

AIMDB

Generated by Doxygen 1.9.4

Contents

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 AggreCondition Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Member Data Documentation	7
4.1.2.1 column_rank	7
4.1.2.2 method	7
4.2 BasicType Class Reference	8
4.2.1 Detailed Description	9
4.2.2 Constructor & Destructor Documentation	9
4.2.2.1 BasicType()	9
4.2.2.2 ~BasicType()	9
4.2.3 Member Function Documentation	9
4.2.3.1 cmpEQ()	9
4.2.3.2 cmpGE()	9
4.2.3.3 cmpGT()	10
4.2.3.4 cmpLE()	10

4.2.3.5 cmpLT()	10
4.2.3.6 copy()	10
4.2.3.7 formatBin()	11
4.2.3.8 formatTxt()	11
4.2.3.9 getTypeCode()	11
4.2.3.10 getTypeSize()	11
4.2.4 Member Data Documentation	11
4.2.4.1 b_type_code	11
4.2.4.2 b_type_size	12
4.3 bnode Class Reference	12
4.3.1 Member Function Documentation	12
4.3.1.1 ch()	12
4.3.1.2 chEndAddr()	12
4.3.1.3 k()	13
4.3.2 Member Data Documentation	13
4.3.2.1 child	13
4.3.2.2 key	13
4.4 Catalog Class Reference	13
4.4.1 Detailed Description	14
4.4.2 Member Function Documentation	14
4.4.2.1 createColumn()	14
4.4.2.2 createDatabase()	14
4.4.2.3 createIndex()	15
4.4.2.4 createTable()	15
4.4.2.5 getObjById()	16
4.4.2.6 getObjByName()	16
4.4.2.7 init()	16
4.4.2.8 initColumn()	17
4.4.2.9 initDatabase()	17
4.4.2.10 initIndex()	17

4.4.2.11 initTable()	18
4.4.2.12 obtainId()	18
4.4.2.13 print()	19
4.4.2.14 registerObj()	19
4.4.2.15 shut()	19
4.4.2.16 shutDatabase()	19
4.4.3 Member Data Documentation	20
4.4.3.1 cl_id_obj	20
4.4.3.2 cl_name_obj	20
4.5 Column Class Reference	20
4.5.1 Detailed Description	21
4.5.2 Constructor & Destructor Documentation	21
4.5.2.1 Column()	21
4.5.2.2 ~Column()	21
4.5.3 Member Function Documentation	21
4.5.3.1 finish()	22
4.5.3.2 getCOffset()	22
4.5.3.3 getCSize()	22
4.5.3.4 getCType()	22
4.5.3.5 getDataType()	22
4.5.3.6 init()	22
4.5.3.7 print()	22
4.5.3.8 setCOffset()	23
4.5.3.9 shut()	23
4.5.4 Member Data Documentation	23
4.5.4.1 c_datatype	23
4.5.4.2 c_offset	23
4.5.4.3 c_size	23
4.5.4.4 c_type	23
4.6 Condition Struct Reference	24

4.6.1 Detailed Description	24
4.6.2 Member Data Documentation	24
4.6.2.1 column	24
4.6.2.2 compare	24
4.6.2.3 value	24
4.7 Conditions Struct Reference	24
4.7.1 Detailed Description	25
4.7.2 Member Data Documentation	25
4.7.2.1 condition	25
4.7.2.2 condition_num	25
4.8 Database Class Reference	25
4.8.1 Detailed Description	26
4.8.2 Constructor & Destructor Documentation	26
4.8.2.1 Database()	26
4.8.2.2 ~Database()	26
4.8.3 Member Function Documentation	27
4.8.3.1 addTable()	27
4.8.3.2 finish()	27
4.8.3.3 getTables()	27
4.8.3.4 init()	27
4.8.3.5 insert() [1/2]	27
4.8.3.6 insert() [2/2]	28
4.8.3.7 loadData()	28
4.8.3.8 print()	29
4.8.3.9 shut()	29
4.8.4 Member Data Documentation	29
4.8.4.1 d_table	29
4.9 ErrorLog Class Reference	29
4.9.1 Detailed Description	31
4.9.2 Constructor & Destructor Documentation	31

4.9.2.1 ErrorLog()	31
4.9.2.2 ~ErrorLog()	31
4.9.3 Member Function Documentation	31
4.9.3.1 closeLog()	32
4.9.3.2 flushLog()	32
4.9.3.3 getErrorCode()	32
4.9.3.4 getErrorMsg()	32
4.9.3.5 getFuncNameGCC()	32
4.9.3.6 id2Name()	33
4.9.3.7 init()	33
4.9.3.8 log()	33
4.9.3.9 name2Id()	34
4.9.3.10 reset()	34
4.9.3.11 setLevel()	34
4.9.4 Member Data Documentation	35
4.9.4.1 el_bt_buffer	35
4.9.4.2 el_demangle_buf	35
4.9.4.3 el_demangle_len	35
4.9.4.4 el_err_code	35
4.9.4.5 el_fp	35
4.9.4.6 el_level	36
4.9.4.7 el_level_name	36
4.9.4.8 el_lock	36
4.9.4.9 el_logfile	36
4.9.4.10 el_msg_buf	36
4.9.4.11 el_msg_cap	36
4.9.4.12 el_msg_cur	37
4.9.4.13 el_name_2_id	37
4.9.4.14 el_thread_name	37
4.9.4.15 el_tloc	37

4.9.4.16 el_tm	37
4.10 Executor Class Reference	37
4.10.1 Detailed Description	38
4.10.2 Member Function Documentation	38
4.10.2.1 close()	38
4.10.2.2 exec()	38
4.10.2.3 findCol()	39
4.10.2.4 getRank()	39
4.10.2.5 planner()	40
4.10.3 Member Data Documentation	40
4.10.3.1 current_query	40
4.10.3.2 root	40
4.11 Filter Class Reference	40
4.11.1 Detailed Description	41
4.11.2 Constructor & Destructor Documentation	42
4.11.2.1 Filter() [1/3]	42
4.11.2.2 ~Filter()	42
4.11.2.3 Filter() [2/3]	42
4.11.2.4 Filter() [3/3]	42
4.11.3 Member Function Documentation	43
4.11.3.1 close()	43
4.11.3.2 cmpEQ()	43
4.11.3.3 cmpGE()	44
4.11.3.4 cmpGT()	44
4.11.3.5 cmpLE()	44
4.11.3.6 cmpLT()	45
4.11.3.7 cmpNE()	45
4.11.3.8 getNext()	46
4.11.3.9 initCmpFunc()	46
4.11.3.10 open()	46

4.11.3.11 setChild()	47
4.11.3.12 setColumn() [1/2]	47
4.11.3.13 setColumn() [2/2]	47
4.11.3.14 setFiltCond()	48
4.11.4 Member Data Documentation	48
4.11.4.1 buf_for_child	48
4.11.4.2 child	48
4.11.4.3 child_buf_size	48
4.11.4.4 cmp_func	49
4.11.4.5 cmp_mtd	49
4.11.4.6 cmp_table	49
4.11.4.7 filt_off	49
4.11.4.8 filt_pos	49
4.11.4.9 filt_type	49
4.11.4.10 in_tuple_size	49
4.11.4.11 input_cid	49
4.11.4.12 value	50
4.12 GrAggRecord Class Reference	50
4.12.1 Detailed Description	50
4.12.2 Constructor & Destructor Documentation	50
4.12.2.1 GrAggRecord()	50
4.12.3 Member Data Documentation	51
4.12.3.1 count	51
4.12.3.2 middle_record	51
4.12.3.3 sum	51
4.13 GroupbyAggre::group_by_hash Struct Reference	51
4.13.1 Member Function Documentation	51
4.13.1.1 operator>()	51
4.14 GroupbyAggre::group_by_key Struct Reference	52
4.14.1 Detailed Description	52

4.14.2 Member Function Documentation	52
4.14.2.1 operator==()	52
4.14.3 Member Data Documentation	52
4.14.3.1 type_array	52
4.14.3.2 value_array	52
4.15 GroupbyAggre Class Reference	53
4.15.1 Detailed Description	55
4.15.2 Member Typedef Documentation	55
4.15.2.1 group_by_hash_t	55
4.15.2.2 group_by_key_t	55
4.15.2.3 group_by_type_t	55
4.15.3 Constructor & Destructor Documentation	55
4.15.3.1 GroupbyAggre()	56
4.15.3.2 ~GroupbyAggre()	56
4.15.4 Member Function Documentation	56
4.15.4.1 avgFloat32()	56
4.15.4.2 avgFloat64()	56
4.15.4.3 avgInt16()	57
4.15.4.4 avgInt32()	57
4.15.4.5 avgInt64()	57
4.15.4.6 avgInt8()	58
4.15.4.7 close()	58
4.15.4.8 count()	58
4.15.4.9 finalCount()	59
4.15.4.10 finalFloat32Avg()	59
4.15.4.11 finalFloat32Sum()	59
4.15.4.12 finalFloat64Avg()	60
4.15.4.13 finalFloat64Sum()	60
4.15.4.14 finalInt16Sum()	60
4.15.4.15 finalInt32Sum()	61

4.15.4.16 finalInt64Sum()	61
4.15.4.17 finalInt8Sum()	61
4.15.4.18 finalIntAvg()	62
4.15.4.19 getNext()	62
4.15.4.20 initAvg()	62
4.15.4.21 initCount()	63
4.15.4.22 initFloat32Max()	63
4.15.4.23 initFloat32Min()	63
4.15.4.24 initFloat64Max()	63
4.15.4.25 initFloat64Min()	64
4.15.4.26 initInt16Max()	64
4.15.4.27 initInt16Min()	64
4.15.4.28 initInt32Max()	65
4.15.4.29 initInt32Min()	65
4.15.4.30 initInt64Max()	65
4.15.4.31 initInt64Min()	66
4.15.4.32 initInt8Max()	66
4.15.4.33 initInt8Min()	66
4.15.4.34 initSum()	67
4.15.4.35 maxFloat32()	67
4.15.4.36 maxFloat64()	67
4.15.4.37 maxInt16()	68
4.15.4.38 maxInt32()	68
4.15.4.39 maxInt64()	68
4.15.4.40 maxInt8()	69
4.15.4.41 minFloat32()	69
4.15.4.42 minFloat64()	69
4.15.4.43 minInt16()	70
4.15.4.44 minInt32()	70
4.15.4.45 minInt64()	70

4.15.4.46 minInt8()	71
4.15.4.47 open()	71
4.15.4.48 set()	71
4.15.4.49 setChild()	72
4.15.4.50 sumFloat32()	72
4.15.4.51 sumFloat64()	72
4.15.4.52 sumInt16()	73
4.15.4.53 sumInt32()	73
4.15.4.54 sumInt64()	73
4.15.4.55 sumInt8()	74
4.15.5 Member Data Documentation	74
4.15.5.1 aggr_method	74
4.15.5.2 aggr_pos	74
4.15.5.3 aggr_type	74
4.15.5.4 avg_table	74
4.15.5.5 buf_for_child	75
4.15.5.6 child	75
4.15.5.7 child_buf_size	75
4.15.5.8 child_tuple_size	75
4.15.5.9 conditions	75
4.15.5.10 final_avg_table	75
4.15.5.11 final_method	75
4.15.5.12 final_sum_table	75
4.15.5.13 group_by_pos	76
4.15.5.14 group_by_size	76
4.15.5.15 group_by_type	76
4.15.5.16 groupby_rank	76
4.15.5.17 hash_group	76
4.15.5.18 in_cid	76
4.15.5.19 init_max_table	76

4.15.5.20 init_method	77
4.15.5.21 init_min_table	77
4.15.5.22 max_table	77
4.15.5.23 middle_buf_array	77
4.15.5.24 middle_buf_size	77
4.15.5.25 middle_tuple_size	77
4.15.5.26 min_table	77
4.15.5.27 next_iter	77
4.15.5.28 out_cid	78
4.15.5.29 sum_table	78
4.16 HashCell Class Reference	78
4.16.1 Detailed Description	78
4.16.2 Member Data Documentation	78
4.16.2.1 capacity	78
4.16.2.2 ent	79
4.16.2.3 ents	79
4.16.2.4 hc_num	79
4.16.2.5	79
4.16.2.6	79
4.17 Hashcode_Ptr Class Reference	79
4.17.1 Detailed Description	80
4.17.2 Member Data Documentation	80
4.17.2.1 hash_code	80
4.17.2.2 tuple	80
4.18 HashIndex Class Reference	80
4.18.1 Detailed Description	81
4.18.2 Constructor & Destructor Documentation	81
4.18.2.1 HashIndex()	81
4.18.3 Member Function Documentation	82
4.18.3.1 addIndexDTpye()	82

4.18.3.2 cmpEQ() [1/2]	82
4.18.3.3 cmpEQ() [2/2]	83
4.18.3.4 del() [1/2]	83
4.18.3.5 del() [2/2]	83
4.18.3.6 finish()	84
4.18.3.7 hash()	84
4.18.3.8 init()	85
4.18.3.9 insert() [1/2]	85
4.18.3.10 insert() [2/2]	85
4.18.3.11 lookup() [1/2]	86
4.18.3.12 lookup() [2/2]	86
4.18.3.13 print()	87
4.18.3.14 set_ls() [1/2]	87
4.18.3.15 set_ls() [2/2]	87
4.18.3.16 setCellCap()	88
4.18.3.17 shut()	88
4.18.3.18 tranToInt64() [1/2]	89
4.18.3.19 tranToInt64() [2/2]	89
4.18.4 Member Data Documentation	89
4.18.4.1 ih_cell_capbits	89
4.18.4.2 ih_column_cap	90
4.18.4.3 ih_column_num	90
4.18.4.4 ih_datatype	90
4.18.4.5 ih_hash_bits	90
4.18.4.6 ih_hashtable	90
4.18.4.7 ih_table_offset	90
4.19 HashInfo Struct Reference	90
4.19.1 Detailed Description	91
4.19.2 Member Data Documentation	91
4.19.2.1 hash	91

4.19.2.2 last	91
4.19.2.3 ppos	91
4.19.2.4 result	91
4.19.2.5 rnum	92
4.20 HashJoin Class Reference	92
4.20.1 Detailed Description	93
4.20.2 Constructor & Destructor Documentation	93
4.20.2.1 HashJoin() [1/2]	93
4.20.2.2 ~HashJoin()	93
4.20.2.3 HashJoin() [2/2]	93
4.20.3 Member Function Documentation	93
4.20.3.1 close()	94
4.20.3.2 getNext()	94
4.20.3.3 open()	94
4.20.4 Member Data Documentation	94
4.20.4.1 hash_index	95
4.20.4.2 last_iter	95
4.20.4.3 left_buf	95
4.20.4.4 left_key_off	95
4.20.4.5 left_key_type	95
4.20.4.6 left_tuple_size	95
4.20.4.7 middle_buf_array	95
4.20.4.8 middle_buf_size	95
4.20.4.9 right_buf	96
4.20.4.10 right_buf_size	96
4.20.4.11 right_has_next	96
4.20.4.12 right_key_pos	96
4.20.4.13 right_key_type	96
4.20.4.14 right_tuple_size	96
4.20.4.15 txt_buf	96

4.20.4.16 upper_iter	97
4.21 HashTable Class Reference	97
4.21.1 Detailed Description	98
4.21.2 Constructor & Destructor Documentation	98
4.21.2.1 HashTable()	98
4.21.2.2 ~HashTable()	98
4.21.3 Member Function Documentation	98
4.21.3.1 add()	98
4.21.3.2 allocate()	99
4.21.3.3 del()	99
4.21.3.4 free()	99
4.21.3.5 probe()	100
4.21.3.6 probe_contd()	100
4.21.3.7 show()	100
4.21.3.8 size_to_slot()	101
4.21.3.9 utilization()	101
4.21.4 Member Data Documentation	101
4.21.4.1 avail	101
4.21.4.2 begin	101
4.21.4.3 end	101
4.21.4.4 estimated_duplicates_per_key	101
4.21.4.5 estimated_num_distinct_keys	102
4.21.4.6 free_header	102
4.21.4.7 initial_array_size	102
4.21.4.8 more_allocated	102
4.21.4.9 pointer2size	102
4.21.4.10 table	102
4.21.4.11 table_size	102
4.22 Index Class Reference	103
4.22.1 Detailed Description	104

4.22.2 Constructor & Destructor Documentation	104
4.22.2.1 Index()	104
4.22.2.2 ~Index()	104
4.22.3 Member Function Documentation	104
4.22.3.1 del() [1/4]	104
4.22.3.2 del() [2/4]	105
4.22.3.3 del() [3/4]	105
4.22.3.4 del() [4/4]	106
4.22.3.5 finish()	106
4.22.3.6 getlKey()	106
4.22.3.7 getIndexTid()	107
4.22.3.8 getlType()	107
4.22.3.9 init()	107
4.22.3.10 insert() [1/2]	107
4.22.3.11 insert() [2/2]	108
4.22.3.12 lookup() [1/4]	108
4.22.3.13 lookup() [2/4]	108
4.22.3.14 lookup() [3/4]	109
4.22.3.15 lookup() [4/4]	109
4.22.3.16 print()	110
4.22.3.17 scan()	110
4.22.3.18 scan_1() [1/2]	110
4.22.3.19 scan_1() [2/2]	112
4.22.3.20 scan_2() [1/2]	112
4.22.3.21 scan_2() [2/2]	113
4.22.3.22 set_ls() [1/2]	113
4.22.3.23 set_ls() [2/2]	114
4.22.3.24 setIndexTid()	114
4.22.3.25 shut()	114
4.22.3.26 tranToInt64() [1/2]	114

4.22.3.27 tranToInt64() [2/2]	115
4.22.3.28 update() [1/2]	115
4.22.3.29 update() [2/2]	115
4.22.4 Member Data Documentation	116
4.22.4.1 i_key	116
4.22.4.2 i_t_id	116
4.22.4.3 i_type	116
4.23 IndexJoin Class Reference	116
4.23.1 Detailed Description	117
4.23.2 Constructor & Destructor Documentation	117
4.23.2.1 IndexJoin() [1/2]	117
4.23.2.2 ~IndexJoin()	117
4.23.2.3 IndexJoin() [2/2]	117
4.23.3 Member Function Documentation	118
4.23.3.1 close()	118
4.23.3.2 getNext()	118
4.23.3.3 open()	118
4.23.4 Member Data Documentation	119
4.23.4.1 current_key	119
4.23.4.2 left_buf	119
4.23.4.3 left_buf_size	119
4.23.4.4 left_tuple_size	119
4.23.4.5 right_buf	119
4.23.4.6 right_buf_size	119
4.23.4.7 right_has_next	119
4.23.4.8 right_tuple_size	120
4.24 IndexScan Class Reference	120
4.24.1 Detailed Description	120
4.24.2 Constructor & Destructor Documentation	121
4.24.2.1 IndexScan() [1/2]	121

4.24.2.2 IndexScan() [2/2]	121
4.24.2.3 ~IndexScan()	121
4.24.3 Member Function Documentation	121
4.24.3.1 close()	121
4.24.3.2 getNext()	122
4.24.3.3 open()	122
4.24.3.4 setTabIdx()	122
4.24.3.5 updateKey()	123
4.24.4 Member Data Documentation	123
4.24.4.1 current_key	123
4.24.4.2 from	123
4.24.4.3 i_type	123
4.24.4.4 index	123
4.24.4.5 info_ptr	123
4.24.4.6 key_end	124
4.25 Join Class Reference	124
4.25.1 Detailed Description	125
4.25.2 Constructor & Destructor Documentation	125
4.25.2.1 Join()	125
4.25.2.2 ~Join()	125
4.25.3 Member Function Documentation	125
4.25.3.1 close()	125
4.25.3.2 getLeftCol()	126
4.25.3.3 getLeftOp()	126
4.25.3.4 getLeftRank()	126
4.25.3.5 getNext()	126
4.25.3.6 getRightCol()	127
4.25.3.7 getRightOp()	127
4.25.3.8 getRightRank()	127
4.25.3.9 open()	127

4.25.3.10 setJoinCol()	128
4.25.3.11 setLeftOp()	128
4.25.3.12 setRightOp()	128
4.25.4 Member Data Documentation	129
4.25.4.1 left	129
4.25.4.2 left_cid	129
4.25.4.3 left_rank	129
4.25.4.4 right	129
4.25.4.5 right_cid	129
4.25.4.6 right_rank	129
4.26 Key Class Reference	130
4.26.1 Detailed Description	130
4.26.2 Constructor & Destructor Documentation	130
4.26.2.1 Key()	130
4.26.3 Member Function Documentation	130
4.26.3.1 contain()	130
4.26.3.2 getKey()	131
4.26.3.3 operator=()	131
4.26.3.4 print()	131
4.26.3.5 set()	131
4.26.4 Member Data Documentation	131
4.26.4.1 key	132
4.27 Memory Class Reference	132
4.27.1 Member Function Documentation	132
4.27.1.1 alloc()	132
4.27.1.2 alloc_default()	133
4.27.1.3 allocTableAddr()	133
4.27.1.4 free()	134
4.27.1.5 init()	134
4.27.1.6 print()	134

4.27.1.7 shut()	135
4.27.1.8 slot()	135
4.27.2 Member Data Documentation	135
4.27.2.1 m_array_list	135
4.27.2.2 m_curr	135
4.27.2.3 m_head	135
4.27.2.4 m_mins	135
4.27.2.5 m_table_addr	135
4.27.2.6 m_tail	136
4.27.2.7 m_total	136
4.28 MStorage Class Reference	136
4.28.1 Detailed Description	136
4.28.2 Member Function Documentation	137
4.28.2.1 allocRow()	137
4.28.2.2 expand()	137
4.28.2.3 getRecordNum()	137
4.28.2.4 getRow()	137
4.28.2.5 init()	138
4.28.2.6 shut()	138
4.28.3 Member Data Documentation	138
4.28.3.1 ms_memory	138
4.28.3.2 ms_memory_cur	139
4.28.3.3 ms_memory_size	139
4.28.3.4 ms_record_max	139
4.28.3.5 ms_record_num	139
4.28.3.6 ms_record_size	139
4.28.3.7 pad	139
4.29 Object Class Reference	139
4.29.1 Detailed Description	140
4.29.2 Constructor & Destructor Documentation	140

