p. 46).

9.4.2.15 template<class T > virtual const PositionListPairPtr CoGaDB::ColumnBaseTyped< T >::parallel_hash_join (ColumnPtr *join_column,* unsigned int *number_of_threads*) [virtual]

Returns

PositionListPairPtr to a PositionListPair, which represents the result

Implements **CoGaDB::ColumnBase** (p. 46).

9.4.2.16 template<class Type > const PositionListPairPtr CoGaDB::ColumnBaseTyped< Type >::sort_merge_join (ColumnPtr *join_column*) [virtual]

Returns

PositionListPairPtr to a PositionListPair, which represents the result

Implements **CoGaDB::ColumnBase** (p. 46).

9.4.2.17 template<class Type > const PositionListPairPtr CoGaDB::ColumnBaseTyped< Type >::nested_loop_join (ColumnPtr *join_column*) [virtual]

Returns

PositionListPairPtr to a PositionListPair, which represents the result

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.18 template<class Type > bool CoGaDB::ColumnBaseTyped< Type >::add (const boost::any & *new_Value*) [virtual]

for all indeces i holds the following property: B[i]=A[i]+new_Value

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.19 template<class Type > bool CoGaDB::ColumnBaseTyped< Type >::add (ColumnPtr *column*) [virtual]

for all indeces i holds the following property: C[i]=A[i]+B[i]

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.20 template<class Type > bool CoGaDB::ColumnBaseTyped< Type >::minus (const boost::any & *new_Value*) [virtual]

for all indeces i holds the following property: B[i]=A[i]-new_Value

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.21 template<class Type > bool CoGaDB::ColumnBaseTyped< Type >::minus (ColumnPtr *column*) [virtual]

for all indeces i holds the following property: C[i]=A[i]-B[i]

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.22 template<class Type > bool CoGaDB::ColumnBaseTyped< Type >::multiply (const boost::any & *new_Value*) [virtual]

for all indeces i holds the following property: B[i]=A[i]∗new_Value

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.23 template<class Type > bool CoGaDB::ColumnBaseTyped< Type >::multiply (ColumnPtr *column*) [virtual]

for all indeces i holds the following property: C[i]=A[i]∗B[i]

Implements **CoGaDB::ColumnBase** (p. 48).

9.4.2.24 template<class Type > bool CoGaDB::ColumnBaseTyped< Type >::division (const boost::any & *new_Value*) [virtual]

for all indeces i holds the following property: B[i]=A[i]/new_Value

Implements **CoGaDB::ColumnBase** (p. 48).

9.4.2.25 template<class Type > bool CoGaDB::ColumnBaseTyped< Type >::division (ColumnPtr *column*) [virtual]

for all indeces i holds the following property: C[i]=A[i]/B[i]

Implements **CoGaDB::ColumnBase** (p. 48).

9.4.2.26 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::store (const std::string & *path*) [pure virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBase** (p. 48).

Implemented in **CoGaDB::LookupArray< T >** (p. 82).

9.4.2.27 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::load (const std::string & *path*) [pure virtual]

calling load on a column that is not empty yields undefined behaviour

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBase** (p. 48).

Implemented in **CoGaDB::LookupArray< T >** (p. 82).

9.4.2.28 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::isMaterialized ( ) const throw () [pure virtual]

Returns

true in case the column is storing the plain values (without compression) and false in case the column is a LookupColumn.

Implements **CoGaDB::ColumnBase** (p. 48).

Implemented in **CoGaDB::LookupArray< T >** (p. 82).

9.4.2.29 template<class T > virtual bool CoGaDB::ColumnBaseTyped< T >::isCompressed ( ) const throw () [pure virtual]

Returns

true in case the column is storing the compressed values and false otherwise.

Implements **CoGaDB::ColumnBase** (p. 49).

Implemented in **CoGaDB::LookupArray< T >** (p. 83).