4.29.2.1 Object()	140
4.29.3 Member Function Documentation	140
4.29.3.1 changeName()	140
4.29.3.2 getOid()	141
4.29.3.3 getOname()	141
4.29.3.4 getOtype()	141
4.29.3.5 print()	141
4.29.3.6 shut()	141
4.29.4 Member Data Documentation	141
4.29.4.1 o_id	142
4.29.4.2 o_name	142
4.29.4.3 o_type	142
4.30 Operator Class Reference	142
4.30.1 Constructor & Destructor Documentation	143
4.30.1.1 Operator()	143
4.30.1.2 ~Operator()	143
4.30.2 Member Function Documentation	143
4.30.2.1 close()	143
4.30.2.2 getBuffer()	143
4.30.2.3 getNext()	144
4.30.2.4 open()	144
4.30.2.5 setBuffer()	144
4.30.3 Member Data Documentation	144
4.30.3.1 buffer_from_father	145
4.31 Orderby Class Reference	145
4.31.1 Constructor & Destructor Documentation	145
4.31.1.1 Orderby()	146
4.31.1.2 ~Orderby()	146
4.31.2 Member Function Documentation	146
4.31.2.1 close()	146

4.31.2.2 getNext()	146
4.31.2.3 open()	147
4.31.2.4 set()	147
4.31.2.5 setChild()	147
4.31.3 Member Data Documentation	147
4.31.3.1 arrayid	148
4.31.3.2 child	148
4.31.3.3 child_buffer	148
4.31.3.4 colid	148
4.31.3.5 coloff	148
4.31.3.6 colrank	148
4.31.3.7 coltype	148
4.31.3.8 middle_buf_array	148
4.31.3.9 middle_buf_size	149
4.31.3.10 orderby_num	149
4.31.3.11 self_buf_size	149
4.31.3.12 tuple_size	149
4.32 Pbtrees Class Reference	149
4.32.1 Member Function Documentation	150
4.32.1.1 allocate()	150
4.32.1.2 cap2leve()	150
4.32.1.3 del()	150
4.32.1.4 free()	150
4.32.1.5 get_recptr()	150
4.32.1.6 init()	151
4.32.1.7 insert()	151
4.32.1.8 leve2cap()	151
4.32.1.9 leve2size()	151
4.32.1.10 lookup()	151
4.32.1.11 lookup_s()	151

4.32.1.12 print()	151
4.32.1.13 scan()	152
4.32.1.14 shut()	152
4.32.1.15 size2leve()	152
4.32.2 Member Data Documentation	152
4.32.2.1 p_free_header	152
4.32.2.2 p_pbtree	152
4.33 Pbtree Class Reference	152
4.33.1 Member Function Documentation	153
4.33.1.1 allocate()	153
4.33.1.2 cap2leve()	153
4.33.1.3 del()	153
4.33.1.4 free()	154
4.33.1.5 get_recptr()	154
4.33.1.6 init()	154
4.33.1.7 insert()	154
4.33.1.8 leve2cap()	154
4.33.1.9 leve2size()	154
4.33.1.10 lookup()	155
4.33.1.11 lookup_s()	155
4.33.1.12 print()	155
4.33.1.13 scan()	155
4.33.1.14 shut()	155
4.33.1.15 size2leve()	155
4.33.2 Member Data Documentation	156
4.33.2.1 p_free_header	156
4.33.2.2 p_pbtree	156
4.34 PbtreeIndex Class Reference	156
4.34.1 Constructor & Destructor Documentation	157
4.34.1.1 PbtreeIndex()	157

4.34.2 Member Function Documentation	157
4.34.2.1 del()	157
4.34.2.2 init()	158
4.34.2.3 insert()	158
4.34.2.4 lookup()	159
4.34.2.5 print()	159
4.34.2.6 scan()	160
4.34.2.7 set_Is()	160
4.34.2.8 setIndexDTpye()	161
4.34.2.9 shut()	161
4.34.3 Member Data Documentation	162
4.34.3.1 pi_datatype	162
4.34.3.2 pi_pbtree	162
4.35 PbtreeInfo Struct Reference	162
4.35.1 Detailed Description	162
4.35.2 Member Data Documentation	163
4.35.2.1 area	163
4.35.2.2 cr_area	163
4.35.2.3 cr_resu	163
4.35.2.4 l_ptr	163
4.35.2.5 le_resu	163
4.35.2.6 left	163
4.35.2.7 pos_resu	163
4.35.2.8 result	164
4.35.2.9 right	164
4.35.2.10 s_end	164
4.35.2.11 s_num	164
4.35.2.12 s_pos	164
4.35.2.13 s_ptr	164
4.36 Pointer8B Class Reference	164

4.36.1 Member Function Documentation	165
4.36.1.1 operator char *()	165
4.36.1.2 operator struct bleaf *()	165
4.36.1.3 operator struct bnode *()	165
4.36.1.4 operator unsigned long long()	165
4.36.1.5 operator void *()	166
4.36.1.6 operator=() [1/2]	166
4.36.1.7 operator=() [2/2]	166
4.36.1.8 print()	166
4.36.2 Member Data Documentation	166
4.36.2.1 value	166
4.37 Project Class Reference	166
4.37.1 Detailed Description	167
4.37.2 Constructor & Destructor Documentation	167
4.37.2.1 Project() [1/2]	167
4.37.2.2 ~Project()	168
4.37.2.3 Project() [2/2]	168
4.37.3 Member Function Documentation	168
4.37.3.1 close()	168
4.37.3.2 getColnum()	168
4.37.3.3 getNext()	169
4.37.3.4 getSchema()	169
4.37.3.5 open()	169
4.37.3.6 setChild()	169
4.37.3.7 setProjCol()	171
4.37.3.8 top()	171
4.37.4 Member Data Documentation	171
4.37.4.1 buf_for_child	171
4.37.4.2 child	171
4.37.4.3 in_buf_size	172

4.37.4.4 in_tuple_size	172
4.37.4.5 input_cid	172
4.37.4.6 input_off	172
4.37.4.7 input_pos	172
4.37.4.8 input_type	172
4.37.4.9 out_to_in	172
4.37.4.10 output_cid	172
4.37.4.11 output_type	173
4.37.4.12 output_type_buf_size	173
4.37.4.13 output_type_size	173
4.37.4.14 self_buf_size	173
4.37.4.15 topid	173
4.38 RequestColumn Struct Reference	173
4.38.1 Detailed Description	174
4.38.2 Member Data Documentation	174
4.38.2.1 aggregate_method	174
4.38.2.2 name	174
4.39 RequestTable Struct Reference	174
4.39.1 Detailed Description	174
4.39.2 Member Data Documentation	174
4.39.2.1 name	175
4.40 ResultTable Class Reference	175
4.40.1 Detailed Description	175
4.40.2 Member Function Documentation	175
4.40.2.1 append()	175
4.40.2.2 dump()	176
4.40.2.3 getRC()	176
4.40.2.4 init()	176
4.40.2.5 print()	177
4.40.2.6 shut()	177

4.40.2.7 writeRC()	177
4.40.3 Member Data Documentation	178
4.40.3.1 buffer	178
4.40.3.2 buffer_size	178
4.40.3.3 column_number	178
4.40.3.4 column_type	178
4.40.3.5 offset	178
4.40.3.6 offset_size	178
4.40.3.7 row_capacity	179
4.40.3.8 row_length	179
4.40.3.9 row_number	179
4.41 RowTable Class Reference	179
4.41.1 Detailed Description	180
4.41.2 Constructor & Destructor Documentation	180
4.41.2.1 RowTable()	180
4.41.3 Member Function Documentation	181
4.41.3.1 access()	181
4.41.3.2 accessCol()	181
4.41.3.3 del() [1/2]	182
4.41.3.4 del() [2/2]	182
4.41.3.5 finish()	183
4.41.3.6 getMStorage()	183
4.41.3.7 getRecordNum()	183
4.41.3.8 getRecordPtr()	183
4.41.3.9 getRPattern()	184
4.41.3.10 init()	184
4.41.3.11 insert() [1/2]	184
4.41.3.12 insert() [2/2]	184
4.41.3.13 invalid()	185
4.41.3.14 isValid()	185

4.41.3.15 loadData()	185
4.41.3.16 printData()	186
4.41.3.17 select() [1/2]	186
4.41.3.18 select() [2/2]	186
4.41.3.19 selectCol() [1/2]	187
4.41.3.20 selectCol() [2/2]	187
4.41.3.21 selectCols() [1/2]	188
4.41.3.22 selectCols() [2/2]	188
4.41.3.23 shut()	189
4.41.3.24 updateCol() [1/2]	189
4.41.3.25 updateCol() [2/2]	190
4.41.3.26 updateCols() [1/4]	190
4.41.3.27 updateCols() [2/4]	191
4.41.3.28 updateCols() [3/4]	191
4.41.3.29 updateCols() [4/4]	192
4.41.4 Member Data Documentation	192
4.41.4.1 r_pattern	192
4.41.4.2 r_storage	192
4.42 RPattern Class Reference	193
4.42.1 Detailed Description	193
4.42.2 Member Function Documentation	193
4.42.2.1 addColumn()	193
4.42.2.2 getColumnOffset()	194
4.42.2.3 getColumnType()	194
4.42.2.4 getRowSize()	194
4.42.2.5 init()	195
4.42.2.6 print()	195
4.42.2.7 reset()	195
4.42.2.8 shut()	195
4.42.3 Member Data Documentation	196

4.42.3.1 par	196
4.42.3.2 rp_colnum	196
4.42.3.3 rp_current	196
4.42.3.4 rp_dtype	196
4.42.3.5 rp_mem_sz	196
4.42.3.6 rp_memory	196
4.42.3.7 rp_offset	196
4.42.3.8 rp_row_sz	197
4.43 Scan Class Reference	197
4.43.1 Detailed Description	197
4.43.2 Constructor & Destructor Documentation	197
4.43.2.1 Scan()	198
4.43.2.2 ~Scan()	198
4.43.3 Member Function Documentation	198
4.43.3.1 close()	198
4.43.3.2 getNext()	198
4.43.3.3 open()	199
4.43.3.4 setTable()	199
4.43.4 Member Data Documentation	199
4.43.4.1 next_record	199
4.43.4.2 scan_table	199
4.43.4.3 total_record	200
4.44 SelectQuery Class Reference	200
4.44.1 Detailed Description	200
4.44.2 Member Data Documentation	200
4.44.2.1 database_id	200
4.44.2.2 from_number	201
4.44.2.3 from_table	201
4.44.2.4 groupby	201
4.44.2.5 groupby_number	201

4.44.2.6 having	201
4.44.2.7 orderby	201
4.44.2.8 orderby_number	201
4.44.2.9 select_column	201
4.44.2.10 select_number	202
4.44.2.11 where	202
4.45 Table Class Reference	202
4.45.1 Detailed Description	203
4.45.2 Constructor & Destructor Documentation	203
4.45.2.1 ~Table()	203
4.45.2.2 Table()	203
4.45.3 Member Function Documentation	204
4.45.3.1 addColumn()	204
4.45.3.2 addIndex()	204
4.45.3.3 del() [1/3]	204
4.45.3.4 del() [2/3]	204
4.45.3.5 del() [3/3]	205
4.45.3.6 finish()	205
4.45.3.7 getColumnRank()	205
4.45.3.8 getColumns()	206
4.45.3.9 getIndexRank()	206
4.45.3.10 getIndexes()	206
4.45.3.11 getRank()	207
4.45.3.12 getRecordNum()	207
4.45.3.13 getRecordPtr()	207
4.45.3.14 getType()	207
4.45.3.15 init()	208
4.45.3.16 insert() [1/2]	208
4.45.3.17 insert() [2/2]	208
4.45.3.18 loadData()	209

4.45.3.19 print()	209
4.45.3.20 printData()	209
4.45.3.21 select() [1/2]	209
4.45.3.22 select() [2/2]	210
4.45.3.23 selectCol() [1/2]	210
4.45.3.24 selectCol() [2/2]	211
4.45.3.25 selectCols() [1/2]	211
4.45.3.26 selectCols() [2/2]	212
4.45.3.27 shut()	212
4.45.3.28 updateCol() [1/2]	212
4.45.3.29 updateCol() [2/2]	213
4.45.3.30 updateCols() [1/4]	213
4.45.3.31 updateCols() [2/4]	214
4.45.3.32 updateCols() [3/4]	214
4.45.3.33 updateCols() [4/4]	215
4.45.4 Member Data Documentation	215
4.45.4.1 t_columns	215
4.45.4.2 t_index	215
4.45.4.3 t_type	215
4.46 TypeCharN Class Reference	216
4.46.1 Detailed Description	216
4.46.2 Constructor & Destructor Documentation	216
4.46.2.1 TypeCharN() [1/2]	216
4.46.2.2 TypeCharN() [2/2]	216
4.46.3 Member Function Documentation	217
4.46.3.1 cmpEQ()	217
4.46.3.2 cmpGE()	217
4.46.3.3 cmpGT()	217
4.46.3.4 cmpLE()	217
4.46.3.5 cmpLT()	218

4.46.3.6 copy()	218
4.46.3.7 formatBin()	218
4.46.3.8 formatTxt()	218
4.47 TypeDate Class Reference	219
4.47.1 Detailed Description	219
4.47.2 Constructor & Destructor Documentation	219
4.47.2.1 TypeDate()	219
4.47.3 Member Function Documentation	219
4.47.3.1 cmpEQ()	220
4.47.3.2 cmpGE()	220
4.47.3.3 cmpGT()	220
4.47.3.4 cmpLE()	220
4.47.3.5 cmpLT()	221
4.47.3.6 copy()	221
4.47.3.7 formatBin()	221
4.47.3.8 formatTxt()	221
4.48 TypeDateTime Class Reference	222
4.48.1 Detailed Description	222
4.48.2 Constructor & Destructor Documentation	222
4.48.2.1 TypeDateTime()	222
4.48.3 Member Function Documentation	222
4.48.3.1 cmpEQ()	223
4.48.3.2 cmpGE()	223
4.48.3.3 cmpGT()	223
4.48.3.4 cmpLE()	223
4.48.3.5 cmpLT()	224
4.48.3.6 copy()	224
4.48.3.7 formatBin()	224
4.48.3.8 formatTxt()	224
4.49 TypeFloat32 Class Reference	225

4.49.1 Detailed Description	225
4.49.2 Constructor & Destructor Documentation	225
4.49.2.1 TypeFloat32()	225
4.49.3 Member Function Documentation	225
4.49.3.1 cmpEQ()	226
4.49.3.2 cmpGE()	226
4.49.3.3 cmpGT()	226
4.49.3.4 cmpLE()	226
4.49.3.5 cmpLT()	227
4.49.3.6 copy()	227
4.49.3.7 formatBin()	227
4.49.3.8 formatTxt()	227
4.50 TypeFloat64 Class Reference	228
4.50.1 Detailed Description	228
4.50.2 Constructor & Destructor Documentation	228
4.50.2.1 TypeFloat64()	228
4.50.3 Member Function Documentation	228
4.50.3.1 cmpEQ()	229
4.50.3.2 cmpGE()	229
4.50.3.3 cmpGT()	229
4.50.3.4 cmpLE()	229
4.50.3.5 cmpLT()	230
4.50.3.6 copy()	230
4.50.3.7 formatBin()	230
4.50.3.8 formatTxt()	230
4.51 TypeInt16 Class Reference	231
4.51.1 Detailed Description	231
4.51.2 Constructor & Destructor Documentation	231
4.51.2.1 TypeInt16()	231
4.51.3 Member Function Documentation	231

4.51.3.1 cmpEQ()	232
4.51.3.2 cmpGE()	232
4.51.3.3 cmpGT()	232
4.51.3.4 cmpLE()	232
4.51.3.5 cmpLT()	233
4.51.3.6 copy()	233
4.51.3.7 formatBin()	233
4.51.3.8 formatTxt()	233
4.52 TypeInt32 Class Reference	234
4.52.1 Detailed Description	234
4.52.2 Constructor & Destructor Documentation	234
4.52.2.1 TypeInt32()	234
4.52.3 Member Function Documentation	234
4.52.3.1 cmpEQ()	235
4.52.3.2 cmpGE()	235
4.52.3.3 cmpGT()	235
4.52.3.4 cmpLE()	235
4.52.3.5 cmpLT()	236
4.52.3.6 copy()	236
4.52.3.7 formatBin()	236
4.52.3.8 formatTxt()	236
4.53 TypeInt64 Class Reference	237
4.53.1 Detailed Description	237
4.53.2 Constructor & Destructor Documentation	237
4.53.2.1 TypeInt64()	237
4.53.3 Member Function Documentation	237
4.53.3.1 cmpEQ()	238
4.53.3.2 cmpGE()	238
4.53.3.3 cmpGT()	238
4.53.3.4 cmpLE()	238

4.53.3.5 cmpLT()	239
4.53.3.6 copy()	239
4.53.3.7 formatBin()	239
4.53.3.8 formatTxt()	239
4.54 TypeInt8 Class Reference	240
4.54.1 Detailed Description	240
4.54.2 Constructor & Destructor Documentation	240
4.54.2.1 TypeInt8()	240
4.54.3 Member Function Documentation	240
4.54.3.1 cmpEQ()	241
4.54.3.2 cmpGE()	241
4.54.3.3 cmpGT()	241
4.54.3.4 cmpLE()	241
4.54.3.5 cmpLT()	242
4.54.3.6 copy()	242
4.54.3.7 formatBin()	242
4.54.3.8 formatTxt()	242
4.55 TypeTime Class Reference	243
4.55.1 Detailed Description	243
4.55.2 Constructor & Destructor Documentation	243
4.55.2.1 TypeTime()	243
4.55.3 Member Function Documentation	243
4.55.3.1 cmpEQ()	244
4.55.3.2 cmpGE()	244
4.55.3.3 cmpGT()	244
4.55.3.4 cmpLE()	244
4.55.3.5 cmpLT()	245
4.55.3.6 copy()	245
4.55.3.7 formatBin()	245
4.55.3.8 formatTxt()	245

5 File Documentation	247
5.1 system/catalog.cc File Reference	247
5.1.1 Detailed Description	247
5.1.2 DESCRIPTION	247
5.1.3 Variable Documentation	248
5.1.3.1 g_catalog	248
5.2 system/catalog.d File Reference	248
5.3 system/catalog.h File Reference	248
5.3.1 Detailed Description	248
5.3.2 DESCRIPTION	248
5.3.3 Variable Documentation	249
5.3.3.1 g_catalog	249
5.4 catalog.h	249
5.5 system/datatype.h File Reference	250
5.5.1 Detailed Description	250
5.5.2 DESCRIPTION	250
5.5.3 Enumeration Type Documentation	250
5.5.3.1 TypeCode	250
5.6 datatype.h	251
5.7 system/errorlog.cc File Reference	256
5.7.1 Detailed Description	257
5.7.2 Description	257
5.7.3 Macro Definition Documentation	257
5.7.3.1 EL_TOTAL_FILES	257
5.7.4 Variable Documentation	257
5.7.4.1 EL_src_file_name	257
5.7.4.2 thread_el	258
5.8 system/errorlog.d File Reference	258
5.9 system/errorlog.h File Reference	258
5.9.1 Detailed Description	259

5.9.2 Description	259
5.9.3 Macro Definition Documentation	260
5.9.3.1 <code>_Thread_local</code>	260
5.9.3.2 <code>EL_ASSERT</code>	260
5.9.3.3 <code>EL_BAD_FILEID</code>	260
5.9.3.4 <code>EL_DEBUG</code>	260
5.9.3.5 <code>EL_ERRCODE</code>	260
5.9.3.6 <code>EL_ERRMSG</code>	260
5.9.3.7 <code>EL_ERROR</code>	260
5.9.3.8 <code>EL_ERROR_CODE</code>	261
5.9.3.9 <code>EL_GET_FILEID</code>	261
5.9.3.10 <code>EL_GET_FILENAME</code>	261
5.9.3.11 <code>EL_GET_LINENO</code>	261
5.9.3.12 <code>EL_INFO</code>	261
5.9.3.13 <code>EL_LEVEL_COMPILE</code>	261
5.9.3.14 <code>EL_LOG_DEBUG</code>	261
5.9.3.15 <code>EL_LOG_ERROR</code>	262
5.9.3.16 <code>EL_LOG_INFO</code>	262
5.9.3.17 <code>EL_LOG_SERIOUS</code>	262
5.9.3.18 <code>EL_LOG_WARN</code>	262
5.9.3.19 <code>EL_OK</code>	262
5.9.3.20 <code>EL_RESET</code>	262
5.9.3.21 <code>EL_SERIOUS</code>	263
5.9.3.22 <code>EL_WARN</code>	263
5.9.4 Variable Documentation	263
5.9.4.1 <code>EL_src_file_name</code>	263
5.9.4.2 <code>thread_el</code>	263
5.10 <code>errorlog.h</code>	263
5.11 <code>system/executor.cc</code> File Reference	265
5.11.1 Detailed Description	266

5.11.2 DESCRIPTION	266
5.11.3 Function Documentation	266
5.11.3.1 allocColBuf()	266
5.11.3.2 easyAlloc()	266
5.11.3.3 getTupleSize()	267
5.12 system/executor.d File Reference	267
5.13 system/executor.h File Reference	267
5.13.1 Detailed Description	268
5.13.2 DESCRIPTION	268
5.13.3 Enumeration Type Documentation	268
5.13.3.1 AggregateMethod	268
5.13.3.2 CompareMethod	269
5.13.4 Function Documentation	269
5.13.4.1 easyAlloc()	269
5.14 executor.h	270
5.15 system/gcc_pf_p3.h File Reference	279
5.15.1 Macro Definition Documentation	279
5.15.1.1 pfld	279
5.15.1.2 pfldnta	279
5.15.1.3 pfst	280
5.15.1.4 pfstnta	280
5.15.1.5 prefetchnta	280
5.15.1.6 prefetcht0	280
5.15.1.7 prefetcht1	280
5.15.1.8 ptouch	280
5.16 gcc_pf_p3.h	281
5.17 system/global.cc File Reference	281
5.17.1 Function Documentation	281
5.17.1.1 global_init()	282
5.17.1.2 global_shut()	282

5.18 system/global.d File Reference	282
5.19 system/global.h File Reference	282
5.19.1 Macro Definition Documentation	282
5.19.1.1 BNODE_POINTERS_NUM	282
5.19.1.2 GLOBAL_MEMORY_MINIMUM	283
5.19.1.3 GLOBAL_MEMORY_SIZE	283
5.19.2 Function Documentation	283
5.19.2.1 global_init()	283
5.19.2.2 global_shut()	283
5.19.3 Variable Documentation	283
5.19.3.1 g_catalog	283
5.19.3.2 g_memory	283
5.20 global.h	284
5.21 system/hashindex.cc File Reference	284
5.21.1 Detailed Description	284
5.21.2 DESCRIPTION	284
5.22 system/hashindex.d File Reference	284
5.23 system/hashindex.h File Reference	284
5.23.1 Detailed Description	285
5.23.2 DESCRIPTION	285
5.23.3 Macro Definition Documentation	285
5.23.3.1 HASHINFO_CAPICITY	285
5.24 hashindex.h	286
5.25 system/hashtable.cc File Reference	286
5.25.1 Macro Definition Documentation	287
5.25.1.1 ESTIMATE_ERROR	287
5.26 system/hashtable.d File Reference	287
5.27 system/hashtable.h File Reference	287
5.27.1 Macro Definition Documentation	287
5.27.1.1 hc_capacity	287

5.27.1.2 hc_ent	288
5.27.1.3 hc_ents	288
5.28 hashtable.h	288
5.29 system/mymemory.cc File Reference	289
5.29.1 Detailed Description	289
5.29.2 DESCRIPTION	289
5.29.3 Variable Documentation	289
5.29.3.1 g_memory	289
5.30 system/mymemory.d File Reference	290
5.31 system/mymemory.h File Reference	290
5.31.1 Detailed Description	290
5.31.2 DESCRIPTION	290
5.31.3 Macro Definition Documentation	290
5.31.3.1 MEMORY_OK	291
5.31.3.2 NON_TABLE_MEMORY_ADDR	291
5.31.3.3 TABLE_MEMORY_ALLOC_INC	291
5.31.3.4 TABLE_MEMORY_ALLOC_MAX	291
5.31.3.5 TABLE_MEMORY_INIT_ADDR	291
5.31.3.6 TABLE_MEMORY_MAX_ADDR	291
5.31.4 Variable Documentation	291
5.31.4.1 g_memory	291
5.32 mymemory.h	292
5.33 system/nodepref.h File Reference	292
5.33.1 Macro Definition Documentation	292
5.33.1.1 AREA_LINE_NUM	293
5.33.1.2 BNODE_SIZE	293
5.33.1.3 CACHE_LINE_SIZE	293
5.33.1.4 ITEM_SIZE	293
5.33.1.5 L3_CACHE_LINE	293
5.33.1.6 LEAF_PREF	293

5.33.1.7 LEAF_PREF_ST	293
5.33.1.8 NODE_LINE_NUM	293
5.34 nodepref.h	294
5.35 system/pbtree.cc File Reference	297
5.35.1 Detailed Description	297
5.35.2 LICENSE	297
5.35.3 DESCRIPTION	297
5.35.4 Macro Definition Documentation	298
5.35.4.1 LEFT_KEY_NUM [1/2]	298
5.35.4.2 LEFT_KEY_NUM [2/2]	298
5.35.4.3 RIGHT_KEY_NUM [1/2]	298
5.35.4.4 RIGHT_KEY_NUM [2/2]	298
5.36 system/pbtree.d File Reference	298
5.37 system/pbtree.h File Reference	298
5.37.1 Detailed Description	299
5.37.2 LICENSE	299
5.37.3 DESCRIPTION	299
5.37.4 Macro Definition Documentation	299
5.37.4.1 BKEY_NUM	299
5.37.4.2 bleaf	300
5.37.4.3 bnext	300
5.37.4.4 bnum	300
5.37.4.5 KEY_SIZE	300
5.37.4.6 LEAF_KEY_NUM	300
5.37.4.7 MAX_KEY	300
5.37.4.8 MIN_KEY	300
5.37.4.9 NON_LEAF_KEY_NUM	301
5.37.4.10 POINTER8B_SIZE	301
5.37.4.11 POINTER_SIZE	301
5.37.5 Typedef Documentation	301

5.37.5.1 <code>key_type</code>	301
5.38 <code>pbtrees.h</code>	301
5.39 <code>system/pbtreesindex.cc</code> File Reference	305
5.40 <code>system/pbtreesindex.d</code> File Reference	305
5.41 <code>system/pbtreesindex.h</code> File Reference	305
5.41.1 Detailed Description	306
5.41.2 DESCRIPTION	306
5.41.3 Macro Definition Documentation	306
5.41.3.1 <code>PBTREEINFO_CAPACITY</code>	306
5.42 <code>pbtreesindex.h</code>	306
5.43 <code>system/rowtable.cc</code> File Reference	307
5.43.1 Detailed Description	307
5.43.2 DESCRIPTION	307
5.44 <code>system/rowtable.d</code> File Reference	307
5.45 <code>system/rowtable.h</code> File Reference	307
5.45.1 Detailed Description	308
5.45.2 DESCRIPTION	308
5.45.3 Variable Documentation	308
5.45.3.1 <code>g_memory</code>	308
5.46 <code>rowtable.h</code>	309
5.47 <code>system/runaimdb.cc</code> File Reference	311
5.47.1 Detailed Description	312
5.47.2 DESCRIPTION	312
5.47.3 Function Documentation	312
5.47.3.1 <code>load_data()</code>	312
5.47.3.2 <code>load_schema()</code>	313
5.47.3.3 <code>main()</code>	313
5.47.3.4 <code>test()</code>	313
5.47.3.5 <code>testOne()</code>	314
5.47.4 Variable Documentation	314

5.47.4.1 print_flag	314
5.47.4.2 querys	314
5.47.4.3 table_name	314
5.48 system/runaimdb.d File Reference	314
5.49 system/schema.h File Reference	314
5.49.1 Detailed Description	315
5.49.2 DESCRIPTION	315
5.49.3 Macro Definition Documentation	316
5.49.3.1 OBJ_NAME_MAX	316
5.49.4 Enumeration Type Documentation	316
5.49.4.1 ColumnType	316
5.49.4.2 IndexType	316
5.49.4.3 ObjectType	317
5.49.4.4 TableType	317
5.50 schema.h	318
Index	325

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

AggreCondition	7
BasicType	8
TypeCharN	216
TypeDate	219
TypeDateTime	222
TypeFloat32	225
TypeFloat64	228
TypeInt16	231
TypeInt32	234
TypeInt64	237
TypeInt8	240
TypeTime	243
bnode	12
Catalog	13
Condition	24
Conditions	24
ErrorLog	29
Executor	37
GrAggRecord	50
GroupbyAggre::group_by_hash	51
GroupbyAggre::group_by_key	52
HashCell	78
HashCode_Ptr	79
HashInfo	90
HashTable	97
Key	130
Memory	132
MStorage	136
Object	139
Column	20
Database	25
Index	103
HashIndex	80
PbtreeIndex	156
Table	202

RowTable	179
Operator	142
Filter	40
GroupbyAggre	53
IndexScan	120
Join	124
HashJoin	92
IndexJoin	116
Orderby	145
Project	166
Scan	197
pbtrees	??
Pbtrees	152
PbtreesInfo	162
Pointer8B	164
RequestColumn	173
RequestTable	174
ResultTable	175
RPattern	193
SelectQuery	200

Chapter 2

Class Index

2.1 Class List

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

AggreCondition	7
BasicType	8
bnode	12
Catalog	13
Column	20
Condition	24
Conditions	24
Database	25
ErrorLog	
Array of source file names	29
Executor	37
Filter	40
GrAggRecord	50
GroupbyAggre::group_by_hash	51
GroupbyAggre::group_by_key	52
GroupbyAggre	53
HashCell	78
HashCode_Ptr	79
HashIndex	80
HashInfo	90
HashJoin	92
HashTable	97
Index	103
IndexJoin	116
IndexScan	120
Join	124
Key	130
Memory	132
MStorage	136
Object	139
Operator	142
Orderby	145
pbtrees	??
Pbtrees	152
PbtreesIndex	156

PbtreeInfo	162
Pointer8B	164
Project	166
RequestColumn	173
RequestTable	174
ResultTable	175
RowTable	179
RPattern	193
Scan	197
SelectQuery	200
Table	202
TypeCharN	216
TypeDate	219
TypeDateTime	222
TypeFloat32	225
TypeFloat64	228
TypeInt16	231
TypeInt32	234
TypeInt64	237
TypeInt8	240
TypeTime	243

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

system/catalog.cc	247
system/catalog.d	248
system/catalog.h	248
system/datatype.h	250
system/errorlog.cc	256
system/errorlog.d	258
system/errorlog.h	258
system/executor.cc	265
system/executor.d	267
system/executor.h	267
system/gcc_pf_p3.h	279
system/global.cc	281
system/global.d	282
system/global.h	282
system/hashindex.cc	284
system/hashindex.d	284
system/hashindex.h	284
system/hashtable.cc	286
system/hashtable.d	287
system/hashtable.h	287
system/mymemory.cc	289
system/mymemory.d	290
system/mymemory.h	290
system/nodepref.h	292
system/pbtree.cc	297
system/pbtree.d	298
system/pbtree.h	298
system/pbtreeindex.cc	305
system/pbtreeindex.d	305
system/pbtreeindex.h	305
system/rowtable.cc	307
system/rowtable.d	307
system/rowtable.h	307
system/runaimdb.cc	311
system/runaimdb.d	314
system/schema.h	314

Chapter 4

Class Documentation

4.1 AggreCondition Struct Reference

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

Public Attributes

- int [column_rank](#)
- [AggregateMethod](#) [method](#)

4.1.1 Detailed Description

definition of aggregete condition.

4.1.2 Member Data Documentation

4.1.2.1 column_rank

```
int AggreCondition::column_rank
```

the rank of the column has aggregate method

4.1.2.2 method

```
AggregateMethod AggreCondition::method
```

the aggregate method on that column

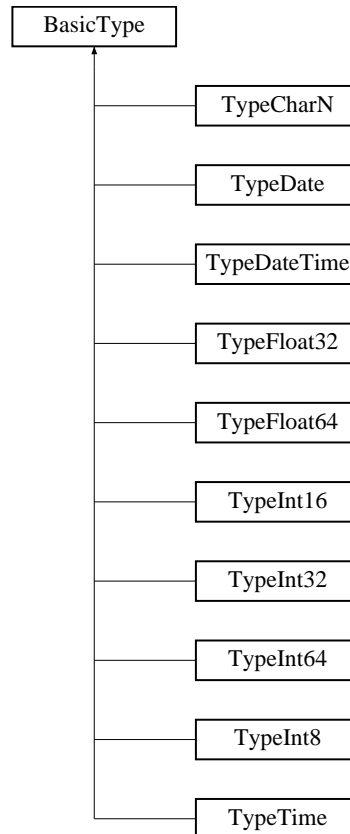
The documentation for this struct was generated from the following file:

- system/[executor.h](#)

4.2 BasicType Class Reference

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

Inheritance diagram for BasicType:



Public Member Functions

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

Protected Attributes

- [TypeCode](#) [b_type_code](#)
- int64_t [b_type_size](#)

4.2.1 Detailed Description

definition of class [BasicType](#).

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BasicType()

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

constructor.

4.2.2.2 ~BasicType()

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

destructor.

4.2.3 Member Function Documentation

4.2.3.1 cmpEQ()

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

equal to.

Reimplemented in [TypeInt8](#), [TypeInt16](#), [TypeInt32](#), [TypeInt64](#), [TypeFloat32](#), [TypeFloat64](#), [TypeCharN](#), [TypeDate](#), [TypeTime](#), and [TypeDateTime](#).

4.2.3.2 cmpGE()

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

greater than or equal to

Reimplemented in [TypeInt8](#), [TypeInt16](#), [TypeInt32](#), [TypeInt64](#), [TypeFloat32](#), [TypeFloat64](#), [TypeCharN](#), [TypeDate](#), [TypeTime](#), and [TypeDateTime](#).

4.2.3.3 cmpGT()

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

greater than.

Reimplemented in [TypeInt8](#), [TypeInt16](#), [TypeInt32](#), [TypeInt64](#), [TypeFloat32](#), [TypeFloat64](#), [TypeCharN](#), [TypeDate](#), [TypeTime](#), and [TypeDateTime](#).

4.2.3.4 cmpLE()

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

less than or equal to.

Reimplemented in [TypeInt8](#), [TypeInt16](#), [TypeInt32](#), [TypeInt64](#), [TypeFloat32](#), [TypeFloat64](#), [TypeCharN](#), [TypeDate](#), [TypeTime](#), and [TypeDateTime](#).

4.2.3.5 cmpLT()

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

less than.

Reimplemented in [TypeInt8](#), [TypeInt16](#), [TypeInt32](#), [TypeInt64](#), [TypeFloat32](#), [TypeFloat64](#), [TypeCharN](#), [TypeDate](#), [TypeTime](#), and [TypeDateTime](#).

4.2.3.6 copy()

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

copy from data to dest.

Reimplemented in [TypeInt8](#), [TypeInt16](#), [TypeInt32](#), [TypeInt64](#), [TypeFloat32](#), [TypeFloat64](#), [TypeCharN](#), [TypeDate](#), [TypeTime](#), and [TypeDateTime](#).

4.2.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented in [TypeInt8](#), [TypeInt16](#), [TypeInt32](#), [TypeInt64](#), [TypeFloat32](#), [TypeFloat64](#), [TypeCharN](#), [TypeDate](#), [TypeTime](#), and [TypeDateTime](#).

4.2.3.8 formatTxt()

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

extract txt format from data(bin) to dest.

Reimplemented in [TypeInt8](#), [TypeInt16](#), [TypeInt32](#), [TypeInt64](#), [TypeFloat32](#), [TypeFloat64](#), [TypeCharN](#), [TypeDate](#), [TypeTime](#), and [TypeDateTime](#).

4.2.3.9 getTypeCode()

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

get type code of this data type.

4.2.3.10 getTypeSize()

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

get data size when stored in bin format.

4.2.4 Member Data Documentation

4.2.4.1 b_type_code

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

data type code

4.2.4.2 b_type_size

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

data type size

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

- [system/datatype.h](#)

4.3 bnode Class Reference

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

Public Member Functions

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

Public Attributes

- [Pointer8B](#) [child](#) [[BKEY_NUM](#)+1]
- [key_type](#) [key](#) [[BKEY_NUM](#)+1]

4.3.1 Member Function Documentation

4.3.1.1 ch()

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

4.3.1.2 chEndAddr()

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


4.3.1.3 k()

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

4.3.2 Member Data Documentation

4.3.2.1 child

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

4.3.2.2 key

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

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

- [system/pbtree.h](#)

4.4 Catalog Class Reference

```
#include <catalog.h>
```

Public Member Functions

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

Private Member Functions

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

Private Attributes

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

4.4.1 Detailed Description

definition of class [Catalog](#).

4.4.2 Member Function Documentation

4.4.2.1 `createColumn()`

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

create column.

Parameters

<i>name</i>	column name
<i>type</i>	column type: [INT8,INT16,...]
<i>option_size</i>	only work for column type CHARN(option_size)
<i>c_id</i>	reference of column identifier, position in cl_id_obj

Return values

<i>true</i>	success
<i>false</i>	failure

4.4.2.2 `createDatabase()`

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

create database.

Parameters

<i>name</i>	database name
<i>d_id</i>	reference of database identifier, position in cl_id_obj

Return values

<i>true</i>	success
<i>false</i>	failure

4.4.2.3 createIndex()

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

create index.

Parameters

<i>name</i>	index name
<i>type</i>	index type: [HASHINDEX,BPTREEINDEX,ARTTREEINDEX]
<i>i_key</i>	stores column identifiers of this index
<i>i_id</i>	reference of index identifier, position in cl_id_obj

Return values

<i>true</i>	success
<i>false</i>	failure

4.4.2.4 createTable()

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

create table.

Parameters

<i>name</i>	table name
<i>type</i>	tabletype: [ROWTABLE,COLUMNTABLE]
<i>t_id</i>	reference of table identifier, position in cl_id_obj

Return values

<i>true</i>	success
-------------	---------

Return values

<i>false</i>	failure
--------------	---------

4.4.2.5 getObjById()

```
Object * Catalog::getObjById (
    int64_t o_id )
```

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

Parameters

<i>o</i> ↔ <i>_id</i>	identifier of object
--------------------------	----------------------

Return values

<i>!=</i>	NULL available
<i>==</i>	NULL unavaliable, deleted or not exist

4.4.2.6 getObjByName()

```
Object * Catalog::getObjByName (
    char * o_name )
```

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

Parameters

<i>name</i>	of an object
-------------	--------------

Return values

<i>!=</i>	NULL available
<i>==</i>	NULL unavaliable, deleted or not exist

4.4.2.7 init()

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

init operation.

4.4.2.8 initColumn()

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

init column

Parameters

$c \leftrightarrow$ _id	which column to prepare
----------------------------	-------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

4.4.2.9 initDatabase()

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

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

Parameters

$d \leftrightarrow$ _id	which database to prepare
----------------------------	---------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

4.4.2.10 initIndex()

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

init index

Parameters

$i \leftarrow$ _id	which index to prepare
$t \leftarrow$ _id	index in which table

Return values

<i>true</i>	success
<i>false</i>	failure

4.4.2.11 initTable()

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

init table

Parameters

$t \leftarrow$ _id	which table to prepare
-----------------------	------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

4.4.2.12 obtainId()

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

get a free object identifier

Return values

>0	success
≤ 0	failure

4.4.2.13 print()

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

print the catalog

4.4.2.14 registerObj()

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

put object in cl_id_obj and cl_name_obj

Parameters

<i>obj</i>	pointer of object to put
------------	--------------------------

Return values

>0	success
≤ 0	failure

4.4.2.15 shut()

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

shut down, free all memory of Objects.

4.4.2.16 shutDatabase()

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

shutdownm database

Parameters

d_{\leftarrow} <i>_id</i>	which database to shut
--------------------------------	------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

4.4.3 Member Data Documentation

4.4.3.1 cl_id_obj

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

container of Objects

4.4.3.2 cl_name_obj

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

name map of Objects ,name should not be the same in the whole [Catalog](#)

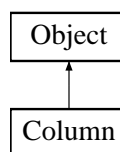
The documentation for this class was generated from the following files:

- [system/catalog.h](#)
- [system/catalog.cc](#)

4.5 Column Class Reference

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

Inheritance diagram for Column:



Public Member Functions

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

Private Attributes

- [BasicType](#) * [c_datatype](#)
- [int64_t](#) [c_offset](#)
- [int64_t](#) [c_size](#)
- [ColumnType](#) [c_type](#)

4.5.1 Detailed Description

definition of class [Column](#).

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Column()

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

constructor.

Parameters

<i>c_id</i>	column identifier
<i>c_name</i>	column name
<i>c_type</i>	column type
<i>c_size</i>	data type size

4.5.2.2 ~Column()

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

destructor.

4.5.3 Member Function Documentation

4.5.3.1 finish()

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

finish column setting.

4.5.3.2 getCoffset()

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

get column offset.

4.5.3.3 getCSize()

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

get data dize.

4.5.3.4 getCType()

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

get column type.

4.5.3.5 getDataType()

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

get data type of the column

4.5.3.6 init()

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

init column.

4.5.3.7 print()

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

print column information

Reimplemented from [Object](#).

4.5.3.8 setCoffset()

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

get column offset.

4.5.3.9 shut()

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

shut down column.

Reimplemented from [Object](#).

4.5.4 Member Data Documentation

4.5.4.1 c_datatype

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

column data type

4.5.4.2 c_offset

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

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

4.5.4.3 c_size

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

column size

4.5.4.4 c_type

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

column type

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

- [system/schema.h](#)

4.6 Condition Struct Reference

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

Public Attributes

- [RequestColumn](#) column
- [CompareMethod](#) compare
- char [value](#) [128]

4.6.1 Detailed Description

definition of compare condition.

4.6.2 Member Data Documentation

4.6.2.1 column

[RequestColumn](#) Condition::column

which column

4.6.2.2 compare

[CompareMethod](#) Condition::compare

which method

4.6.2.3 value

```
char Condition::value[128]
```

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

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

- system/[executor.h](#)

4.7 Conditions Struct Reference

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

Public Attributes

- [Condition condition](#) [4]
- int [condition_num](#)

4.7.1 Detailed Description

definition of conditions.

4.7.2 Member Data Documentation

4.7.2.1 condition

[Condition](#) Conditions::condition[4]

support maximum 4 & conditions

4.7.2.2 condition_num

int Conditions::condition_num

number of condition in use

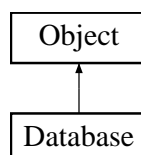
The documentation for this struct was generated from the following file:

- system/[executor.h](#)

4.8 Database Class Reference

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

Inheritance diagram for Database:



Public Member Functions

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

Private Attributes

- std::vector< int64_t > [d_table](#)

4.8.1 Detailed Description

definition of class [Database](#).