9.4.2.30 template<class T > virtual const ColumnPtr CoGaDB::ColumnBaseTyped< T >::materialize ( ) throw () [pure virtual]

Returns

a ColumnPtr to an materialized column

Implements **CoGaDB::ColumnBase** (p. 45).

Implemented in **CoGaDB::LookupArray< T >** (p. 83).

9.4.2.31 template<class T > bool CoGaDB::ColumnBaseTyped< T >::is_equal (ColumnPtr *column*) [virtual]

Returns

returns true if columns are equal and false otherwise

Implements **CoGaDB::ColumnBase** (p. 49).

9.4.2.32 template<class T > virtual T& CoGaDB::ColumnBaseTyped< T >::operator[] (const int *index*) [pure virtual]

Note that this method is pure virtual, so it has to be defined in a derived class.

Returns

a reference to the value at position index

Implemented in **CoGaDB::LookupArray< T >** (p. 80).

9.4.2.33 template<> bool CoGaDB::ColumnBaseTyped< std::string >::add (const boost::any & *new_Value*) [virtual]

for all indeces i holds the following property: B[i]=A[i]+new_Value

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.34 template<> bool CoGaDB::ColumnBaseTyped< std::string >::add (ColumnPtr *column*) [virtual]

for all indeces i holds the following property: C[i]=A[i]+B[i]

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.35 template<> bool CoGaDB::ColumnBaseTyped< std::string >::minus (const boost::any & *new_Value*) [virtual]

for all indeces i holds the following property: B[i]=A[i]-new_Value

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.36 template<> bool CoGaDB::ColumnBaseTyped< std::string >::minus (ColumnPtr *column*) [virtual]

for all indeces i holds the following property: C[i]=A[i]-B[i]

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.37 template<> bool CoGaDB::ColumnBaseTyped< std::string >::multiply (const boost::any & *new_Value*) [virtual]

for all indeces i holds the following property: B[i]=A[i]∗new_Value

Implements **CoGaDB::ColumnBase** (p. 47).

9.4.2.38 template<> bool CoGaDB::ColumnBaseTyped< std::string >::multiply (ColumnPtr *column*) [virtual]

for all indeces i holds the following property: C[i]=A[i]∗B[i]

Implements **CoGaDB::ColumnBase** (p. 48).

9.4.2.39 template<> bool CoGaDB::ColumnBaseTyped< std::string >::division (const boost::any & *new_Value*) [virtual]

for all indeces i holds the following property: B[i]=A[i]/new_Value

Implements **CoGaDB::ColumnBase** (p. 48).

9.4.2.40 template<> bool CoGaDB::ColumnBaseTyped< std::string >::division (ColumnPtr *column*) [virtual]

for all indeces i holds the following property: C[i]=A[i]/B[i]

Implements **CoGaDB::ColumnBase** (p. 48).

9.5 CoGaDB::query_processing::physical_operator::ColumnComparatorOperation Class Reference

Public Types

- typedef column_processing::cpu::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **ColumnComparatorOperation** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr left_child, TypedOperatorPtr right_child, Predicate op, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.6 CoGaDB::query_processing::physical_operator::CPU_AddConstantValueColumn_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_AddConstantValueColumn_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const std::string &col_name, AttributeType type, const boost::any &value)
- virtual bool **execute** ()

9.7 CoGaDB::query_processing::physical_operator::CPU_column-_constant_filter_operator Class Reference

Public Types

- typedef column_processing::cpu::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_column_constant_filter_operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const Predicate &pred)
- virtual bool **execute** ()
- virtual ∼**CPU_column_constant_filter_operator** ()

9.8 CoGaDB::query_processing::physical_operator::CPU_Column-AlgebraOperation Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < Column-Ptr > ::TypedOperatorPtr **ColumnWise_TypedOperatorPtr**
- typedef column_processing::cpu::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_ColumnAlgebraOperation** (const hype::SchedulingDecision &sched_dec, ColumnWise_TypedOperatorPtr left_child, ColumnWise_TypedOperatorPtr right-_child, ColumnAlgebraOperation op, MaterializationStatus mat_stat=MATERIAL-IZE)
- virtual bool **execute** ()
- **CPU_ColumnAlgebraOperation** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr left_child, TypedOperatorPtr right_child, ColumnAlgebra-Operation op, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.9 CoGaDB::query_processing::physical_operator::CPU_Column-AlgebraOperator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_ColumnAlgebraOperator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const std::string &column1_name, const std::string &column2_name, const std::string &result_col_name, ColumnAlgebraOperation operation)
- virtual bool **execute** ()

9.10 CoGaDB::query_processing::physical_operator::CPU_ColumnConstantOperator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_ColumnConstantOperator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, std::string column_name, const boost::any &value, const std::string &result_col_name, ColumnAlgebraOperation operation)
- virtual bool **execute** ()

9.11 CoGaDB::query_processing::physical_operator::CPU_ComplexSelection_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_ComplexSelection_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const KNF_Selection_Expression &knf_expr, hype::DeviceConstraint dev_constr=CoGaDB::RuntimeConfiguration::instance().getGlobalDeviceConstraint(), MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.12 CoGaDB::query_processing::physical_operator::CPU_CrossJoin_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_CrossJoin_Operator** (const hype::SchedulingDecision &sched_dec, - TypedOperatorPtr left_child, TypedOperatorPtr right_child, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.13 CoGaDB::query_processing::physical_operator::CPU_Groupby-_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_Groupby_Operator** (const hype::SchedulingDecision &sched_dec, Typed-OperatorPtr child, const std::list< std::string > &grouping_columns, const std-::list< ColumnAggregation > &aggregation_functions, MaterializationStatus mat-_stat=MATERIALIZE)
- virtual bool **execute** ()

9.14 CoGaDB::query_processing::physical_operator::CPU_Hash-Join_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_HashJoin_Operator** (const hype::SchedulingDecision &sched_dec, - TypedOperatorPtr left_child, TypedOperatorPtr right_child, const std::string &join-_column1_name, const std::string &join_column2_name, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.15 CoGaDB::query_processing::physical_operator::CPU_NestedLoopJoin_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_NestedLoopJoin_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr left_child, TypedOperatorPtr right_child, const std::string &join_column1_name, const std::string &join_column2_name, MaterializationStatus mat_stat=LOOKUP)
- virtual bool **execute** ()

9.16 CoGaDB::query_processing::physical_operator::CPU_Parallel_HashJoin_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_Parallel_HashJoin_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr left_child, TypedOperatorPtr right_child, const std::string &join_column1_name, const std::string &join_column2_name, MaterializationStatus mat_stat=LOOKUP)
- virtual bool **execute** ()

9.17 CoGaDB::query_processing::physical_operator::CPU_Parallel_Selection_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_ParallelSelection_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, Predicate pred, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.18 CoGaDB::query_processing::physical_operator::CPU_PositionList_Operator Class Reference

Public Types

- typedef column_processing::cpu::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_PositionList_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr left_child, TypedOperatorPtr right_child, PositionListOperation op, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.19 CoGaDB::query_processing::physical_operator::CPU_Projection_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_Projection_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const std::list< std::string > &columns_to_select, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.20 CoGaDB::query_processing::physical_operator::CPU_Selection_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_Selection_Operator** (const hype::SchedulingDecision &sched_dec, - TypedOperatorPtr child, Predicate pred, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.21 CoGaDB::query_processing::physical_operator::CPU_Sort_-Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_Sort_Operator** (const hype::SchedulingDecision &sched_dec, Typed-OperatorPtr child, const std::list< std::string > &column_names, SortOrder order=ASCENDING, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.22 CoGaDB::query_processing::physical_operator::CPU_Sort-MergeJoin_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **CPU_SortMergeJoin_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr left_child, TypedOperatorPtr right_child, const std::string &join-_column1_name, const std::string &join_column2_name, MaterializationStatus mat_stat=LOOKUP)
- virtual bool **execute** ()