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Database()

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

constructor.

Parameters

<i>d_id</i>	database identifier
<i>d_name</i>	database name

4.8.2.2 ~Database()

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

destructor.

4.8.3 Member Function Documentation

4.8.3.1 addTable()

```
virtual bool Database::addTable (
    int64_t table_id ) [inline], [virtual]
```

add table identifier to this database.

Parameters

<i>table_id</i>	table identifier
-----------------	------------------

Return values

<i>true</i>	success
<i>false</i>	failure

4.8.3.2 finish()

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

finish, important interface for son class

4.8.3.3 getTables()

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

get table identifier container.

4.8.3.4 init()

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

init, important interface for son class

4.8.3.5 insert() [1/2]

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

insert a record to this database's table.

Parameters

<i>table_id</i>	table identifier
<i>columns</i>	each element of columns is a pointer to data of the column

Return values

<i>true</i>	success
<i>false</i>	failure

4.8.3.6 insert() [2/2]

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

insert a record to this database's table.

Parameters

<i>table_id</i>	table identifier
<i>source</i>	buffer of data to insert

Return values

<i>true</i>	success
<i>false</i>	failure

4.8.3.7 loadData()

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

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

Parameters

<i>table_id</i>	table identifier
<i>filename</i>	data file to load

Return values

<i>true</i>	success
<i>false</i>	failure

4.8.3.8 print()

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

print database information

Reimplemented from [Object](#).

4.8.3.9 shut()

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

shut down this database, free all memory.

Reimplemented from [Object](#).

4.8.4 Member Data Documentation

4.8.4.1 d_table

```
std::vector< int64_t > Database::d_table [private]
```

table identifier container.

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

- system/[schema.h](#)

4.9 ErrorLog Class Reference

an array of source file names

```
#include <errorlog.h>
```

Public Member Functions

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

Static Public Member Functions

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

Static Public Attributes

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

Private Member Functions

- int [getFuncNameGCC](#) (char *bt_symbol)

Private Attributes

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

Static Private Attributes

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

4.9.1 Detailed Description

an array of source file names

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ErrorLog()

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

constructor

Parameters

<i>threadid</i>	the current thread id to generate the log for
<i>level</i>	the dynamic logging level
<i>logfile</i>	if not NULL then specify the log file path

4.9.2.2 ~ErrorLog()

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

destructor

4.9.3 Member Function Documentation

4.9.3.1 closeLog()

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

close the log file

4.9.3.2 flushLog()

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

flush the log file

4.9.3.3 getErrorCode()

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

return the last error code

4.9.3.4 getErrorMsg()

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

return the error message accumulated since last [reset\(\)](#)

4.9.3.5 getFuncNameGCC()

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

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

A symbol has the following format:

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

Parameters

<i>bt_symbol</i>	a symbol returned from <code>backtrace_symbols()</code>
------------------	---

Return values

0	success, output is in <code>el_demangle_buf</code>
---	--

Return values

-1	function name is not available
----	--------------------------------

4.9.3.6 id2Name()

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

get the file name for a given id

Parameters

<i>src_id</i>	the file id
---------------	-------------

Return values

<i>!=NULL</i>	the file name
<i>==NULL</i>	the file id is out of range

4.9.3.7 init()

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

global initiator

Parameters

<i>level</i>	the dynamic logging level
<i>logfile</i>	if not NULL then specify the log file path

4.9.3.8 log()

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

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

Parameters

<i>level</i>	the level of the message
<i>src_name</i>	the source file name
<i>lineno</i>	the line number in the source file
...	printf-like format string and arguments

4.9.3.9 name2Id()

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

get fileid for a given file name

Parameters

<i>src_name</i>	the file name
-----------------	---------------

Return values

≥ 0	the file id
< 0	the file name does not exist

4.9.3.10 reset()

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

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

4.9.3.11 setLevel()

```
void ErrorLog::setLevel (
    int level ) [static]
```

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

Parameters

<i>level</i>	the dynamic logging level
--------------	---------------------------

4.9.4 Member Data Documentation

4.9.4.1 el_bt_buffer

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

allow up to 256 levels of calls

4.9.4.2 el_demangle_buf

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

buffer needed for demangle

4.9.4.3 el_demangle_len

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

demangle buffer length

4.9.4.4 el_err_code

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

current error code

4.9.4.5 el_fp

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

file handle of el_logfile

4.9.4.6 el_level

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

logging level

4.9.4.7 el_level_name

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

level => name

4.9.4.8 el_lock

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

protect the global states

4.9.4.9 el_logfile

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

file path to write log to

4.9.4.10 el_msg_buf

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

a buffer to hold the error message

4.9.4.11 el_msg_cap

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

the message buffer size

4.9.4.12 el_msg_cur

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

point to the end of the message

4.9.4.13 el_name_2_id

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

file name => id

4.9.4.14 el_thread_name

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

the thread name

4.9.4.15 el_tloc

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

local time in seconds at last message

4.9.4.16 el_tm

```
struct tm ErrorLog::el_tm [private]
```

broken down time at last message

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

- [system/errorlog.h](#)
- [system/errorlog.cc](#)

4.10 Executor Class Reference

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

Public Member Functions

- int [close](#) ()
- int [exec](#) ([SelectQuery](#) *query, [ResultTable](#) *result)
- int [findCol](#) (char *[table_name](#), char *[column_name](#))
find the rank of selected column in given table
- int64_t [getRank](#) (std::vector< int64_t > &vec, int64_t id)
- [Operator](#) * [planner](#) ([SelectQuery](#) *query)

Public Attributes

- [Operator](#) * [root](#)

Private Attributes

- [SelectQuery](#) * [current_query](#)

4.10.1 Detailed Description

definition of class executor.

4.10.2 Member Function Documentation

4.10.2.1 [close\(\)](#)

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

close function.

Parameters

<i>None</i>	
-------------	--

Return values

<i>==0</i>	succeed to close
<i>!=0</i>	fail to close

4.10.2.2 [exec\(\)](#)

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

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

exec function.

Parameters

<i>query</i>	to execute, if NULL, execute query at last time
<i>result</i>	table generated by an execution, store result in pattern defined by the result table

Return values

>0	number of result rows stored in result
≤ 0	no more result

4.10.2.3 findCol()

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

find the rank of selected column in given table

Parameters

<i>table_name</i>	the name of the given table
<i>column_name</i>	the name of the selected column

Returns

the rank of the column in table

4.10.2.4 getRank()

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

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

Parameters

<i>vec</i>	the id vector to be searched through
<i>id</i>	the id to be searched

4.10.2.5 planner()

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

generate operator tree and return root

Parameters

<i>query</i>	the query to form an operator tree
--------------	------------------------------------

Return values

<i>the</i>	root of the operator tree
------------	---------------------------

4.10.3 Member Data Documentation

4.10.3.1 current_query

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

selectquery to iterately execute

4.10.3.2 root

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

the root of the operator tree

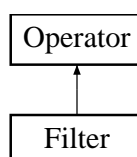
The documentation for this class was generated from the following files:

- [system/executor.h](#)
- [system/executor.cc](#)

4.11 Filter Class Reference

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

Inheritance diagram for Filter:



Public Member Functions

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

Private Member Functions

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

Static Private Member Functions

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

Private Attributes

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

Additional Inherited Members

4.11.1 Detailed Description

definition of the class [Filter](#) the filter operator

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Filter() [1/3]

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

Constructor

4.11.2.2 ~Filter()

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

Destructor

4.11.2.3 Filter() [2/3]

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

Overload constructor, designate some properties when created operator

Parameters

<i>child</i>	child operator to set
<i>input_cid</i>	ID of input columns
<i>filt_rank</i>	rank of filtering column from input
<i>cmp_mtd</i>	compare method
<i>value</i>	text of value compared with

4.11.2.4 Filter() [3/3]

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

Overload constructor

Parameters

<i>child</i>	child operator to set
<i>c_id</i>	ID of input columns
<i>num_column</i>	number of input columns
<i>filt_rank</i>	rank of filtering column from input
<i>cmp_mtd</i>	compare method
<i>value</i>	text of value compared with

4.11.3 Member Function Documentation

4.11.3.1 close()

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

close this operator, release all resources

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.11.3.2 cmpEQ()

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

Equal to

Parameters

<i>data1</i>	first operand
<i>data2</i>	second operand
<i>data_type</i>	basic type of these two operands

Return values

<i>true</i>	<code>data1 == data2</code>
<i>false</i>	otherwise

4.11.3.3 cmpGE()

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

Greater than or equal to

Parameters

<i>data1</i>	first operand
<i>data2</i>	second operand
<i>data_type</i>	basic type of these two operands

Return values

<i>true</i>	$data1 \geq data2$
<i>false</i>	otherwise

4.11.3.4 cmpGT()

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

Greater than

Parameters

<i>data1</i>	first operand
<i>data2</i>	second operand
<i>data_type</i>	basic type of these two operands

Return values

<i>true</i>	$data1 > data2$
<i>false</i>	otherwise

4.11.3.5 cmpLE()

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



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

Less than or equal to

Parameters

<i>data1</i>	first operand
<i>data2</i>	second operand
<i>data_type</i>	basic type of these two operands

Return values

<i>true</i>	$\text{data1} \leq \text{data2}$
<i>false</i>	otherwise

4.11.3.6 cmpLT()

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

Comparison function for this class (wrapped method of [BasicType](#)) Less than

Parameters

<i>data1</i>	first operand
<i>data2</i>	second operand
<i>data_type</i>	basic type of these two operands

Return values

<i>true</i>	$\text{data1} < \text{data2}$
<i>false</i>	otherwise

4.11.3.7 cmpNE()

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

Not equal to

Parameters

<i>data1</i>	first operand
<i>data2</i>	second operand
<i>data_type</i>	basic type of these two operands

Return values

<i>true</i>	<code>data1 != data2</code>
<i>false</i>	otherwise

4.11.3.8 getNext()

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

get next tuple iterately

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.11.3.9 initCmpFunc()

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

Initialize compare function

4.11.3.10 open()

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

open a [Filter](#) operator

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.11.3.11 setChild()

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

Set child operator, should be called before open

Parameters

<i>child</i>	child operator
--------------	----------------

4.11.3.12 setColumn() [1/2]

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

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

Parameters

<i>c_id</i>	the column id of input
<i>num_column</i>	number of columns in the input
<i>filt_rank</i>	the filter column is which column from the input
<i>cmp_mtd</i>	compare method
<i>value</i>	text of designated value

4.11.3.13 setColumn() [2/2]

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

Overload, set input_cid

Parameters

<i>input_cid</i>	the column id of input
<i>filt_rank</i>	the filter column is which column from the input
<i>cmp_mtd</i>	compare method
<i>value</i>	text of designated value

Return values

<i>true</i>	success
<i>false</i>	failure

4.11.3.14 setFiltCond()

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

Set properties for filter

Parameters

<i>filt_rank</i>	the filter column is which column from the input
<i>cmp_mtd</i>	compare method
<i>value</i>	text value to compare with

4.11.4 Member Data Documentation**4.11.4.1 buf_for_child**

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

child operator

4.11.4.2 child

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

4.11.4.3 child_buf_size

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

buffer allocated for child

4.11.4.4 cmp_func

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

datatype of column to filter on

4.11.4.5 cmp_mtd

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

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

4.11.4.6 cmp_table

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

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

4.11.4.7 filt_off

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

position of column to filter on

4.11.4.8 filt_pos

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

identifiers of columns to operate

4.11.4.9 filt_type

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

store compare method

4.11.4.10 in_tuple_size

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

compare function for filter

4.11.4.11 input_cid

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

size of buffer for child

4.11.4.12 value

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

size of the input tuple

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

- [system/executor.h](#)
- [system/executor.cc](#)

4.12 GrAggRecord Class Reference

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

Public Member Functions

- [GrAggRecord](#) (char *[middle_record](#), int64_t num_aggr)

Public Attributes

- std::vector< int64_t > [count](#)
- char * [middle_record](#)
- std::vector< int64_t > [sum](#)

4.12.1 Detailed Description

Definition of class [GrAggRecord](#) an intermediate record of [GroupbyAggre](#)

4.12.2 Constructor & Destructor Documentation

4.12.2.1 GrAggRecord()

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

Count of this class, use int64 to store Constructor

Parameters

<i>middle_record</i>	initial middle_record
<i>num_aggr</i>	number of aggr, size of sum/count

4.12.3 Member Data Documentation

4.12.3.1 count

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

Sum of this class, use int64 to store

4.12.3.2 middle_record

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

4.12.3.3 sum

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

Points to a class of middle record

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

- [system/executor.h](#)

4.13 GroupbyAggre::group_by_hash Struct Reference

Public Member Functions

- [size_t operator\(\)](#) (const [group_by_key_t](#) &key) const

4.13.1 Member Function Documentation

4.13.1.1 operator>()

```
size_t GroupbyAggre::group_by_hash::operator() (
    const group\_by\_key\_t & key ) const [inline]
```

A buffer used to compare a new record with existing record

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

- [system/executor.h](#)

4.14 GroupbyAggre::group_by_key Struct Reference

Public Member Functions

- bool [operator==](#) (const [group_by_key](#) &k) const

Public Attributes

- [group_by_type_t](#) [type_array](#)
- std::vector< char * > [value_array](#)

4.14.1 Detailed Description

Position of aggr columns

4.14.2 Member Function Documentation

4.14.2.1 operator==()

```
bool GroupbyAggre::group_by_key::operator== (
    const group\_by\_key & k ) const    [inline]
```

4.14.3 Member Data Documentation

4.14.3.1 type_array

[group_by_type_t](#) GroupbyAggre::group_by_key::type_array

4.14.3.2 value_array

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

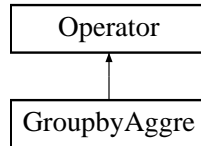
The documentation for this struct was generated from the following file:

- [system/executor.h](#)

4.15 GroupbyAggre Class Reference

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

Inheritance diagram for GroupbyAggre:



Classes

- struct [group_by_hash](#)
- struct [group_by_key](#)

Public Member Functions

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

Private Types

- typedef struct [GroupbyAggre::group_by_hash](#) [group_by_hash_t](#)
- typedef struct [GroupbyAggre::group_by_key](#) [group_by_key_t](#)
- typedef std::vector< [BasicType](#) * > [group_by_type_t](#)

Static Private Member Functions

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

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

Private Attributes

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

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

Additional Inherited Members

4.15.1 Detailed Description

Definition of class [GroupbyAggre](#) the groupby-aggregation operator

4.15.2 Member Typedef Documentation

4.15.2.1 [group_by_hash_t](#)

```
typedef struct GroupbyAggre::group\_by\_hash GroupbyAggre::group\_by\_hash\_t [private]
```

4.15.2.2 [group_by_key_t](#)

```
typedef struct GroupbyAggre::group\_by\_key GroupbyAggre::group\_by\_key\_t [private]
```

Position of aggr columns

4.15.2.3 [group_by_type_t](#)

```
typedef std::vector< BasicType * > GroupbyAggre::group\_by\_type\_t [private]
```

Each element points to an intermediate record

4.15.3 Constructor & Destructor Documentation

4.15.3.1 GroupbyAggre()

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

Finalize method table Constructor Initialize some data structure

4.15.3.2 ~GroupbyAggre()

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

Destructor

4.15.4 Member Function Documentation

4.15.4.1 avgFloat32()

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

Aggregate AVG method for FLOAT32

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.2 avgFloat64()

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

Aggregate AVG method for FLOAT64

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.3 avgInt16()

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

Aggregate AVG method for INT16

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.4 avgInt32()

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

Aggregate AVG method for INT32

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.5 avgInt64()

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

Aggregate AVG method for INT64

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.6 avgInt8()

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

Aggregate AVG method for INT8

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.7 close()

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

close this operator, release all resources

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.15.4.8 count()

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

Aggregate COUNT method

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to original value of sum

4.15.4.9 finalCount()

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

Finalize for COUNT method

Parameters

<i>sum</i>	sum to be finalized
<i>count</i>	count to be finalized
<i>result</i>	place to save result

4.15.4.10 finalFloat32Avg()

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

Finalize for AVG method of FLOAT32

Parameters

<i>sum</i>	sum to be finalized
<i>count</i>	count to be finalized
<i>result</i>	place to save result

4.15.4.11 finalFloat32Sum()

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

Finalize for SUM method of FLOAT32

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.12 finalFloat64Avg()

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

Finalize for AVG method of FLOAT64

Parameters

<i>sum</i>	sum to be finalized
<i>count</i>	count to be finalized
<i>result</i>	place to save result

4.15.4.13 finalFloat64Sum()

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

Finalize for SUM method of FLOAT64

Parameters

<i>sum</i>	sum to be finalized
<i>count</i>	count to be finalized
<i>result</i>	place to save result

4.15.4.14 finalInt16Sum()

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

Finalize for SUM method of INT16

Parameters

<i>sum</i>	sum to be finalized
<i>count</i>	count to be finalized
<i>result</i>	place to save result

4.15.4.15 finalInt32Sum()

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

Finalize for SUM method of INT32

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.16 finalInt64Sum()

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

Finalize for SUM method of INT64

Parameters

<i>sum</i>	sum to be finalized
<i>count</i>	count to be finalized
<i>result</i>	place to save result

4.15.4.17 finalInt8Sum()

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

Init method table Finalize for SUM method of INT8

Parameters

<i>sum</i>	sum to be finalized
<i>count</i>	count to be finalized
<i>result</i>	place to save result

4.15.4.18 finalIntAvg()

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

Finalize for AVG method of INT

Parameters

<i>sum</i>	sum to be finalized
<i>count</i>	count to be finalized
<i>result</i>	place to save result

4.15.4.19 getNext()

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

get next record from the table iterately

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.15.4.20 initAvg()

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

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

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.21 initCount()

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

Init for count method

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.22 initFloat32Max()

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

Init for MAX method of FLOAT32

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.23 initFloat32Min()

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

Init for MIN method of FLOAT32

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.24 initFloat64Max()

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

Init for MAX method of FLOAT64

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.25 initFloat64Min()

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

Init for MIN method of FLOAT64

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.26 initInt16Max()

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

Init for MAX method of INT16

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.27 initInt16Min()

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

Init for MIN method of INT16

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.28 initInt32Max()

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

Init for MAX method of INT32

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.29 initInt32Min()

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

Init for MIN method of INT32

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.30 initInt64Max()

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

Init for MAX method of INT64

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.31 initInt64Min()

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

Init for MIN method of INT64

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.32 initInt8Max()

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

Init for MAX method of INT8

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.33 initInt8Min()

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

Init for MIN method of INT8

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.34 initSum()

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

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

Parameters

<i>sum</i>	sum to be initialized
<i>count</i>	count to be initialized

4.15.4.35 maxFloat32()

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

Aggregate MAX method for FLOAT32

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.36 maxFloat64()

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

Aggregate MAX method for FLOAT64

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.37 maxInt16()

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

Aggregate MAX method for INT16

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.38 maxInt32()

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

Aggregate MAX method for INT32

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.39 maxInt64()

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

Aggregate MAX method for INT64

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.40 maxInt8()

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

Aggregate MAX method for INT8

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.41 minFloat32()

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

Aggregate MIN method for FLOAT32

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.42 minFloat64()

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

Aggregate MIN method for FLOAT64

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.43 minInt16()

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

Aggregate MIN method for INT16

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.44 minInt32()

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

Aggregate MIN method for INT32

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.45 minInt64()

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

Aggregate MIN method for INT64

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.46 minInt8()

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

Aggregate MIN method for INT8

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.47 open()

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

open a scan operator

Return values

<i>true</i>	successfully opened
<i>false</i>	failed

Reimplemented from [Operator](#).

4.15.4.48 set()

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

Set input and output columns

Parameters

<i>input_colid</i>	ID of input columns
<i>groupby_rank</i>	rank of group-by columns from input
<i>conditions</i>	aggregate method with its column
<i>output_colid</i>	ID of output columns

4.15.4.49 setChild()

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

Set child operator, should be called before open

Parameters

<i>child</i>	child operator
--------------	----------------

4.15.4.50 sumFloat32()

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

Aggregate SUM method for FLOAT32

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.51 sumFloat64()

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

Aggregate SUM method for FLOAT64

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.52 sumInt16()

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

Aggregate SUM method for INT16

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.53 sumInt32()

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

Aggregate SUM method for INT32

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.54 sumInt64()

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

Aggregate SUM method for INT64

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.4.55 sumInt8()

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

Iterator for next record Aggregate SUM method for INT8

Parameters

<i>sum</i>	pointed to original value of sum
<i>count</i>	pointed to original value of count
<i>x</i>	pointed to x to be added

4.15.5 Member Data Documentation

4.15.5.1 aggr_method

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

4.15.5.2 aggr_pos

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

Basic type of aggr columns

4.15.5.3 aggr_type

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

Size of group-by columns

4.15.5.4 avg_table

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

Search table for AVG

4.15.5.5 buf_for_child

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

Aggregate method with column

4.15.5.6 child

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

4.15.5.7 child_buf_size

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

Buffer for child

4.15.5.8 child_tuple_size

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

Buffer size allocated for child

4.15.5.9 conditions

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

Output column ID

4.15.5.10 final_avg_table

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

Search table for AVG final

4.15.5.11 final_method

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

4.15.5.12 final_sum_table

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

Search table for SUM final

4.15.5.13 group_by_pos

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

Basic type of group-by columns

4.15.5.14 group_by_size

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

Position of group-by columns

4.15.5.15 group_by_type

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

4.15.5.16 groupby_rank

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

Input column ID

4.15.5.17 hash_group

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

4.15.5.18 in_cid

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

Child operator

4.15.5.19 init_max_table

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

Search table for MAX init

4.15.5.20 init_method

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

4.15.5.21 init_min_table

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

Search table for MIN init

4.15.5.22 max_table

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

Search table for MAX

4.15.5.23 middle_buf_array

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

Size of buffer allocated for intermediate record

4.15.5.24 middle_buf_size

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

Size of an intermediate record, not including c and s

4.15.5.25 middle_tuple_size

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

Size of a tuple from child

4.15.5.26 min_table

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

Search table for MIN

4.15.5.27 next_iter

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

Hash map for groupby

4.15.5.28 out_cid

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

Rank of group-by columns

4.15.5.29 sum_table

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

Search table for SUM

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

- [system/executor.h](#)
- [system/executor.cc](#)

4.16 HashCell Class Reference

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

Public Attributes

- int [hc_num](#)
- union {
 - [Hashcode_Ptr](#) ent
 - struct {
 - int [capacity](#)
 - [Hashcode_Ptr](#) * [ents](#)
 - } [num_2_or_more](#)
- } [hc_union](#)

4.16.1 Detailed Description

definition of class [HashCell](#).

4.16.2 Member Data Documentation

4.16.2.1 capacity

```
int HashCell::capacity
```

maximun number of array in this structure

4.16.2.2 ent

`Hashcode_Ptr HashCell::ent`

`Hashcode_Ptr`, hit, decrease cache miss

4.16.2.3 ents

`Hashcode_Ptr* HashCell::ents`

pointer of `Hashcode_Ptr` array

4.16.2.4 hc_num

`int HashCell::hc_num`

`Hashcode_Ptr` number in this `HashCell`

4.16.2.5

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

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

4.16.2.6

`struct { ... } HashCell::num_2_or_more`

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

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

- `system/hashtable.h`

4.17 Hashcode_Ptr Class Reference

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

Public Attributes

- `int64_t hash_code`
- `char * tuple`

4.17.1 Detailed Description

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

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

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

4.17.2 Member Data Documentation

4.17.2.1 hash_code

```
int64_t Hashcode_Ptr::hash_code
```

hash_code of specific data

4.17.2.2 tuple

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

pointer of a record tuple

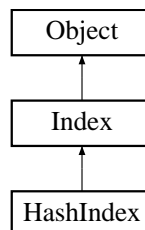
The documentation for this class was generated from the following file:

- system/[hashtable.h](#)

4.18 HashIndex Class Reference

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

Inheritance diagram for HashIndex:



Public Member Functions

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

Private Member Functions

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

Private Attributes

- int64_t [ih_cell_capbits](#)
- int64_t [ih_column_cap](#)
- int64_t [ih_column_num](#)
- [BasicType](#) ** [ih_datatype](#)
- int64_t * [ih_hash_bits](#)
- [HashTable](#) * [ih_hashtable](#)
- int64_t * [ih_table_offset](#)

Additional Inherited Members

4.18.1 Detailed Description

definition of [HashIndex](#).

4.18.2 Constructor & Destructor Documentation

4.18.2.1 HashIndex()

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

constructor.

Parameters

<i>h_id</i>	hash index identifier
<i>i_name</i>	index name
<i>i_key</i>	key of this index

4.18.3 Member Function Documentation

4.18.3.1 addIndexDTpye()

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

add indexed column's data type.

Parameters

<i>i_dt</i>	data type of indexed column
<i>offset</i>	offset of column in a rowtable

4.18.3.2 cmpEQ() [1/2]

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

whether *i_data* is result got from hash table

Parameters

<i>i_data</i>	buffer data of columns
<i>result</i>	got from hash table

Return values

<i>true</i>	equal
<i>false</i>	not equal

4.18.3.3 cmpEQ() [2/2]

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

whether *i_data* is result got from hash table

Parameters

<i>i_data</i>	pointers of columns
<i>result</i>	got from hash table

Return values

<i>true</i>	equal
<i>false</i>	not equal

4.18.3.4 del() [1/2]

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

del an entry in hash index.

Parameters

<i>i_data</i>	buffer of column data
---------------	-----------------------

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.18.3.5 del() [2/2]

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

del an entry in hash index.

Parameters

<i>i_data</i>	pointers of column data
---------------	-------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.18.3.6 finish()

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

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

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.18.3.7 hash()

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

hash method for CHARN.

Parameters

<i>str</i>	pointer of string
<i>maxlen</i>	maximum length of str

Return values

<i>int64</i>	hash value of CHARN
--------------	---------------------

4.18.3.8 init()

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

init hashindex, to calculate initial value.

Return values

<i>true</i>	init success
-------------	--------------

Reimplemented from [Index](#).

4.18.3.9 insert() [1/2]

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

insert an entry to hash index.

Parameters

<i>i_data</i>	buffer of column data in patten
<i>p_in</i>	pointer of record to make index

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.18.3.10 insert() [2/2]

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

insert an entry to hash index.

Parameters

<i>i_data</i>	each element of i_data pointed to a column data
<i>p_in</i>	pointer of record to make index

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.18.3.11 `lookup()` [1/2]

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

lookup hash index.

Parameters

<i>i_data</i>	buffer of column data
<i>info</i>	HashInfo pointer processed by <code>set_ls</code>
<i>result</i>	reference of record pointer

Return values

<i>true</i>	found
<i>false</i>	not found

Reimplemented from [Index](#).

4.18.3.12 `lookup()` [2/2]

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

lookup hash index.

Parameters

<i>i_data</i>	pointers of column data
<i>info</i>	HashInfo pointer processed by <code>set_ls</code>
<i>result</i>	reference of record pointer

Return values

<i>true</i>	found
<i>false</i>	not found

Reimplemented from [Index](#).

4.18.3.13 print()

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

print hash index information.

Reimplemented from [Index](#).

4.18.3.14 set_ls() [1/2]

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

setup for hash index lookup.

Parameters

<i>i_data1</i>	buffer of column data for lookup or scan(">=")
<i>i_data2</i>	set NULL
<i>info</i>	HashInfo pointer

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.18.3.15 set_ls() [2/2]

```
bool HashIndex::set_ls (
    void * i_data1[],
```

```
void * i_data2[],  
void * info ) [virtual]
```

setup for hash index lookup.

Parameters

<i>i_data1</i>	pointers of column data for lookup
<i>i_data2</i>	set NULL when call
<i>info</i>	HashInfo pointer

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.18.3.16 setCellCap()

```
void HashIndex::setCellCap (  
    int64_t cell_capbits ) [inline]
```

set hashtable cell capacity.

Parameters

<i>cell_capbits</i>	the number of cells in hashtable is $2^{\text{cell_capbits}}$
---------------------	--

4.18.3.17 shut()

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

free memory of hash table and other strucure.

Return values

<i>true</i>	success
-------------	---------

Reimplemented from [Index](#).

4.18.3.18 tranToInt64() [1/2]

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

assemble hash keys, INT and CHARN.

Parameters

<i>i_data</i>	buffer of column data
---------------	-----------------------

Return values

<i>int64</i>	hash index code
--------------	-----------------

Reimplemented from [Index](#).

4.18.3.19 tranToInt64() [2/2]

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

assemble hash keys, INT and CHARN.

Parameters

<i>i_data</i>	pointers of column data
---------------	-------------------------

Return values

<i>int64</i>	hash index code
--------------	-----------------

Reimplemented from [Index](#).

4.18.4 Member Data Documentation**4.18.4.1 ih_cell_capbits**

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

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

4.18.4.2 ih_column_cap

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

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

4.18.4.3 ih_column_num

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

current number of added columns

4.18.4.4 ih_datatype

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

each column data type

4.18.4.5 ih_hash_bits

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

each column assign bits when hashing

4.18.4.6 ih_hashtable

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

main part hash table

4.18.4.7 ih_table_offset

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

offsets of column key in rowtable

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

- [system/hashindex.h](#)
- [system/hashindex.cc](#)

4.19 HashInfo Struct Reference

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

Public Attributes

- `int64_t` [hash](#)
- `int` [last](#)
- `int` [ppos](#)
- `char *` [result](#) [[HASHINFO_CAPICITY](#)]
- `int` [rnum](#)

4.19.1 Detailed Description

definition of [HashInfo](#).

4.19.2 Member Data Documentation

4.19.2.1 hash

```
int64_t HashInfo::hash
```

hashcell position in [HashTable](#)

4.19.2.2 last

```
int HashInfo::last
```

retval of [HashTable](#) lookup

4.19.2.3 ppos

```
int HashInfo::ppos
```

pair with last, |last|

4.19.2.4 result

```
char* HashInfo::result [HASHINFO\_CAPICITY]
```

buffer for value of void *pointer

4.19.2.5 rnum

```
int HashInfo::rnum
```

current result number

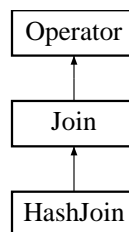
The documentation for this struct was generated from the following file:

- [system/hashindex.h](#)

4.20 HashJoin Class Reference

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

Inheritance diagram for HashJoin:



Public Member Functions

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

Private Attributes

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

Additional Inherited Members

4.20.1 Detailed Description

Definition of class [HashJoin](#) the simple [HashJoin](#) operator with arbitrary left and right

4.20.2 Constructor & Destructor Documentation

4.20.2.1 HashJoin() [1/2]

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

Upper bound of iterator for current key Constructor

4.20.2.2 ~HashJoin()

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

Destructor

4.20.2.3 HashJoin() [2/2]

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

Overload constructor, set some attributes if already known

Parameters

<i>left_cid</i>	column ID from left
<i>right_cid</i>	column ID from right
<i>left_rank</i>	rank of join key from left
<i>right_rank</i>	rank of join key from right

4.20.3 Member Function Documentation

4.20.3.1 close()

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

Close this operator

Return values

<i>true</i>	success
<i>false</i>	failure

Implements [Join](#).

4.20.3.2 getNext()

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

Generate next record

Return values

<i>true</i>	success
<i>false</i>	failure

Implements [Join](#).

4.20.3.3 open()

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

Open a [HashJoin](#) operator

Return values

<i>true</i>	success
<i>false</i>	failure

Implements [Join](#).

4.20.4 Member Data Documentation

4.20.4.1 hash_index

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

A buffer to receive text value

4.20.4.2 last_iter

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

Newly created hash index, will not be added to catalog

4.20.4.3 left_buf

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

4.20.4.4 left_key_off

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

Position for join key from the right

4.20.4.5 left_key_type

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

Basic type of join key on the right

4.20.4.6 left_tuple_size

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

Size of each middle buffer

4.20.4.7 middle_buf_array

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

Buffer allocated for right child

4.20.4.8 middle_buf_size

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

Each element points to an intermediate record

4.20.4.9 right_buf

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

Points to current area for left child to write

4.20.4.10 right_buf_size

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

Size of each record from right

4.20.4.11 right_has_next

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

Size of buffer allocated for right

4.20.4.12 right_key_pos

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

Whether read to the end of right input

4.20.4.13 right_key_type

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

Offset for join key from the left

4.20.4.14 right_tuple_size

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

Size of each record from left

4.20.4.15 txt_buf

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

Basic type of join key on the left

4.20.4.16 upper_iter

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

Last time iterator when searching from the left

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

- [system/executor.h](#)
- [system/executor.cc](#)

4.21 HashTable Class Reference

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

Public Member Functions

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

Public Attributes

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

Private Member Functions

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

Private Attributes

- std::unordered_map< void *, int > [pointer2size](#)

4.21.1 Detailed Description

definition of class [HashTable](#).

4.21.2 Constructor & Destructor Documentation

4.21.2.1 HashTable()

```
HashTable::HashTable (
    int estimatedNumDistinctKeys,
    double estimatedDupPerKey,
    int num_partitions = 0 )
```

constructor.

Parameters

<i>estimatedNumDistinctKeys</i>	estimated number of distinct keys,pre-knowledge for this HashTable usage
<i>estimatedDupPerKey</i>	estimated number of duplicate keys in average,pre-knowledge for this HashTable usage
<i>num_partitions</i>	leave it 0, unuseable

4.21.2.2 ~HashTable()

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

destructor, free [HashTable](#) memory to g_memory.

4.21.3 Member Function Documentation

4.21.3.1 add()

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

add an entry.

Parameters

<i>hashCode</i>	hash code of specified data
<i>tup</i>	pointer of a record tuple

Return values

<i>true</i>	success
<i>false</i>	failure

4.21.3.2 allocate()

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

mymemory alloc interface like malloc

4.21.3.3 del()

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

del an entry.

Parameters

<i>hashCode</i>	hash code of specified data
<i>tup</i>	pointer of a record tuple

Return values

<i>true</i>	success
<i>false</i>	failure

4.21.3.4 free()

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

mymemory free interface like free in stdlib

4.21.3.5 probe()

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

probe(lookup) entries with specified hashCode.

Parameters

<i>hashCode</i>	the specified hashCode to find
<i>match</i>	the buffer to store the result matched
<i>capacity</i>	the maximum number of tuple pointers in this buffer

Return values

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

4.21.3.6 probe_contd()

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

probe_contd(lookup) more entries with specified hashCode.

Parameters

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

Return values

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

4.21.3.7 show()

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


display data in this hash table, for debug use.

4.21.3.8 size_to_slot()

```
int HashTable::size_to_slot (
    int array_size ) [private]
```

find the offset in free_header array to get suitable free memory used in this [HashTable](#)

4.21.3.9 utilization()

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

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

4.21.4 Member Data Documentation

4.21.4.1 avail

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

pointer of next available [Hashcode_Ptr](#)

4.21.4.2 begin

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

start pointer of HashCells

4.21.4.3 end

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

the end pointer of [Hashcode_Ptr](#) in array

4.21.4.4 estimated_duplicates_per_key

```
double HashTable::estimated_duplicates_per_key
```

estimated number of duplicate keys in average,pre-knowledge for this [HashTable](#) usage

4.21.4.5 estimated_num_distinct_keys

```
int HashTable::estimated_num_distinct_keys
```

estimated number of distinct keys,pre-knowledge for this [HashTable](#) usage

4.21.4.6 free_header

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

free memory in the list with different number of [Hashcode_Ptr](#),link-list

4.21.4.7 initial_array_size

```
int HashTable::initial_array_size
```

when hc_num of [HashCell](#) exceeds 1 at the first time, number of [Hashcode_Ptr](#) allocated for [HashCell](#)

4.21.4.8 more_allocated

```
int HashTable::more_allocated
```

analysis of more memory allocated from g_memory

4.21.4.9 pointer2size

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

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

4.21.4.10 table

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

pointer of an array of [HashCell](#)

4.21.4.11 table_size

```
int HashTable::table_size
```

the number of HashCells in this table

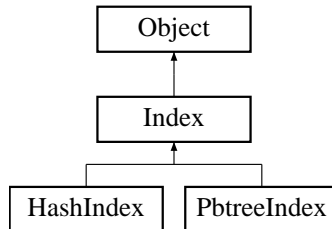
The documentation for this class was generated from the following files:

- system/[hashtable.h](#)
- system/[hashtable.cc](#)

4.22 Index Class Reference

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

Inheritance diagram for Index:



Public Member Functions

- [Index](#) (int64_t i_id, const char *i_name, [IndexType](#) i_type, [Key](#) &i_key)
- virtual [~Index](#) (void)
- virtual bool [del](#) (void *i_data)
- virtual bool [del](#) (void *i_data, void *p_del)
- virtual bool [del](#) (void *i_data[])
- virtual bool [del](#) (void *i_data[], void *p_del)
- virtual bool [finish](#) (void)
- [Key](#) & [getKey](#) (void)
- virtual int64_t [getIndexTid](#) (void)
- [IndexType](#) [getType](#) (void)
- virtual bool [init](#) (void)
- virtual bool [insert](#) (void *i_data, void *p_in)
- virtual bool [insert](#) (void *i_data[], void *p_in)
- virtual bool [lookup](#) (void *i_data, void *&result)
- virtual bool [lookup](#) (void *i_data, void *info, void *&result)
- virtual bool [lookup](#) (void *i_data[], void *&result)
- virtual bool [lookup](#) (void *i_data[], void *info, void *&result)
- virtual void [print](#) (void)
- virtual bool [scan](#) (void *info, void *&result)
- virtual bool [scan_1](#) (void *i_left, void *info)
- virtual bool [scan_1](#) (void *i_left[], void *info)
- virtual bool [scan_2](#) (void *i_right, void *info, void *&result)
- virtual bool [scan_2](#) (void *i_right[], void *info, void *&result)
- virtual bool [set_Is](#) (void *i_data1, void *i_data2, void *info)
- virtual bool [set_Is](#) (void *i_data1[], void *i_data2[], void *info)
- virtual void [setIndexTid](#) (int64_t tid)
- virtual bool [shut](#) (void)
- virtual int64_t [tranToInt64](#) (void *i_data)
- virtual int64_t [tranToInt64](#) (void *i_data[])
- virtual bool [update](#) (void *i_data, void *p_in)
- virtual bool [update](#) (void *i_data[], void *p_in)

Protected Attributes

- [Key](#) i_key
- int64_t i_t_id
- [IndexType](#) i_type

4.22.1 Detailed Description

definition of class [Index](#)

4.22.2 Constructor & Destructor Documentation

4.22.2.1 Index()

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

constructor.

Parameters

<i>i_id</i>	index identifier
<i>i_name</i>	index name
<i>i_type</i>	index type
<i>i_key</i>	key of this index

4.22.2.2 ~Index()

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

destructor.

4.22.3 Member Function Documentation

4.22.3.1 del() [1/4]

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

del an entry in [Index](#).

Parameters

<i>i_data</i>	char buffer to store data in pattern
---------------	--------------------------------------

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

Reimplemented in [HashIndex](#).

4.22.3.2 del() [2/4]

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

del an entry in [Index](#).

Parameters

<i>i_data</i>	char buffer to store data in pattern
<i>p_del</i>	address of specified row

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

Reimplemented in [PbtreeIndex](#).

4.22.3.3 del() [3/4]

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

del an entry in [BptreeIndex](#).

Parameters

<i>i_data</i>	each element of the array stores a pointer to column key
---------------	--

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

Reimplemented in [HashIndex](#).

4.22.3.4 del() [4/4]

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

del an entry in BptreeIndex.

Parameters

<i>i_data</i>	each element of the array stores a pointer to column key
<i>p_del</i>	address of specified row

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

4.22.3.5 finish()

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

finish index, important interface for son class

Reimplemented in [HashIndex](#).

4.22.3.6 getIKey()

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

get index key

4.22.3.7 getIndexTid()

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

get index in which table

4.22.3.8 getIType()

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

get index type

4.22.3.9 init()

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

init index, important interface for son class

Reimplemented in [HashIndex](#), and [PbtreeIndex](#).

4.22.3.10 insert() [1/2]

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

insert an entry to [Index](#).

Parameters

<i>i_data</i>	char buffer to store data in pattern
<i>p_in</i>	pointer of a row to make index

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

Reimplemented in [HashIndex](#), and [PbtreeIndex](#).

4.22.3.11 insert() [2/2]

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

insert an entry to [Index](#).

Parameters

<i>i_data</i>	each element of the array stores a pointer to column key
<i>p_in</i>	pointer of a row to make index

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

Reimplemented in [HashIndex](#).

4.22.3.12 lookup() [1/4]

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

lookup nonduplicate key in [Index](#).

Parameters

<i>i_data</i>	char buffer to store data in pattern
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

4.22.3.13 lookup() [2/4]

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

lookup duplicate key, iterate through the [Index](#).

Parameters

<i>i_data</i>	char buffer to store data in pattern
<i>info</i>	pointer of a BptreeInfo
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

Reimplemented in [HashIndex](#), and [PbtreeIndex](#).

4.22.3.14 lookup() [3/4]

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

lookup nonduplicate key in [Index](#).

Parameters

<i>i_data</i>	each element of the array stores a pointer to column key
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

4.22.3.15 lookup() [4/4]

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

lookup duplicate key,iterate through the [Index](#).

Parameters

<i>i_data</i>	each element of the array stores a pointer to column key
<i>info</i>	pointer of an index info
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	has more values
<i>false</i>	no more values

Reimplemented in [HashIndex](#).

4.22.3.16 print()

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

print index information

Reimplemented from [Object](#).

Reimplemented in [HashIndex](#), and [PbtreeIndex](#).

4.22.3.17 scan()

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

iterate on calling to scan for values.

Parameters

<i>info</i>	pointer of an index info
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	has more values
<i>false</i>	no more values

Reimplemented in [PbtreeIndex](#).

4.22.3.18 scan_1() [1/2]

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

prepare for scan operation.

Parameters

<i>i_left</i>	char buffer to store data in pattern, ">="
<i>info</i>	pointer of an index info

Return values

<i>true</i>	has more values
<i>false</i>	no more values

4.22.3.19 scan_1() [2/2]

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

prepare for scan operation.

Parameters

<i>i_left</i>	each element of the array stores a pointer to column key, ">="
<i>info</i>	pointer of an index info

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

4.22.3.20 scan_2() [1/2]

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

iterate on calling to scan for values.

Parameters

<i>i_right</i>	char buffer to store data in pattern, "<"
<i>info</i>	pointer of an index info
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	has more values
<i>false</i>	no more values

4.22.3.21 scan_2() [2/2]

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

iterate on calling to scan for values.

Parameters

<i>i_right</i>	each element of the array stores a pointer to column key, "<"
<i>info</i>	pointer of an index info
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	has more values
<i>false</i>	no more values

4.22.3.22 set_ls() [1/2]

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

prepare for lookup/scan operation.

Parameters

<i>i_data1</i>	char buffer to store data in pattern
<i>i_data2</i>	char buffer to store data in pattern, for lookup, <i>i_data2</i> =NULL
<i>info</i>	pointer of an index info

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

Reimplemented in [HashIndex](#), and [PbtreeIndex](#).

4.22.3.23 `set_ls()` [2/2]

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

prepare for lookup/scan operation.