9.23 CoGaDB::query_processing::physical_operator::GPU_AddConstantValueColumn_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_AddConstantValueColumn_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const std::string &col_name, AttributeType type, const boost::any &value)
- virtual bool **execute** ()

9.24 CoGaDB::query_processing::physical_operator::GPU_column_constant_filter_operator Class Reference

Public Types

- typedef column_processing::cpu::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_column_constant_filter_operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const Predicate &pred)
- virtual bool **execute** ()
- virtual ~**GPU_column_constant_filter_operator** ()

9.25 CoGaDB::query_processing::physical_operator::GPU_ColumnAlgebraOperation Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < gpu::GPU_Base_ColumnPtr > ::TypedOperatorPtr **GPU_ColumnWise_TypedOperatorPtr**

Public Member Functions

- **GPU_ColumnAlgebraOperation** (const hype::SchedulingDecision &sched_dec, GPU_ColumnWise_TypedOperatorPtr left_child, GPU_ColumnWise_TypedOperatorPtr right_child, ColumnAlgebraOperation op, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.26 CoGaDB::query_processing::physical_operator::GPU_ColumnAlgebraOperator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_ColumnAlgebraOperator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const std::string &column1_name, const std::string &column2_name, const std::string &result_col_name, ColumnAlgebraOperation operation)
- virtual bool **execute** ()

9.27 CoGaDB::query_processing::physical_operator::GPU_ColumnConstantOperator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_ColumnConstantOperator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, std::string column_name, const boost::any &value, const std::string &result_col_name, ColumnAlgebraOperation operation)
- virtual bool **execute** ()

9.28 CoGaDB::query_processing::physical_operator::GPU_Groupby_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_Groupby_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const std::list< std::string > &grouping_columns, const std::list< ColumnAggregation > &aggregation_functions, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.29 CoGaDB::query_processing::physical_operator::GPU_Join_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_Join_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr left_child, TypedOperatorPtr right_child, const std::string &join_column1_name, const std::string &join_column2_name, MaterializationStatus mat_stat=LOOKUP)
- virtual bool **execute** ()

9.30 CoGaDB::query_processing::physical_operator::GPU_Projection_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_Projection_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const std::list< std::string > &columns_to_select, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.31 CoGaDB::query_processing::physical_operator::GPU_Selection_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_Selection_Operator** (const hype::SchedulingDecision &sched_dec, - TypedOperatorPtr child, Predicate pred, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.32 CoGaDB::query_processing::physical_operator::GPU_Sort_Operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **GPU_Sort_Operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr child, const std::list< std::string > &column_names, SortOrder order=ASCENDING, MaterializationStatus mat_stat=MATERIALIZE)
- virtual bool **execute** ()

9.33 CoGaDB::query_processing::logical_operator::Logical_AddConstantValueColumn Class Reference

Public Member Functions

- **Logical_AddConstantValueColumn** (const std::string &col_name, AttributeType type, const boost::any &value)
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const std::string & **getColumnName** ()

- const AttributeType & **getAttributeType** () const
- const boost::any & **getConstantValue** () const

9.34 CoGaDB::query_processing::logical_operator::Logical_Column_Constant_Filter Class Reference

Public Member Functions

- **Logical_Column_Constant_Filter** (const Predicate &, hype::DeviceConstraint dev_constr=hype::DeviceConstraint())
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- const Predicate & **getPredicate** () const
- std::string **toString** (bool verbose) const

9.35 CoGaDB::query_processing::logical_operator::Logical_Column_Scan Class Reference

Public Member Functions

- **Logical_Column_Scan** (const std::string &table_name, const std::string &column_name)
- **Logical_Column_Scan** (TablePtr table, const std::string &column_name)
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- const std::string & **getTableName** () const
- const TablePtr **getTablePtr** ()
- const std::string & **getColumnName** () const
- std::string **toString** (bool verbose) const
- virtual column_processing::cpu::TypedOperatorPtr **getOptimalOperator** (column_processing::cpu::TypedOperatorPtr left_child, column_processing::cpu::TypedOperatorPtr right_child, hype::DeviceTypeConstraint dev_constr)

9.36 CoGaDB::query_processing::logical_operator::Logical_ColumnAlgebraOperation Class Reference

Public Member Functions

- **Logical_ColumnAlgebraOperation** (ColumnAlgebraOperation op, MaterializationStatus mat_stat=MATERIALIZE)

- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- virtual ColumnAlgebraOperation **getColumnAlgebraOperation** () const
- const MaterializationStatus & **getMaterializationStatus** () const

9.37 CoGaDB::query_processing::logical_operator::Logical_ColumnAlgebraOperator Class Reference

Public Member Functions

- **Logical_ColumnAlgebraOperator** (const std::string &column1_name, const std::string &column2_name, const std::string &result_col_name, ColumnAlgebraOperation operation, hype::DeviceConstraint dev_constr=CoGaDB::RuntimeConfiguration::instance().getGlobalDeviceConstraint())
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const std::string & **getColumn1Name** ()
- const std::string & **getColumn2Name** ()
- const std::string & **getResultColumnName** ()
- CoGaDB::ColumnAlgebraOperation **getColumnAlgebraOperation** ()

9.38 CoGaDB::query_processing::logical_operator::Logical_ColumnComparatorOperation Class Reference

Public Member Functions

- **Logical_ColumnComparatorOperation** (Predicate pred, hype::DeviceConstraint dev_constr=hype::DeviceConstraint(hype::CPU_ONLY))
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- virtual Predicate **getPredicate** () const
- const MaterializationStatus & **getMaterializationStatus** () const
- std::string **toString** (bool verbose) const

9.39 CoGaDB::query_processing::logical_operator::Logical_ColumnConstantOperator Class Reference

Public Member Functions

- **Logical_ColumnConstantOperator** (std::string column_name, const boost::any &value, const std::string &result_col_name, ColumnAlgebraOperation operation, hype::DeviceConstraint dev_constr=CoGaDB::RuntimeConfiguration::instance().getGlobalDeviceConstraint())
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const std::string & **getColumnName** ()
- const boost::any & **getValue** ()
- const std::string & **getResultColumnName** ()
- CoGaDB::ColumnAlgebraOperation **getColumnAlgebraOperation** ()

9.40 CoGaDB::query_processing::logical_operator::Logical_ComplexSelection Class Reference

Public Member Functions

- **Logical_ComplexSelection** (const KNF_Selection_Expression &knf_expr, MaterializationStatus mat_stat=MATERIALIZE, hype::DeviceConstraint dev_constr=CoGaDB::RuntimeConfiguration::instance().getGlobalDeviceConstraint())
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const KNF_Selection_Expression & **getKNF_Selection_Expression** ()
- const MaterializationStatus & **getMaterializationStatus** () const
- bool **couldNotBePushedDownFurther** ()
- void **couldNotBePushedDownFurther** (bool val)

9.41 CoGaDB::query_processing::logical_operator::Logical_CPU_ColumnAlgebraOperation Class Reference

Public Member Functions

- **Logical_CPU_ColumnAlgebraOperation** (ColumnAlgebraOperation op, MaterializationStatus mat_stat=MATERIALIZE, hype::DeviceConstraint dev_constr=hype::DeviceConstraint(hype::CPU_ONLY))
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const

- virtual ColumnAlgebraOperation **getColumnAlgebraOperation** () const
- const MaterializationStatus & **getMaterializationStatus** () const