Parameters

<i>i_data1</i>	each element of the array stores a pointer to column key
<i>i_data2</i>	each element of the array stores a pointer to column key, for lookup, i_data2=NULL
<i>info</i>	pointer of an index info

Return values

<i>true</i>	operation success
<i>false</i>	operation failure

Reimplemented in [HashIndex](#).

4.22.3.24 `setIndexTid()`

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

set index in which table

4.22.3.25 `shut()`

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

shut down the index, free memory.

Reimplemented from [Object](#).

Reimplemented in [HashIndex](#), and [PbtreeIndex](#).

4.22.3.26 `tranToInt64()` [1/2]

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

encode keys.

Parameters

<i>i_data</i>	char buffer to store data in pattern
---------------	--------------------------------------

Return values

<i>int64_t</i>	index code of <i>i_data</i>
----------------	-----------------------------

Reimplemented in [HashIndex](#).

4.22.3.27 tranToInt64() [2/2]

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

encode keys.

Parameters

<i>i_data</i>	each element of the array stores a pointer to column key
---------------	--

Return values

<i>int64_t</i>	index code of <i>i_data</i>
----------------	-----------------------------

Reimplemented in [HashIndex](#).

4.22.3.28 update() [1/2]

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

4.22.3.29 update() [2/2]

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

4.22.4 Member Data Documentation

4.22.4.1 i_key

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

index keys

4.22.4.2 i_t_id

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

in which table

4.22.4.3 i_type

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

index type

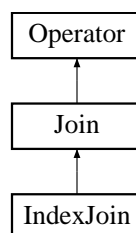
The documentation for this class was generated from the following file:

- `system/schema.h`

4.23 IndexJoin Class Reference

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

Inheritance diagram for IndexJoin:



Public Member Functions

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

Private Attributes

- void * [current_key](#)
- char * [left_buf](#)
- int64_t [left_buf_size](#)
- int64_t [left_tuple_size](#)
- char * [right_buf](#)
- int64_t [right_buf_size](#)
- bool [right_has_next](#)
- int64_t [right_tuple_size](#)

Additional Inherited Members

4.23.1 Detailed Description

Definition of the class [IndexJoin](#), the `IndexNestedLoopJoin` operator

4.23.2 Constructor & Destructor Documentation

4.23.2.1 `IndexJoin()` [1/2]

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

Whether read to the end of right input Constructor

4.23.2.2 `~IndexJoin()`

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

Destructor

4.23.2.3 `IndexJoin()` [2/2]

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

Overload constructor, set some attributes if already known

Parameters

<i>left_cid</i>	column ID from left
<i>right_cid</i>	column ID from left
<i>left_rank</i>	rank of join key from left
<i>right_rank</i>	rank of join key from right

4.23.3 Member Function Documentation

4.23.3.1 close()

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

Close this operator

Return values

<i>true</i>	success
<i>false</i>	failure

Implements [Join](#).

4.23.3.2 getNext()

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

Generate next record

Return values

<i>true</i>	success
<i>false</i>	failure

Implements [Join](#).

4.23.3.3 open()

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

Open a [IndexJoin](#) operator

Return values

<i>true</i>	success
<i>false</i>	failure

Implements [Join](#).

4.23.4 Member Data Documentation

4.23.4.1 `current_key`

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

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

4.23.4.2 `left_buf`

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

Size of each record from right

4.23.4.3 `left_buf_size`

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

Buffer for right child

4.23.4.4 `left_tuple_size`

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

4.23.4.5 `right_buf`

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

Buffer allocated for left child

4.23.4.6 `right_buf_size`

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

Left buffer size

4.23.4.7 `right_has_next`

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

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

4.23.4.8 right_tuple_size

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

Size of each record from left

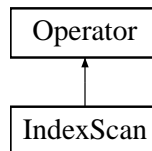
The documentation for this class was generated from the following files:

- [system/executor.h](#)
- [system/executor.cc](#)

4.24 IndexScan Class Reference

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

Inheritance diagram for IndexScan:



Public Member Functions

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

Private Attributes

- void * [current_key](#)
- [Table](#) * [from](#)
- [IndexType](#) [i_type](#)
- [Index](#) * [index](#)
- void * [info_ptr](#)
- bool [key_end](#)

Additional Inherited Members

4.24.1 Detailed Description

Definition of class [IndexScan](#)

4.24.2 Constructor & Destructor Documentation

4.24.2.1 IndexScan() [1/2]

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

Empty constructor, used together with `setTabIdx`

4.24.2.2 IndexScan() [2/2]

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

Constructor, set table and index at the same time

Parameters

<i>table</i>	table to scan
<i>index</i>	index to use when scanning

4.24.2.3 ~IndexScan()

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

Destructor

4.24.3 Member Function Documentation

4.24.3.1 close()

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

Close this operator

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.24.3.2 getNext()

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

Get next record from operator

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.24.3.3 open()

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

Open an [IndexScan](#) operator

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.24.3.4 setTabIdx()

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

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

Parameters

<i>table</i>	table to scan
<i>index</i>	index to use when scanning

4.24.3.5 updateKey()

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

Update search key for [IndexScan](#) Please use it before getNext

Parameters

<i>search_key</i>	new key to use
-------------------	----------------

4.24.4 Member Data Documentation

4.24.4.1 current_key

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

Pointer of info, general type

4.24.4.2 from

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

4.24.4.3 i_type

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

Use which index

4.24.4.4 index

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

From which table

4.24.4.5 info_ptr

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

Type of index

4.24.4.6 key_end

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

Current key for searching

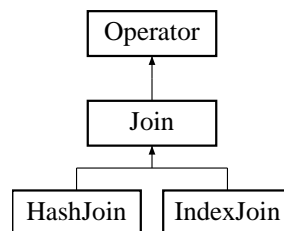
The documentation for this class was generated from the following files:

- [system/executor.h](#)
- [system/executor.cc](#)

4.25 Join Class Reference

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

Inheritance diagram for Join:



Public Member Functions

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

Private Attributes

- [Operator](#) * [left](#)
- std::vector< int64_t > [left_cid](#)
- int64_t [left_rank](#)
- [Operator](#) * [right](#)
- std::vector< int64_t > [right_cid](#)
- int64_t [right_rank](#)

Additional Inherited Members

4.25.1 Detailed Description

Definition of [Join](#) operator

4.25.2 Constructor & Destructor Documentation

4.25.2.1 Join()

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

[Join Key](#) is which column from the right child Constructor

4.25.2.2 ~Join()

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

Destructor

4.25.3 Member Function Documentation

4.25.3.1 close()

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

Close this operator

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

Implemented in [IndexJoin](#), and [HashJoin](#).

4.25.3.2 getLeftCol()

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

Get left cid

Return values

<i>left_cid</i>	
-----------------	--

4.25.3.3 getLeftOp()

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

Get left child operator

Return values

<i>reference</i>	of left child
------------------	---------------

4.25.3.4 getLeftRank()

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

Get left rank

Return values

<i>left_rank</i>	
------------------	--

4.25.3.5 getNext()

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

Generate next record

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

Implemented in [IndexJoin](#), and [HashJoin](#).

4.25.3.6 getRightCol()

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

Get right cid

Return values

<i>right_cid</i>	
------------------	--

4.25.3.7 getRightOp()

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

Get right child operator

Return values

<i>reference</i>	of right child
------------------	----------------

4.25.3.8 getRightRank()

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

Get right rank

Return values

<i>right_rank</i>	
-------------------	--

4.25.3.9 open()

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

Open a [Join](#) operator, [IndexJoin](#) or [HashJoin](#)

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

Implemented in [IndexJoin](#), and [HashJoin](#).

4.25.3.10 setJoinCol()

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

Set input column arrangement

Parameters

<i>left_cid</i>	column ID of left input
<i>right_cid</i>	column ID of right input
<i>left_rank</i>	left rank to set
<i>right_rank</i>	right rank to set

4.25.3.11 setLeftOp()

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

Set left child operator

Parameters

<i>lchild</i>	operator to set
---------------	-----------------

4.25.3.12 setRightOp()

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

Set right child operator

Parameters

<i>rchild</i>	operator to set
---------------	-----------------

4.25.4 Member Data Documentation

4.25.4.1 left

`Operator* Join::left [private]`

4.25.4.2 left_cid

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

Right child operator

4.25.4.3 left_rank

`int64_t Join::left_rank [private]`

Input column ID of right child

4.25.4.4 right

`Operator* Join::right [private]`

Left child operator, can only be scan

4.25.4.5 right_cid

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

Input column ID of left child

4.25.4.6 right_rank

`int64_t Join::right_rank [private]`

`Join Key` is which column from the left child

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

- [system/executor.h](#)

4.26 Key Class Reference

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

Public Member Functions

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

Private Attributes

- std::vector< int64_t > [key](#)

4.26.1 Detailed Description

definition of class [Key](#).

4.26.2 Constructor & Destructor Documentation

4.26.2.1 Key()

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

constructor.

4.26.3 Member Function Documentation

4.26.3.1 contain()

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

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

Return values

<i>true</i>	contain
<i>false</i>	don't contain

4.26.3.2 getKey()

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

get key data.

4.26.3.3 operator=()

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

over write operator(=).

4.26.3.4 print()

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

print key information.

4.26.3.5 set()

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

set key.

Parameters

<i>in_key</i>	keys(column identifiers) in vector
---------------	------------------------------------

4.26.4 Member Data Documentation

4.26.4.1 key

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

index identifier container

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

- system/[schema.h](#)

4.27 Memory Class Reference

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

Public Member Functions

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

Private Member Functions

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

Private Attributes

- char ** `m_array_list`
- char * `m_curr`
- char * `m_head`
- int64_t `m_mins`
- char * `m_table_addr`
- char * `m_tail`
- int64_t `m_total`

4.27.1 Member Function Documentation

4.27.1.1 alloc()

```
int64_t Memory::alloc (
    char *& p,
    int64_t size )
```

alloc db memory for inside usage.

Parameters

<i>p</i>	store the pointer result allocated from db memory
<i>size</i>	required size by caller, power of 2

Return values

<code>==size</code>	successfully allocated from db memory
<code><=0</code>	failure

4.27.1.2 `alloc_default()`

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

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

Parameters

<i>size</i>	required size by caller, power of 2
-------------	-------------------------------------

Return values

<code>==size</code>	successfully allocated from db memory
<code><=0</code>	failure

4.27.1.3 `allocTableAddr()`

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

alloc table address

Parameters

<i>p</i>	the pointer of memory to free
----------	-------------------------------

Return values

<code>0</code>	successfully free to db memory
----------------	--------------------------------

4.27.1.4 free()

```
int64_t Memory::free (
    char * p,
    int64_t size )
```

free memory to db memory.

Parameters

<i>p</i>	the pointer of memory to free
<i>size</i>	provided by caller, power of 2

Return values

<i>==size</i>	successfully free to db memory
<i><=0</i>	failure

4.27.1.5 init()

```
int Memory::init (
    int64_t total,
    int64_t mins )
```

init db memory.

Parameters

<i>total</i>	total size allocated from operate system, usually large enough
<i>mins</i>	minimux size db object allocated from db memory

Return values

<i>==0</i>	success
<i><0</i>	failure

4.27.1.6 print()

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

print memory usage.

4.27.1.7 shut()

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

free db memory to operate system.

4.27.1.8 slot()

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

calculate position of free list.

4.27.2 Member Data Documentation

4.27.2.1 m_array_list

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

free array list

4.27.2.2 m_curr

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

pointer of db memory already in use

4.27.2.3 m_head

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

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

4.27.2.4 m_mins

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

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

4.27.2.5 m_table_addr

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

current mmap table addr usage

4.27.2.6 m_tail

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

end pointer of db memory allocated from operate system

4.27.2.7 m_total

```
int64_t Memory::m_total [private]
```

total size of database system

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

- [system/mymemory.h](#)
- [system/mymemory.cc](#)

4.28 MStorage Class Reference

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

Public Member Functions

- [int64_t allocRow](#) (char *&pointer)
- [int64_t getRecordNum](#) (void)
- [char * getRow](#) (int64_t record_rank)
- [bool init](#) (int64_t record_size)
- [void shut](#) (void)

Private Member Functions

- [bool expand](#) (void)

Private Attributes

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

4.28.1 Detailed Description

definition of [MStorage](#), table storage manager.

4.28.2 Member Function Documentation

4.28.2.1 allocRow()

```
int64_t MStorage::allocRow (
    char *& pointer ) [inline]
```

alloc an empty row.

Parameters

<i>pointer</i>	reference of pointer result
----------------	-----------------------------

Return values

≥ 0	row rank in all table
< 0	failure

4.28.2.2 expand()

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

expand slots for more storage available for this table.

Return values

<i>true</i>	success
<i>false</i>	lack memory

4.28.2.3 getRecordNum()

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

get the last record rank till now.

4.28.2.4 getRow()

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

get the pointer of a row specified by *record_rank*.

Parameters

<i>record_rank</i>	the n th row in the table
--------------------	---------------------------

Return values

<i>!=NULL</i>	valid
<i>==NULL</i>	param error

4.28.2.5 init()

```
bool MStorage::init (
    int64_t record_size ) [inline]
```

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

Parameters

<i>record_size</i>	size of a row record
--------------------	----------------------

Return values

<i>true</i>	success
<i>false</i>	failure

4.28.2.6 shut()

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

shut down, free memory to system.

4.28.3 Member Data Documentation**4.28.3.1 ms_memory**

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

memory address

4.28.3.2 ms_memory_cur

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

current memory address

4.28.3.3 ms_memory_size

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

memory size actually mmaped

4.28.3.4 ms_record_max

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

max records with actual mmaped memory

4.28.3.5 ms_record_num

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

current record used

4.28.3.6 ms_record_size

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

size per record

4.28.3.7 pad

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

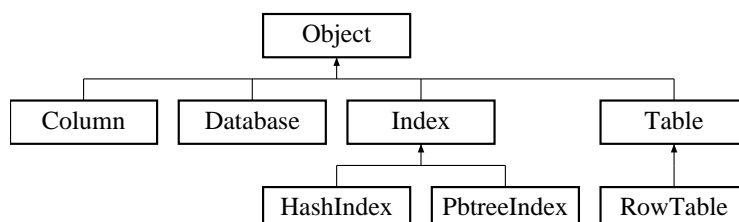
The documentation for this class was generated from the following file:

- [system/rowtable.h](#)

4.29 Object Class Reference

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

Inheritance diagram for Object:



Public Member Functions

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

Private Attributes

- int64_t [o_id](#)
- char [o_name](#) [[OBJ_NAME_MAX](#)]
- [ObjectType](#) [o_type](#)

4.29.1 Detailed Description

definition of [Object](#), basic element in database.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 Object()

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

constructor.

Parameters

<i>o_id</i>	object identifier
<i>o_type</i>	object type
<i>o_name</i>	object name

4.29.3 Member Function Documentation

4.29.3.1 changeName()

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


change object name(not in use).

4.29.3.2 getOid()

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

get identifier of object.

4.29.3.3 getOname()

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

get object name.

4.29.3.4 getOtype()

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

get object type.

4.29.3.5 print()

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

print the object infomation.

Reimplemented in [HashIndex](#), [PbtreeIndex](#), [Column](#), [Table](#), [Database](#), and [Index](#).

4.29.3.6 shut()

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

shut down the object.

Reimplemented in [HashIndex](#), [PbtreeIndex](#), [RowTable](#), [Column](#), [Table](#), [Database](#), and [Index](#).

4.29.4 Member Data Documentation

4.29.4.1 o_id

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

object identifier

4.29.4.2 o_name

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

object name, max length OBJ_NAME_MAX

4.29.4.3 o_type

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

object type

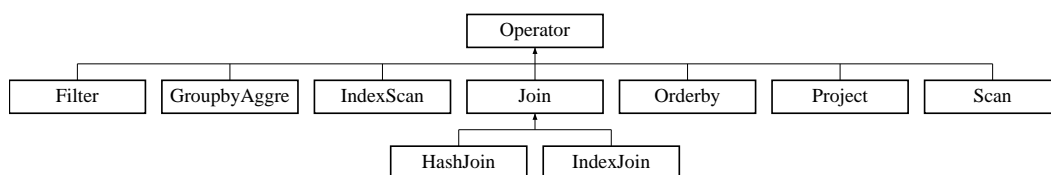
The documentation for this class was generated from the following file:

- system/[schema.h](#)

4.30 Operator Class Reference

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

Inheritance diagram for Operator:



Public Member Functions

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

Public Attributes

- char * [buffer_from_father](#)

4.30.1 Constructor & Destructor Documentation

4.30.1.1 Operator()

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

keep this buffer for passing record Constructor

4.30.1.2 ~Operator()

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

Destructor

4.30.2 Member Function Documentation

4.30.2.1 close()

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

close this operator, release all resources

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [Scan](#), [IndexScan](#), [Filter](#), [IndexJoin](#), [HashJoin](#), [Project](#), [GroupbyAggre](#), [Orderby](#), and [Join](#).

4.30.2.2 getBuffer()

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

get current operator's buffer to write

4.30.2.3 getNext()

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

get next record from the table iterately

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [Scan](#), [IndexScan](#), [Filter](#), [IndexJoin](#), [HashJoin](#), [Project](#), [GroupbyAggre](#), [Orderby](#), and [Join](#).

4.30.2.4 open()

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

open an operator

Return values

<i>true</i>	successfully opened
<i>false</i>	failed

Reimplemented in [Scan](#), [IndexScan](#), [Filter](#), [IndexJoin](#), [HashJoin](#), [Project](#), [GroupbyAggre](#), [Orderby](#), and [Join](#).

4.30.2.5 setBuffer()

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

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

Parameters

<i>buffer_allocated</i>	a static buffer to store the next record
-------------------------	--

4.30.3 Member Data Documentation

4.30.3.1 buffer_from_father

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

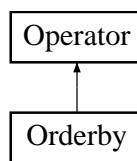
The documentation for this class was generated from the following file:

- [system/executor.h](#)

4.31 Orderby Class Reference

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

Inheritance diagram for Orderby:



Public Member Functions

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

Private Attributes

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

Additional Inherited Members

4.31.1 Constructor & Destructor Documentation

4.31.1.1 `OrderBy()`

```
OrderBy::OrderBy ( ) [inline]
```

Constructor

4.31.1.2 `~OrderBy()`

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

Destructor

4.31.2 Member Function Documentation

4.31.2.1 `close()`

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

close this operator, release all resources

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.31.2.2 `getNext()`

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

get next record from the table iterately

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.31.2.3 open()

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

open a [Orderby](#) operator

Return values

<i>true</i>	successfully opened
<i>false</i>	failed

Reimplemented from [Operator](#).

4.31.2.4 set()

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

set the variables of [Orderby](#)

Parameters

<i>input_colid</i>	the input column id
<i>orderby_rank</i>	the rank of the column needed to order by

4.31.2.5 setChild()

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

Set child operator, should be called before open

Parameters

<i>child</i>	child operator
--------------	----------------

4.31.3 Member Data Documentation

4.31.3.1 arrayid

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

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

4.31.3.2 child

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

child of this operator on the operator tree

4.31.3.3 child_buffer

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

the buffer for the child result

4.31.3.4 colid

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

the vector of input column id

4.31.3.5 coloff

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

the offset of column needed to order by

4.31.3.6 colrank

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

the rank of column needed to order by

4.31.3.7 coltype

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

the data type of column needed to order by

4.31.3.8 middle_buf_array

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

array of the middle result buffer

4.31.3.9 middle_buf_size

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

the size of the middle result buffer

4.31.3.10 orderby_num

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

the number of the column needed to order by

4.31.3.11 self_buf_size

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

the size of the self buffer

4.31.3.12 tuple_size

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

the size of one row from child result

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

- [system/executor.h](#)
- [system/executor.cc](#)

4.32 Pbtrees Class Reference

```
#include <pbtrees.h>
```

Public Member Functions

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

Private Attributes

- `char * p_free_header` [16]
- `pbtreet p_pbtreet`

4.32.1 Member Function Documentation

4.32.1.1 `allocate()`

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

4.32.1.2 `cap2leve()`

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

4.32.1.3 `del()`

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

4.32.1.4 `free()`

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

4.32.1.5 `get_recptr()`

```
int Pbtreet::get_recptr (
    void * p,
    void * match[],
    int capacity,
    int & pos )
```

4.32.1.6 init()

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

4.32.1.7 insert()

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

4.32.1.8 leve2cap()

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

4.32.1.9 leve2size()

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

4.32.1.10 lookup()

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

4.32.1.11 lookup_s()

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

4.32.1.12 print()

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

4.32.1.13 scan()

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

4.32.1.14 shut()

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

4.32.1.15 size2leve()

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

4.32.2 Member Data Documentation

4.32.2.1 p_free_header

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

4.32.2.2 p_pbtree

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

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

- system/[pbtree.h](#)
- system/[pbtree.cc](#)

4.33 Pbtree Class Reference

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

Public Member Functions

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

Private Attributes

- char * [p_free_header](#) [16]
- [pbtree](#) [p_pbtree](#)

4.33.1 Member Function Documentation

4.33.1.1 [allocate\(\)](#)

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

4.33.1.2 [cap2leve\(\)](#)

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

4.33.1.3 [del\(\)](#)

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

4.33.1.4 free()

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

4.33.1.5 get_recptr()

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

4.33.1.6 init()

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

4.33.1.7 insert()

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

4.33.1.8 leve2cap()

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

4.33.1.9 leve2size()

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

4.33.1.10 lookup()

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

4.33.1.11 lookup_s()

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

4.33.1.12 print()

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

4.33.1.13 scan()

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

4.33.1.14 shut()

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

4.33.1.15 size2leve()

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

4.33.2 Member Data Documentation

4.33.2.1 p_free_header

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

4.33.2.2 p_pbtree

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

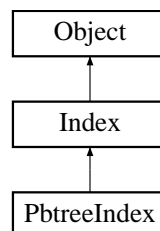
The documentation for this class was generated from the following files:

- [system/pbtree.h](#)
- [system/pbtree.cc](#)

4.34 PbtreeIndex Class Reference

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

Inheritance diagram for PbtreeIndex:



Public Member Functions

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

Private Attributes

- [BasicType](#) * [pi_datatype](#)
- [Pbtree](#) [pi_pbtree](#)

Additional Inherited Members

4.34.1 Constructor & Destructor Documentation

4.34.1.1 PbtreeIndex()

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

constructor.