9.42 CoGaDB::query_processing::logical_operator::Logical_Create_Table Class Reference

Public Member Functions

- **Logical_Create_Table** (const std::string &table_name, const std::string &column_name)
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- const std::string & **getTableName** () const
- const std::string & **getColumnName** () const
- void **addChild** (OperatorInputType child)
- virtual column_processing::cpu::TypedOperatorPtr **getOptimalOperator** (column_processing::cpu::TypedOperatorPtr left_child, column_processing::cpu::TypedOperatorPtr right_child, hype::DeviceTypeConstraint dev_constr)

9.43 CoGaDB::query_processing::logical_operator::Logical_CrossJoin Class Reference

Public Member Functions

- **Logical_CrossJoin** (MaterializationStatus mat_stat=MATERIALIZE, hype::DeviceConstraint dev_constr=hype::DeviceConstraint())
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- const std::string & **getLeftJoinColumnName** ()
- const std::string & **getRightJoinColumnName** ()
- const MaterializationStatus & **getMaterializationStatus** () const

9.44 CoGaDB::query_processing::logical_operator::Logical_Groupby Class Reference

Public Member Functions

- **Logical_Groupby** (const std::list< std::string > &grouping_columns, const std::list< ColumnAggregation > &aggregation_functions, MaterializationStatus mat_stat=LOOKUP, hype::DeviceConstraint dev_constr=CoGaDB::RuntimeConfiguration::instance().getGlobalDeviceConstraint())

- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const std::list< std::string > & **getGroupingColumns** ()
- const std::list < ColumnAggregation > & **getColumnAggregationFunctions** ()
- const MaterializationStatus & **getMaterializationStatus** () const

9.45 CoGaDB::query_processing::logical_operator::Logical_Join Class Reference

Public Member Functions

- **Logical_Join** (const std::string &join_column1_name, const std::string &join_column2_name, MaterializationStatus mat_stat=LOOKUP, hype::DeviceConstraint dev_constr=hype::DeviceConstraint())
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const std::string & **getLeftJoinColumnName** ()
- const std::string & **getRightJoinColumnName** ()
- const MaterializationStatus & **getMaterializationStatus** () const

9.46 CoGaDB::query_processing::logical_operator::Logical_PositionList_Operator Class Reference

Public Member Functions

- **Logical_PositionList_Operator** (PositionListOperation op, MaterializationStatus mat_stat=MATERIALIZE, hype::DeviceConstraint dev_constr=hype::DeviceConstraint(hype::CPU_ONLY))
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- virtual PositionListOperation **getPositionListOperation** () const
- const MaterializationStatus & **getMaterializationStatus** () const
- std::string **toString** (bool verbose) const

9.47 CoGaDB::query_processing::logical_operator::Logical_Projection Class Reference

Public Member Functions

- **Logical_Projection** (const std::list< std::string > &columns_to_select, MaterializationStatus mat_stat=MATERIALIZE)
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const std::list< std::string > & **getColumnList** ()
- const MaterializationStatus & **getMaterializationStatus** () const

9.48 CoGaDB::query_processing::logical_operator::Logical_Rename Class Reference

Public Member Functions

- **Logical_Rename** (const RenameList &rename_list)
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- virtual std::string **toString** (bool verbose) const
- const RenameList & **getRenameList** ()

9.49 CoGaDB::query_processing::logical_operator::Logical_Scan - Class Reference

Public Member Functions

- **Logical_Scan** (std::string table_name)
- **Logical_Scan** (TablePtr table)
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- virtual std::string **toString** (bool verbose) const
- const std::string & **getTableName** ()
- const TablePtr **getTablePtr** ()
- virtual TypedOperatorPtr **getOptimalOperator** (TypedOperatorPtr left_child, TypedOperatorPtr right_child, hype::DeviceTypeConstraint dev_constr)

9.50 CoGaDB::query_processing::logical_operator::Logical_Selection Class Reference

Public Member Functions

- **Logical_Selection** (std::string column_name, const boost::any &value_for_comparison, const ValueComparator &comp, MaterializationStatus mat_stat=LOOKUP, hype::DeviceConstraint dev_constr=hype::DeviceConstraint())
- **Logical_Selection** (Predicate pred, MaterializationStatus mat_stat=LOOKUP, hype::DeviceConstraint dev_constr=hype::DeviceConstraint())
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const Predicate & **getPredicate** ()
- const MaterializationStatus **getMaterializationStatus** () const

9.51 CoGaDB::query_processing::logical_operator::Logical_Sort Class Reference

Public Member Functions

- **Logical_Sort** (const std::list< std::string > &column_names, SortOrder order=ASCENDING, MaterializationStatus mat_stat=MATERIALIZE, hype::DeviceConstraint dev_constr=CoGaDB::RuntimeConfiguration::instance().getGlobalDeviceConstraint())
- virtual unsigned int **getOutputResultSize** () const
- virtual double **getCalculatedSelectivity** () const
- virtual std::string **getOperationName** () const
- std::string **toString** (bool verbose) const
- const std::list< std::string > & **getColumnNames** ()
- SortOrder **getSortOrder** ()
- MaterializationStatus **getMaterializationStatus** ()

Public Attributes

- std::list< std::string > **column_names_**
- SortOrder **order_**
- MaterializationStatus **mat_stat_**

9.52 CoGaDB::LookupArray< T > Class Template Reference

A **LookupArray** (p. 79) is a LookupColumn which is applied on a materialized column (of the table that is indexed by the Lookup column) and hence has a Type. This class represents a column with type T, which is essentially a tid list describing which values of a typed materialized column are included in the **LookupArray** (p. 79).

Inherits **CoGaDB::ColumnBaseTyped< T >**.

Public Member Functions

- **LookupArray** (const std::string &name, AttributeType db_type, **ColumnPtr** column, PositionListPtr tids)
- virtual bool **insert** (const boost::any &new_Value)

  appends a value new_Value to end of column
- virtual bool **insert** (const T &new_Value)
- virtual bool **update** (TID tid, const boost::any &new_value)

  updates the value on position tid with a value new_Value
- virtual bool **update** (PositionListPtr tid, const boost::any &new_value)

  updates the values specified by the position list with a value new_Value
- virtual bool **remove** (TID tid)

  deletes the value on position tid
- virtual bool **remove** (PositionListPtr tid)

  deletes the values defined in the position list
- virtual bool **clearContent** ()
- virtual const boost::any **get** (TID tid)

  generic function for fetching a value form a column (slow)
- virtual void **print** () const throw ()

  prints the content of a column
- virtual size_t **size** () const throw ()

  returns the number of values (rows) in a column
- virtual unsigned int **getSizeinBytes** () const throw ()

  returns the size in bytes the column consumes in main memory
- PositionListPtr **getPositionList** ()
- shared_pointer_namespace::shared_ptr < **ColumnBaseTyped**< T > > **getIndexedColumn** ()
- virtual const **ColumnPtr copy** () const

  virtual copy constructor
- virtual const **ColumnPtr gather** (PositionListPtr tid_list)

  creates a new column by fetching all values identified by the tid_list
- virtual bool **store** (const std::string &path)

  store a column on the disc
- virtual bool **load** (const std::string &path)

  load column from disc
- virtual bool **isMaterialized** () const throw ()

use this method to determine whether the column is materialized or a Lookup Column

- virtual bool **isCompressed** () const throw ()

  use this method to determine whether the column is materialized or a Lookup Column

- virtual const **ColumnPtr materialize** () throw ()

  materializes a column to a normal uncompressed column with dense values

- T ∗ **materializeToArray** () throw ()
- virtual T & **operator[]** (const int index)

  returns type information of internal values

9.52.1 Detailed Description

template<class T>class CoGaDB::LookupArray< T >

This class is indentended to be a base class, so it has a virtual destruktor and pure virtual methods, which need to be implemented in a derived class.