Parameters

<i>pi_id</i>	pbtree index identifier
<i>i_name</i>	index name
<i>i_key</i>	key of this index

4.34.2 Member Function Documentation

4.34.2.1 del()

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

del an entry pbtree index.

Parameters

<i>i_data</i>	buffer of column data
<i>p_del</i>	pointer of row to delete

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.34.2.2 init()

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

init [PbtreeIndex](#)

Return values

<i>true</i>	successfully initied
-------------	----------------------

Reimplemented from [Index](#).

4.34.2.3 insert()

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

insert an entry to pbtree index.

Parameters

<i>i_data</i>	buffer of column data in pattren
<i>p_in</i>	pointer of record to make index

Return values

<i>true</i>	success
<i>false</i>	failure

insert an entry to hash index.

Parameters

<i>i_data</i>	buffer of column data in pattren
<i>p_in</i>	pointer of record to make index

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.34.2.4 lookup()

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

setup for pbtree index lookup.

Parameters

<i>i_data1</i>	pointers of column data for lookup
<i>i_data2</i>	set NULL when call
<i>info</i>	PbtreeInfo pointer

Return values

<i>true</i>	success
<i>false</i>	failure

lookup pbtree index.

Parameters

<i>i_data</i>	buffer of column data
<i>info</i>	PbtreeInfo pointer processed by set_ls
<i>result</i>	reference of record pointer

Return values

<i>true</i>	found
<i>false</i>	not found

Reimplemented from [Index](#).

4.34.2.5 print()

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

print [Pbtree](#) index information.

print pbtree index information.

Reimplemented from [Index](#).

4.34.2.6 scan()

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

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

Parameters

<i>info</i>	pointer of an index info
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	has more values
<i>false</i>	no more values

iterate on calling to scan for values.

Parameters

<i>info</i>	pointer of an index info
<i>result</i>	return the pointer to the indexed row

Return values

<i>true</i>	has more values
<i>false</i>	no more values

Reimplemented from [Index](#).

4.34.2.7 set_ls()

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

setup for pbtree index lookup.

Parameters

<i>i_data1</i>	buffer of column data for lookup or scan(">=")
<i>i_data2</i>	set NULL
<i>info</i>	PbtreeInfo pointer

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Index](#).

4.34.2.8 setIndexDTpye()

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

add indexed column's data type.

Parameters

<i>i_dt</i>	data type of indexed column
<i>offset</i>	offset of column in a rowtable

set indexed column's data type.

Parameters

<i>i_dt</i>	data type of indexed column
<i>offset</i>	offset of column in a rowtable

4.34.2.9 shut()

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

free memory of [Pbtree](#) and other strucure.

Return values

<i>true</i>	success
-------------	---------

free memory of hash table and other strucure.

Return values

<i>true</i>	success
-------------	---------

Reimplemented from [Index](#).

4.34.3 Member Data Documentation

4.34.3.1 pi_datatype

`BasicType* PbtreeIndex::pi_datatype [private]`

datatype of this column

4.34.3.2 pi_pbtree

`Pbtree PbtreeIndex::pi_pbtree [private]`

`Pbtree`, allow duplicated elements

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

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

4.35 PbtreeInfo Struct Reference

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

Public Attributes

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

4.35.1 Detailed Description

definition of `PbtreeInfo`.

4.35.2 Member Data Documentation

4.35.2.1 area

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

buffer for [Pbtree](#)

4.35.2.2 cr_area

```
int PbtreeInfo::cr_area
```

current area array pos in use

4.35.2.3 cr_resu

```
int PbtreeInfo::cr_resu
```

current result array pos in use

4.35.2.4 l_ptr

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

lookup elements ptr

4.35.2.5 le_resu

```
int PbtreeInfo::le_resu
```

len of current result

4.35.2.6 left

```
key\_type PbtreeInfo::left
```

lookup key or scan left edge

4.35.2.7 pos_resu

```
int PbtreeInfo::pos_resu
```

pos in match array, init 0

4.35.2.8 result

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

buffer for element

4.35.2.9 right

```
key_type PbtreeInfo::right
```

scan right edge

4.35.2.10 s_end

```
int PbtreeInfo::s_end
```

scan tag, scan has more? 0 means more

4.35.2.11 s_num

```
int PbtreeInfo::s_num
```

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

4.35.2.12 s_pos

```
int PbtreeInfo::s_pos
```

scan pos in bnode, acquired by lookup_s

4.35.2.13 s_ptr

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

scan bnode ptr, acquired by lookup_s

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

- [system/pbtreeindex.h](#)

4.36 Pointer8B Class Reference

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


Public Member Functions

- `operator char * ()`
- `operator struct bleaf * ()`
- `operator struct bnode * ()`
- `operator unsigned long long ()`
- `operator void * ()`
- `Pointer8B & operator= (const Pointer8B &p)`
- `Pointer8B & operator= (const void *ptr)`
- `void print (void)`

Public Attributes

- unsigned long long `value`

4.36.1 Member Function Documentation

4.36.1.1 `operator char *()`

```
Pointer8B::operator char * ( ) [inline]
```

4.36.1.2 `operator struct bleaf *()`

```
Pointer8B::operator struct bleaf * ( ) [inline]
```

4.36.1.3 `operator struct bnode *()`

```
Pointer8B::operator struct bnode * ( ) [inline]
```

4.36.1.4 `operator unsigned long long()`

```
Pointer8B::operator unsigned long long ( ) [inline]
```

4.36.1.5 operator void *()

```
Pointer8B::operator void * ( ) [inline]
```

4.36.1.6 operator=() [1/2]

```
Pointer8B & Pointer8B::operator= (
    const Pointer8B & p ) [inline]
```

4.36.1.7 operator=() [2/2]

```
Pointer8B & Pointer8B::operator= (
    const void * ptr ) [inline]
```

4.36.1.8 print()

```
void Pointer8B::print (
    void ) [inline]
```

4.36.2 Member Data Documentation

4.36.2.1 value

```
unsigned long long Pointer8B::value
```

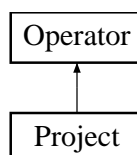
The documentation for this class was generated from the following file:

- [system/pbtree.h](#)

4.37 Project Class Reference

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

Inheritance diagram for Project:



Public Member Functions

- [Project](#) ()
- [Project](#) (std::vector< int64_t > in_cid, std::vector< int64_t > out_cid)
- [~Project](#) ()
- bool [close](#) ()
- int64_t [getColnum](#) ()
- bool [getNext](#) ()
- [BasicType](#) ** [getSchema](#) ()
- bool [open](#) ()
- void [setChild](#) ([Operator](#) *child)
- void [setProjCol](#) (std::vector< int64_t > in_cid, std::vector< int64_t > out_cid)
- bool [top](#) ()

Private Attributes

- char * [buf_for_child](#)
- [Operator](#) * [child](#)
- int64_t [in_buf_size](#)
- int64_t [in_tuple_size](#)
- std::vector< int64_t > [input_cid](#)
- std::vector< int64_t > [input_off](#)
- std::vector< char * > [input_pos](#)
- std::vector< [BasicType](#) * > [input_type](#)
- std::vector< int64_t > [out_to_in](#)
- std::vector< int64_t > [output_cid](#)
- char * [output_type](#)
- int64_t [output_type_buf_size](#)
- int64_t [output_type_size](#)
- int64_t [self_buf_size](#)
- bool [topid](#)

Additional Inherited Members

4.37.1 Detailed Description

Definition of class [Project](#) the projection operator

4.37.2 Constructor & Destructor Documentation

4.37.2.1 [Project\(\)](#) [1/2]

```
Project::Project ( ) [inline]
```

the indicator to show that if this is the top of the operator tree Constructor

4.37.2.2 ~Project()

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

Destructor

4.37.2.3 Project() [2/2]

```
Project::Project (
    std::vector< int64_t > in_cid,
    std::vector< int64_t > out_cid ) [inline]
```

Overload constructor, set some attributes if already known

Parameters

<i>in_cid</i>	input column ID
<i>out_cid</i>	output column ID

4.37.3 Member Function Documentation

4.37.3.1 close()

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

Close this operator

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.37.3.2 getColnum()

```
int64_t Project::getColnum ( ) [inline]
```

Get output column number

Return values

<i>size</i>	of output_cid
-------------	---------------

4.37.3.3 getNext()

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

Generate next record

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.37.3.4 getSchema()

```
BasicType ** Project::getSchema ( ) [inline]
```

Get output data type

Return values

<i>array</i>	of pointers to basic type of output columns
--------------	---

4.37.3.5 open()

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

Open a [Project](#) operator

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.37.3.6 setChild()

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

Set child operator

Parameters

<i>child</i>	child operator to set
--------------	-----------------------

4.37.3.7 setProjCol()

```
void Project::setProjCol (
    std::vector< int64_t > in_cid,
    std::vector< int64_t > out_cid ) [inline]
```

Set input and output columns

Parameters

<i>in_cid</i>	input column ID
<i>out_cid</i>	output column ID

4.37.3.8 top()

```
bool Project::top ( )
```

when this operator be the top of the operator tree, do this, set the buffer for self

Returns

true success
false failure

4.37.4 Member Data Documentation**4.37.4.1 buf_for_child**

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

child operator

4.37.4.2 child

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

4.37.4.3 in_buf_size

```
int64_t Project::in_buf_size [private]
```

input tuple size

4.37.4.4 in_tuple_size

```
int64_t Project::in_tuple_size [private]
```

an output column is which column from input

4.37.4.5 input_cid

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

buffer size allocated for operator on the top/self

4.37.4.6 input_off

```
std::vector< int64_t > Project::input_off [private]
```

basic type of each column from input

4.37.4.7 input_pos

```
std::vector< char * > Project::input_pos [private]
```

offset of each column from input

4.37.4.8 input_type

```
std::vector< BasicType * > Project::input_type [private]
```

input buffer size

4.37.4.9 out_to_in

```
std::vector< int64_t > Project::out_to_in [private]
```

output column ID, can be reordered

4.37.4.10 output_cid

```
std::vector< int64_t > Project::output_cid [private]
```

input column ID

4.37.4.11 output_type

```
char* Project::output_type [private]
```

postion of each column from input

4.37.4.12 output_type_buf_size

```
int64_t Project::output_type_buf_size [private]
```

size of output type array

4.37.4.13 output_type_size

```
int64_t Project::output_type_size [private]
```

basic type of each output column

4.37.4.14 self_buf_size

```
int64_t Project::self_buf_size [private]
```

buffer allocated for child

4.37.4.15 topid

```
bool Project::topid [private]
```

size of buffer allocated for *output_type

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

- [system/executor.h](#)
- [system/executor.cc](#)

4.38 RequestColumn Struct Reference

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

Public Attributes

- [AggregateMethod aggregate_method](#)
- char [name](#) [128]

4.38.1 Detailed Description

definition of request column.

4.38.2 Member Data Documentation

4.38.2.1 `aggregate_method`

`AggregateMethod` `RequestColumn::aggregate_method`

4.38.2.2 `name`

`char` `RequestColumn::name[128]`

name of column

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

- `system/executor.h`

4.39 RequestTable Struct Reference

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

Public Attributes

- `char` `name` [128]

4.39.1 Detailed Description

definition of request table.

4.39.2 Member Data Documentation

4.39.2.1 name

```
char RequestTable::name[128]
```

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

- [system/executor.h](#)

4.40 ResultTable Class Reference

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

Public Member Functions

- int [append](#) (char *src)
write a row into result table
- int [dump](#) (FILE *fp)
- char * [getRC](#) (int row, int column)
- int [init](#) (BasicType *col_types[], int col_num, int64_t capacity=1024)
- int [print](#) (void)
- int [shut](#) (void)
- int [writeRC](#) (int row, int column, void *data)

Public Attributes

- char * [buffer](#)
- int64_t [buffer_size](#)
- int [column_number](#)
- BasicType ** [column_type](#)
- int * [offset](#)
- int [offset_size](#)
- int [row_capacity](#)
- int [row_length](#)
- int [row_number](#)

4.40.1 Detailed Description

definition of result table.

4.40.2 Member Function Documentation

4.40.2.1 [append\(\)](#)

```
int ResultTable::append (  
    char * src )
```

write a row into result table

Parameters

<i>src</i>	the row buffer to be written
------------	------------------------------

Returns

1 success

0 error

4.40.2.2 dump()

```
int ResultTable::dump (
    FILE * fp )
```

write to file with FILE *fp

4.40.2.3 getRC()

```
char * ResultTable::getRC (
    int row,
    int column )
```

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

Parameters

<i>row</i>	row id in result table
<i>column</i>	column id in result table

Return values

<i>!=NULL</i>	pointer of a column
<i>==NULL</i>	error

4.40.2.4 init()

```
int ResultTable::init (
    BaseType * col_types[],
    int col_num,
    int64_t capacity = 1024 )
```

init alloc memory and set initial value

Parameters

<i>col_types</i>	array of column type pointers
<i>col_num</i>	number of columns in this ResultTable
<i>capacity</i>	buffer_size, power of 2

Return values

>0	success
≤ 0	failure

4.40.2.5 print()

```
int ResultTable::print (  
    void )
```

print result table, split by '\t', output a line per row

Return values

<i>the</i>	number of rows printed
------------	------------------------

4.40.2.6 shut()

```
int ResultTable::shut (  
    void )
```

free memory of this result table to g_memory

4.40.2.7 writeRC()

```
int ResultTable::writeRC (  
    int row,  
    int column,  
    void * data )
```

write data to position row,column

Parameters

<i>row</i>	row id in result table
<i>column</i>	column id in result table
<i>data</i>	data pointer of a column

Return values

<code>!=NULL</code>	pointer of a column
<code>==NULL</code>	error

4.40.3 Member Data Documentation

4.40.3.1 buffer

```
char* ResultTable::buffer
```

pointer of buffer allocated from g_memory

4.40.3.2 buffer_size

```
int64_t ResultTable::buffer_size
```

size of buffer, power of 2

4.40.3.3 column_number

```
int ResultTable::column_number
```

columns number that a result row consist of

4.40.3.4 column_type

```
BasicType** ResultTable::column_type
```

each column data type

4.40.3.5 offset

```
int* ResultTable::offset
```

4.40.3.6 offset_size

```
int ResultTable::offset_size
```

4.40.3.7 row_capacity

```
int ResultTable::row_capacity
```

maximum capacity of rows according to buffer size and length of row

4.40.3.8 row_length

```
int ResultTable::row_length
```

length per result row

4.40.3.9 row_number

```
int ResultTable::row_number
```

current usage of rows

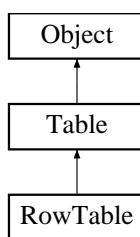
The documentation for this class was generated from the following files:

- [system/executor.h](#)
- [system/executor.cc](#)

4.41 RowTable Class Reference

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

Inheritance diagram for RowTable:



Public Member Functions

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

Private Member Functions

- bool [access](#) (int64_t record_rank, char *&pointer)
- bool [accessCol](#) (int64_t record_rank, int64_t column_rank, char *&pointer)
- bool [invalid](#) (char *record_ptr)
- bool [isValid](#) (char *record_ptr)

Private Attributes

- [RPattern](#) r_pattern
- [MStorage](#) r_storage

4.41.1 Detailed Description

definition of class [RowTable](#).

4.41.2 Constructor & Destructor Documentation

4.41.2.1 RowTable()

```
RowTable::RowTable (
    int64_t r_id,
    const char * r_name ) [inline]
```

constructor.

Parameters

<i>r_id</i>	table ideitifer
<i>r_name</i>	table name

4.41.3 Member Function Documentation

4.41.3.1 access()

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

get a row record pointer.

Parameters

<i>record_rank</i>	the n th record in the table
<i>pointer</i>	result pointer to return

Return values

<i>true</i>	success
<i>false</i>	failure

4.41.3.2 accessCol()

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

get a column of record pointer.

Parameters

<i>record_rank</i>	the n th record in the table
<i>column_rank</i>	the n th cilumn in the pattern
<i>pointer</i>	result pointer to return

Return values

<i>true</i>	success
-------------	---------

Return values

<i>false</i>	failure
--------------	---------

4.41.3.3 del() [1/2]

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

del a row(not in use).

Parameters

<i>row_pointer</i>	the pointer of a row
--------------------	----------------------

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.4 del() [2/2]

```
bool RowTable::del (
    int64_t record_rank ) [virtual]
```

del a row.

Parameters

<i>row_rank</i>	the n th record of the table
-----------------	------------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.5 finish()

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

finish, leave it empty.

Reimplemented from [Table](#).

4.41.3.6 getMStorage()

```
MStorage & RowTable::getMStorage (
    void ) [inline]
```

get storage of table.

4.41.3.7 getRecordNum()

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

get the last record rank.

Reimplemented from [Table](#).

4.41.3.8 getRecordPtr()

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

get row record pointer.

Parameters

<i>row_rank</i>	the n th record in the table
-----------------	------------------------------

Return values

<i>!=NULL</i>	success
<i>==NULL</i>	failure

Reimplemented from [Table](#).

4.41.3.9 getRPattern()

```
RPattern & RowTable::getRPattern (
    void ) [inline]
```

get pattern of table.

4.41.3.10 init()

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

init, leave it empty.

Reimplemented from [Table](#).

4.41.3.11 insert() [1/2]

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

insert a row.

Parameters

<i>columns</i>	each element of the array pointed to a column data
----------------	--

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.12 insert() [2/2]

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

insert a row.

Parameters

<i>source</i>	buffer of a row in pattern
---------------	----------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.13 invalid()

```
bool RowTable::invalid (
    char * record_ptr ) [inline], [private]
```

make the record invalid when del the record

Parameters

<i>record_ptr</i>	the pointer of a record
-------------------	-------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

4.41.3.14 isValid()

```
bool RowTable::isValid (
    char * record_ptr ) [inline], [private]
```

get result,whether the record is valid or not

Parameters

<i>record_ptr</i>	the pointer of a record
-------------------	-------------------------

Return values

<i>true</i>	valid
<i>false</i>	invalid

4.41.3.15 loadData()

```
bool RowTable::loadData (
```

```
const char * filename ) [virtual]
```

load data of the table(not in use).

Reimplemented from [Table](#).

4.41.3.16 printData()

```
bool RowTable::printData (
    void ) [virtual]
```

print table data, for debug.

Reimplemented from [Table](#).

4.41.3.17 select() [1/2]

```
bool RowTable::select (
    char * row_pointer,
    char * dest ) [virtual]
```

select all columns' data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.18 select() [2/2]

```
bool RowTable::select (
    int64_t record_rank,
    char * dest ) [virtual]
```

select all columns' data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.19 selectCol() [1/2]

```
bool RowTable::selectCol (
    char * row_pointer,
    int64_t column_rank,
    char * dest ) [virtual]
```

select one column data by pointer of a row.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_rank</i>	the n th column in table pattern
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.20 selectCol() [2/2]

```
bool RowTable::selectCol (
    int64_t record_rank,
    int64_t column_rank,
    char * dest ) [virtual]
```

select one column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_rank</i>	the n th column in table pattern
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.21 selectCols() [1/2]

```
bool RowTable::selectCols (
    char * row_pointer,
    int64_t column_total,
    int64_t * column_ranks,
    char * dest ) [virtual]
```

select several column data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.22 selectCols() [2/2]

```
bool RowTable::selectCols (
    int64_t record_rank,
    int64_t column_total,
    int64_t * column_ranks,
    char * dest ) [virtual]
```

select several column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.23 shut()

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

shut down r_pattern and r_storage, free their memory.

Reimplemented from [Table](#).

4.41.3.24 updateCol() [1/2]

```
bool RowTable::updateCol (
    char * row_pointer,
    int64_t column_rank,
    char * source ) [virtual]
```

update a column data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_rank</i>	the n th column in table pattern
<i>source</i>	buffer to store data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.25 updateCol() [2/2]

```
bool RowTable::updateCol (
    int64_t record_rank,
    int64_t column_rank,
    char * source ) [virtual]
```

update a column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_rank</i>	the n th column in table pattern
<i>source</i>	buffer to store data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.26 updateCols() [1/4]

```
bool RowTable::updateCols (
    char * row_pointer,
    int64_t column_total,
    int64_t * column_ranks,
    char * source ) [virtual]
```

update several column data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>source</i>	buffer to store data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.27 updateCols() [2/4]

```
bool RowTable::updateCols (
    char * row_pointer,
    int64_t column_total,
    int64_t * column_ranks,
    char * source[] ) [virtual]
```

update several column data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>source</i>	array of columns' pointers, each points a column data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.28 updateCols() [3/4]

```
bool RowTable::updateCols (
    int64_t record_rank,
    int64_t column_total,
    int64_t * column_ranks,
    char * source ) [virtual]
```

update several column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>source</i>	buffer to store data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.3.29 updateCols() [4/4]

```
bool RowTable::updateCols (
    int64_t record_rank,
    int64_t column_total,
    int64_t * column_ranks,
    char * source[] ) [virtual]
```

update several column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>source</i>	array of columns' pointers, each points a column data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Table](#).