Author

Sebastian Breß

Version

0.2

Date

2013

Copyright

GNU LESSER GENERAL PUBLIC LICENSE - Version 3, `http://www.gnu.org/licenses/lgpl-3.0.txt`

9.52.2 Member Function Documentation

9.52.2.1 template<class T > bool CoGaDB::LookupArray< T >::insert (const boost::any & *new_Value*) [virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 53).

9.52.2.2 template<class T > bool CoGaDB::LookupArray< T >::update (TID *tid,* const boost::any & *new_Value*) [virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 54).

9.52.2.3 template<class T > bool CoGaDB::LookupArray< T >::update (PositionListPtr *tids,* const boost::any & *new_value*) [virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 54).

9.52.2.4 template<class T > bool CoGaDB::LookupArray< T >::remove (TID *tid*) [virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 54).

9.52.2.5 template<class T > bool CoGaDB::LookupArray< T >::remove (PositionListPtr *tid*) [virtual]

assumes tid list is sorted ascending

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 54).

9.52.2.6 template<class T > const boost::any CoGaDB::LookupArray< T >::get (TID *tid*) [virtual]

check whether the object is valid (e.g., when a tid is not valid, then the returned object is invalid as well)

Returns

object of type boost::any containing the value on position tid

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 54).

9.52.2.7 template<class T > const ColumnPtr CoGaDB::LookupArray< T >::copy ( ) const [virtual]

Returns

a ColumnPtr to an exakt copy of the current column

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 55).

9.52.2.8 template<class T > const ColumnPtr CoGaDB::LookupArray< T >::gather (PositionListPtr *tid_list*) [virtual]

Returns

a ColumnPtr that contains only values from the tid_list

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 55).

9.52.2.9 template<class T > bool CoGaDB::LookupArray< T >::store (const std::string & *path*) [virtual]

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 58).

9.52.2.10 template<class T > bool CoGaDB::LookupArray< T >::load (const std::string & *path*) [virtual]

calling load on a column that is not empty yields undefined behaviour

Returns

true for sucess and false in case an error occured

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 58).

9.52.2.11 template<class T > bool CoGaDB::LookupArray< T >::isMaterialized ( ) const throw () [virtual]

Returns

true in case the column is storing the plain values (without compression) and false in case the column is a LookupColumn.

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 59).

9.52.2.12 template<class T > bool CoGaDB::LookupArray< T >::isCompressed ( ) const throw () [virtual]

Returns

true in case the column is storing the compressed values and false otherwise.

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 59).

9.52.2.13 template<class T > const ColumnPtr CoGaDB::LookupArray< T >::materialize ( ) throw () [virtual]

Returns

a ColumnPtr to an materialized column

Implements **CoGaDB::ColumnBaseTyped< T >** (p. 59).

9.53 CoGaDB::query_processing::physical_operator::rename_operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **rename_operator** (const hype::SchedulingDecision &sched_dec, TypedOperatorPtr left_child, RenameList rename_list)
- virtual bool **execute** ()
- virtual ~**rename_operator** ()

9.54 CoGaDB::query_processing::physical_operator::scan_operator Class Reference

Public Types

- typedef hype::queryprocessing::OperatorMapper_Helper_Template < TablePtr >::TypedOperatorPtr **TypedOperatorPtr**

Public Member Functions

- **scan_operator** (const hype::SchedulingDecision &sched_dec, TablePtr table)
- virtual bool **execute** ()
- virtual ~**scan_operator** ()

9.55 CoGaDB::TBB_Body_PrefixSum Class Reference

Public Member Functions

- **TBB_Body_PrefixSum** (std::vector< int > *y_, const std::vector< int > *x_)
- int **get_sum** () const
- template<typename Tag >
  void **operator()** (const tbb::blocked_range< int > &r, Tag)
- **TBB_Body_PrefixSum** (**TBB_Body_PrefixSum** &b, tbb::split)
- void **reverse_join** (**TBB_Body_PrefixSum** &a)
- void **assign** (**TBB_Body_PrefixSum** &b)

Index