4.41.4 Member Data Documentation

4.41.4.1 r_pattern

[RPattern](#) RowTable::r_pattern [private]

pattern of row

4.41.4.2 r_storage

[MStorage](#) RowTable::r_storage [private]

storage of table

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

- [system/rowtable.h](#)
- [system/rowtable.cc](#)

4.42 RPattern Class Reference

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

Public Member Functions

- bool [addColumn](#) ([BasicType](#) *col_type)
- int64_t [getColumnOffset](#) (int64_t col_rank)
- [BasicType](#) * [getColumnType](#) (int64_t col_rank)
- int64_t [getRowSize](#) (void)
- bool [init](#) (int64_t col_num)
- int64_t [print](#) (char *r_ptr)
- void [reset](#) (void)
- void [shut](#) (void)

Private Attributes

- char [par](#) [128 - 3 *sizeof(void *) - 4 *sizeof(int64_t)]
- int64_t [rp_colnum](#)
- int64_t [rp_current](#)
- [BasicType](#) ** [rp_dtype](#)
- int64_t [rp_mem_sz](#)
- char * [rp_memory](#)
- int64_t * [rp_offset](#)
- int64_t [rp_row_sz](#)

4.42.1 Detailed Description

definition of class [RPattern](#), describe row struture.

4.42.2 Member Function Documentation

4.42.2.1 addColumn()

```
bool RPattern::addColumn (
    BasicType * col_type ) [inline]
```

add column infomation.

Parameters

<i>col_type</i>	data type of the column
-----------------	-------------------------

4.42.2.2 getColumnOffset()

```
int64_t RPattern::getColumnOffset (
    int64_t col_rank ) [inline]
```

get offset of column in a row record.

Parameters

<i>col_rank</i>	the n th column in the table
-----------------	------------------------------

Return values

<i>>=0</i>	valid offset
<i>==-1</i>	input error

4.42.2.3 getColumnType()

```
BasicType * RPattern::getColumnType (
    int64_t col_rank ) [inline]
```

get data type of column.

Parameters

<i>col_rank</i>	the n th column in the table
-----------------	------------------------------

Return values

<i>!=</i>	NULL valid pointer
<i>==</i>	NULL input error

4.42.2.4 getRowSize()

```
int64_t RPattern::getRowSize (
    void ) [inline]
```

get size of a row record.

Return values

<i>the</i>	size of a row record
------------	----------------------

4.42.2.5 init()

```
bool RPattern::init (
    int64_t col_num ) [inline]
```

init, alloc memory and initial setting.

Parameters

<i>col_num</i>	number of columns, from class Table
----------------	---

Return values

<i>true</i>	success
<i>false</i>	failure

4.42.2.6 print()

```
int64_t RPattern::print (
    char * r_ptr ) [inline]
```

print a row following this pattern.

Parameters

<i>r_ptr</i>	pointer of a row
--------------	------------------

Return values

<i>==rp_row_size</i>	success
<i>!=rp_row_size</i>	error

4.42.2.7 reset()

```
void RPattern::reset (
    void ) [inline]
```

reset if addcolumn error happen.

4.42.2.8 shut()

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

shut down, free memory allocated from g_memory.

4.42.3 Member Data Documentation

4.42.3.1 par

```
char RPattern::par[128 - 3 * sizeof(void *) - 4 * sizeof(int64_t)] [private]
```

make the sizeof [RPattern](#) 128, managed by g_memory

4.42.3.2 rp_colnum

```
int64_t RPattern::rp_colnum [private]
```

total columns

4.42.3.3 rp_current

```
int64_t RPattern::rp_current [private]
```

current columns already set offset and datatype

4.42.3.4 rp_dtype

```
BasicType** RPattern::rp_dtype [private]
```

array of pointers to each column's data type

4.42.3.5 rp_mem_sz

```
int64_t RPattern::rp_mem_sz [private]
```

memory size

4.42.3.6 rp_memory

```
char* RPattern::rp_memory [private]
```

memory to store rp_offset and rp_datatype

4.42.3.7 rp_offset

```
int64_t* RPattern::rp_offset [private]
```

offset of column in a row

4.42.3.8 rp_row_sz

```
int64_t RPattern::rp_row_sz [private]
```

size of a row record

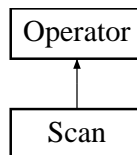
The documentation for this class was generated from the following file:

- [system/rowtable.h](#)

4.43 Scan Class Reference

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

Inheritance diagram for Scan:



Public Member Functions

- [Scan \(\)](#)
- [~Scan \(\)](#)
- [bool close \(\)](#)
- [bool getNext \(\)](#)
- [bool open \(\)](#)
- [void setTable \(Table *table\)](#)

Private Attributes

- [int64_t next_record](#)
- [Table * scan_table](#)
- [int64_t total_record](#)

Additional Inherited Members

4.43.1 Detailed Description

definition of the scan operator located at the bottom of the op-tree

4.43.2 Constructor & Destructor Documentation

4.43.2.1 Scan()

```
Scan::Scan ( ) [inline]
```

next record to check Constructor

4.43.2.2 ~Scan()

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

Destructor

4.43.3 Member Function Documentation

4.43.3.1 close()

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

close this operator, release all resources

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.43.3.2 getNext()

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

get next record from the table iterately

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented from [Operator](#).

4.43.3.3 open()

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

open a scan operator

Return values

<i>true</i>	successfully opened
<i>false</i>	failed

Note: the dest buffer should be allocated when first opened and keep static

Reimplemented from [Operator](#).

4.43.3.4 setTable()

```
void Scan::setTable (
    Table * table ) [inline]
```

Set the table to scan should be called before open

Parameters

<i>table</i>	the designated table
--------------	----------------------

4.43.4 Member Data Documentation

4.43.4.1 next_record

```
int64_t Scan::next_record [private]
```

total record num of this table when first opened

4.43.4.2 scan_table

```
Table* Scan::scan_table [private]
```

4.43.4.3 total_record

```
int64_t Scan::total_record [private]
```

the designated table

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

- [system/executor.h](#)
- [system/executor.cc](#)

4.44 SelectQuery Class Reference

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

Public Attributes

- [int64_t database_id](#)
- [int from_number](#)
- [RequestTable from_table](#) [4]
- [RequestColumn groupby](#) [4]
- [int groupby_number](#)
- [Conditions having](#)
- [RequestColumn orderby](#) [4]
- [int orderby_number](#)
- [RequestColumn select_column](#) [4]
- [int select_number](#)
- [Conditions where](#)

4.44.1 Detailed Description

definition of selectquery.

4.44.2 Member Data Documentation

4.44.2.1 database_id

```
int64_t SelectQuery::database_id
```

database to execute

4.44.2.2 from_number

```
int SelectQuery::from_number
```

number of tables to select from

4.44.2.3 from_table

```
RequestTable SelectQuery::from_table[4]
```

tables to select from, maximum 4

4.44.2.4 groupby

```
RequestColumn SelectQuery::groupby[4]
```

columns to groupby

4.44.2.5 groupby_number

```
int SelectQuery::groupby_number
```

number of columns to groupby

4.44.2.6 having

```
Conditions SelectQuery::having
```

groupby conditions

4.44.2.7 orderby

```
RequestColumn SelectQuery::orderby[4]
```

columns to orderby

4.44.2.8 orderby_number

```
int SelectQuery::orderby_number
```

number of columns to orderby

4.44.2.9 select_column

```
RequestColumn SelectQuery::select_column[4]
```

columns to select, maximum 4

4.44.2.10 select_number

```
int SelectQuery::select_number
```

number of column to select

4.44.2.11 where

```
Conditions SelectQuery::where
```

where meets conditions, maximum 4 & conditions

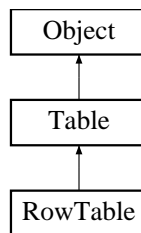
The documentation for this class was generated from the following file:

- [system/executor.h](#)

4.45 Table Class Reference

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

Inheritance diagram for Table:



Public Member Functions

- [Table](#) (int64_t t_id, const char *t_name, [TableType](#) t_type)
- virtual [~Table](#) (void)
- virtual bool [addColumn](#) (int64_t column_id)
- virtual bool [addIndex](#) (int64_t index_id)
- virtual bool [del](#) (char *columns[])
- virtual bool [del](#) (char *row_pointer)
- virtual bool [del](#) (int64_t record_rank)
- virtual bool [finish](#) (void)
- int64_t [getColumnRank](#) (int64_t c_id)
- std::vector< int64_t > & [getColumns](#) (void)
- int64_t [getIndexRank](#) (int64_t i_id)
- std::vector< int64_t > & [getIndexes](#) (void)
- int64_t [getRank](#) (std::vector< int64_t > &vec, int64_t id)
- virtual int64_t [getRecordNum](#) (void)
- virtual void * [getRecordPtr](#) (int64_t row_rank)
- [TableType](#) [getTtype](#) (void)
- virtual bool [init](#) (void)

- virtual bool [insert](#) (char *columns[])
- virtual bool [insert](#) (char *source)
- virtual bool [loadData](#) (const char *filename)
- virtual void [print](#) (void)
- virtual bool [printData](#) (void)
- virtual bool [select](#) (char *row_pointer, char *dest)
- virtual bool [select](#) (int64_t record_rank, char *dest)
- virtual bool [selectCol](#) (char *row_pointer, int64_t column_rank, char *dest)
- virtual bool [selectCol](#) (int64_t record_rank, int64_t column_rank, char *dest)
- virtual bool [selectCols](#) (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *dest)
- virtual bool [selectCols](#) (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *dest)
- virtual bool [shut](#) (void)
- virtual bool [updateCol](#) (char *row_pointer, int64_t column_rank, char *source)
- virtual bool [updateCol](#) (int64_t record_rank, int64_t column_rank, char *source)
- virtual bool [updateCols](#) (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *source)
- virtual bool [updateCols](#) (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *source[])
- virtual bool [updateCols](#) (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *source)
- virtual bool [updateCols](#) (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *source[])

Private Attributes

- std::vector< int64_t > [t_columns](#)
- std::vector< int64_t > [t_index](#)
- [TableType](#) [t_type](#)

4.45.1 Detailed Description

definition of class [Table](#).

4.45.2 Constructor & Destructor Documentation

4.45.2.1 ~Table()

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

destructor.

4.45.2.2 Table()

```
Table::Table (
    int64_t t_id,
    const char * t_name,
    TableType t_type ) [inline]
```

constructor.

Parameters

<i>t_id</i>	table identifier
<i>t_name</i>	table name
<i>t_type</i>	table type

4.45.3 Member Function Documentation

4.45.3.1 addColumn()

```
virtual bool Table::addColumn (
    int64_t column_id ) [inline], [virtual]
```

add column identifier to this table.

4.45.3.2 addIndex()

```
virtual bool Table::addIndex (
    int64_t index_id ) [inline], [virtual]
```

add index identifier to this table.

4.45.3.3 del() [1/3]

```
virtual bool Table::del (
    char * columns[] ) [inline], [virtual]
```

del a row(not in use).

Parameters

<i>columns</i>	array of the pointers in a row
----------------	--------------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

4.45.3.4 del() [2/3]

```
virtual bool Table::del (
    char * row_pointer ) [inline], [virtual]
```


del a row(not in use).

Parameters

<i>row_pointer</i>	the pointer of a row
--------------------	----------------------

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.5 del() [3/3]

```
virtual bool Table::del (
    int64_t record_rank ) [inline], [virtual]
```

del a row.

Parameters

<i>row_rank</i>	the n th record of the table
-----------------	------------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.6 finish()

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

finish, important interface for son class

Reimplemented in [RowTable](#).

4.45.3.7 getColumnRank()

```
int64_t Table::getColumnRank (
    int64_t c_id ) [inline]
```

get column rank in this table

Parameters

$c \leftrightarrow$ _id	column identifier
----------------------------	-------------------

Return values

\geq	0 valid rank
< 0	invalid, not exist

4.45.3.8 getColumnns()

```
std::vector< int64_t > & Table::getColumnns (
    void ) [inline]
```

get column identifier vector.

Return values

<i>vector</i>	of column identifiers
---------------	-----------------------

4.45.3.9 getIndexRank()

```
int64_t Table::getIndexRank (
    int64_t i_id ) [inline]
```

get index rank in this table

Parameters

$i \leftrightarrow$ _id	column identifier
----------------------------	-------------------

Return values

\geq	0 valid rank
< 0	invalid, not exist

4.45.3.10 getIndexes()

```
std::vector< int64_t > & Table::getIndexes (
    void ) [inline]
```

get index identifier vector.

4.45.3.11 getRank()

```
int64_t Table::getRank (
    std::vector< int64_t > & vec,
    int64_t id ) [inline]
```

get rank in a vector

Parameters

<i>vec</i>	vector to search in
<i>id</i>	object identifier

Return values

≥ 0	0 valid rank
< 0	invalid, not exist

4.45.3.12 getRecordNum()

```
virtual int64_t Table::getRecordNum (
    void ) [inline], [virtual]
```

get record number.

Reimplemented in [RowTable](#).

4.45.3.13 getRecordPtr()

```
virtual void * Table::getRecordPtr (
    int64_t row_rank ) [inline], [virtual]
```

get record pointer.

Reimplemented in [RowTable](#).

4.45.3.14 getTtype()

```
TableType Table::getTtype (
    void ) [inline]
```

get table type.

4.45.3.15 init()

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

init, important interface for son class

Reimplemented in [RowTable](#).

4.45.3.16 insert() [1/2]

```
virtual bool Table::insert (
    char * columns[] ) [inline], [virtual]
```

insert a row.

Parameters

<i>columns</i>	each element of the array pointed to a column data
----------------	--

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.17 insert() [2/2]

```
virtual bool Table::insert (
    char * source ) [inline], [virtual]
```

insert a row.

Parameters

<i>source</i>	buffer of a row in pattern
---------------	----------------------------

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.18 loadData()

```
virtual bool Table::loadData (
    const char * filename ) [inline], [virtual]
```

load data(not in use).

Reimplemented in [RowTable](#).

4.45.3.19 print()

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

print table information.

Reimplemented from [Object](#).

4.45.3.20 printData()

```
virtual bool Table::printData (
    void ) [inline], [virtual]
```

print data in table.

Reimplemented in [RowTable](#).

4.45.3.21 select() [1/2]

```
virtual bool Table::select (
    char * row_pointer,
    char * dest ) [inline], [virtual]
```

select all columns' data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.22 select() [2/2]

```
virtual bool Table::select (
    int64_t record_rank,
    char * dest ) [inline], [virtual]
```

select all columns' data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.23 selectCol() [1/2]

```
virtual bool Table::selectCol (
    char * row_pointer,
    int64_t column_rank,
    char * dest ) [inline], [virtual]
```

select one column data by pointer of a row.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_rank</i>	the n th column in table pattern
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.24 selectCol() [2/2]

```
virtual bool Table::selectCol (
    int64_t record_rank,
    int64_t column_rank,
    char * dest ) [inline], [virtual]
```

select one column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_rank</i>	the n th column in table pattern
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.25 selectCols() [1/2]

```
virtual bool Table::selectCols (
    char * row_pointer,
    int64_t column_total,
    int64_t * column_ranks,
    char * dest ) [inline], [virtual]
```

select several column data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.26 selectCols() [2/2]

```
virtual bool Table::selectCols (
    int64_t record_rank,
    int64_t column_total,
    int64_t * column_ranks,
    char * dest ) [inline], [virtual]
```

select several column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>dest</i>	buffer to store result

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.27 shut()

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

shut, important interface for son class

Reimplemented from [Object](#).

Reimplemented in [RowTable](#).

4.45.3.28 updateCol() [1/2]

```
virtual bool Table::updateCol (
    char * row_pointer,
    int64_t column_rank,
    char * source ) [inline], [virtual]
```

update one column data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_rank</i>	the n th column in table pattern
<i>source</i>	buffer to store data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.29 updateCol() [2/2]

```
virtual bool Table::updateCol (
    int64_t record_rank,
    int64_t column_rank,
    char * source ) [inline], [virtual]
```

update one column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_rank</i>	the n th column in table pattern
<i>source</i>	buffer to store data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.30 updateCols() [1/4]

```
virtual bool Table::updateCols (
    char * row_pointer,
    int64_t column_total,
    int64_t * column_ranks,
    char * source ) [inline], [virtual]
```

update several column data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>source</i>	buffer to store data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.31 updateCols() [2/4]

```
virtual bool Table::updateCols (
    char * row_pointer,
    int64_t column_total,
    int64_t * column_ranks,
    char * source[] ) [inline], [virtual]
```

update several column data.

Parameters

<i>row_pointer</i>	the pointer of a row
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>source</i>	array of columns' pointers, each points a column data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.3.32 updateCols() [3/4]

```
virtual bool Table::updateCols (
    int64_t record_rank,
    int64_t column_total,
    int64_t * column_ranks,
    char * source ) [inline], [virtual]
```

Reimplemented in [RowTable](#).

4.45.3.33 updateCols() [4/4]

```
virtual bool Table::updateCols (
    int64_t record_rank,
    int64_t column_total,
    int64_t * column_ranks,
    char * source[] ) [inline], [virtual]
```

update several column data.

Parameters

<i>record_rank</i>	the n th row in the table storage
<i>column_total</i>	total number of columns to select
<i>column_ranks</i>	array of column_rank, column_rank is the n th column in table pattern
<i>source</i>	array of columns' pointers, each points a column data to change for

Return values

<i>true</i>	success
<i>false</i>	failure

Reimplemented in [RowTable](#).

4.45.4 Member Data Documentation**4.45.4.1 t_columns**

```
std::vector< int64_t > Table::t_columns [private]
```

vector of columns' identifier

4.45.4.2 t_index

```
std::vector< int64_t > Table::t_index [private]
```

vector of index's identifier

4.45.4.3 t_type

```
TableType Table::t_type [private]
```

table type

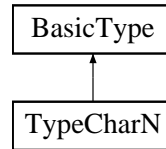
The documentation for this class was generated from the following file:

- system/[schema.h](#)

4.46 TypeCharN Class Reference

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

Inheritance diagram for TypeCharN:



Public Member Functions

- [TypeCharN](#) (int64_t typesize)
- [TypeCharN](#) ([TypeCode](#) typecode=[CHARN_TC](#), int64_t typesize=32)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.46.1 Detailed Description

definition of class [TypeCharN](#), please refer to [BasicType](#), it's same.

4.46.2 Constructor & Destructor Documentation

4.46.2.1 TypeCharN() [1/2]

```
TypeCharN::TypeCharN (
    int64_t typesize ) [inline]
```

constructor.

4.46.2.2 TypeCharN() [2/2]

```
TypeCharN::TypeCharN (
    TypeCode typecode = CHARN\_TC,
    int64_t typesize = 32 ) [inline]
```

4.46.3 Member Function Documentation

4.46.3.1 cmpEQ()

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

equal to.

Reimplemented from [BasicType](#).

4.46.3.2 cmpGE()

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

greater than or equal to

Reimplemented from [BasicType](#).

4.46.3.3 cmpGT()

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

greater than.

Reimplemented from [BasicType](#).

4.46.3.4 cmpLE()

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

less than or equal to.

Reimplemented from [BasicType](#).

4.46.3.5 cmpLT()

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

less than.

Reimplemented from [BasicType](#).

4.46.3.6 copy()

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

copy from data to dest.

Reimplemented from [BasicType](#).

4.46.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.46.3.8 formatTxt()

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

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

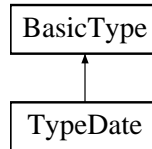
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.47 TypeDate Class Reference

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

Inheritance diagram for TypeDate:



Public Member Functions

- [TypeDate](#) ([TypeCode](#) typecode=[DATE_TC](#), int64_t typesize=sizeof(time_t))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.47.1 Detailed Description

definition of class [TypeDate](#), please refer to [BasicType](#), it's same.

4.47.2 Constructor & Destructor Documentation

4.47.2.1 TypeDate()

```
TypeDate::TypeDate (
    TypeCode typecode = DATE_TC,
    int64_t typesize = sizeof(time_t) ) [inline]
```

constructor.

4.47.3 Member Function Documentation

4.47.3.1 cmpEQ()

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

equal to.

Reimplemented from [BasicType](#).

4.47.3.2 cmpGE()

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

greater than or equal to

Reimplemented from [BasicType](#).

4.47.3.3 cmpGT()

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

greater than.

Reimplemented from [BasicType](#).

4.47.3.4 cmpLE()

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

less than or equal to.

Reimplemented from [BasicType](#).

4.47.3.5 cmpLT()

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

less than.

Reimplemented from [BasicType](#).

4.47.3.6 copy()

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

copy from data to dest.

Reimplemented from [BasicType](#).

4.47.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.47.3.8 formatTxt()

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

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

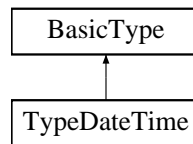
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.48 TypeDateTime Class Reference

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

Inheritance diagram for TypeDateTime:



Public Member Functions

- [TypeDateTime](#) ([TypeCode](#) typecode=[DATETIME_TC](#), int64_t typesize=sizeof(time_t))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.48.1 Detailed Description

definition of class [TypeDateTime](#), please refer to [BasicType](#), it's same.

4.48.2 Constructor & Destructor Documentation

4.48.2.1 TypeDateTime()

```
TypeDateTime::TypeDateTime (
    TypeCode typecode = DATETIME\_TC,
    int64_t typesize = sizeof(time_t) ) [inline]
```

constructor.

4.48.3 Member Function Documentation

4.48.3.1 cmpEQ()

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

equal to.

Reimplemented from [BasicType](#).

4.48.3.2 cmpGE()

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

greater than or equal to

Reimplemented from [BasicType](#).

4.48.3.3 cmpGT()

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

greater than.

Reimplemented from [BasicType](#).

4.48.3.4 cmpLE()

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

less than or equal to.

Reimplemented from [BasicType](#).

4.48.3.5 cmpLT()

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

less than.

Reimplemented from [BasicType](#).

4.48.3.6 copy()

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

copy from data to dest.

Reimplemented from [BasicType](#).

4.48.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.48.3.8 formatTxt()

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

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

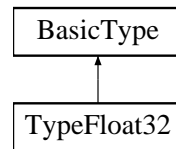
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.49 TypeFloat32 Class Reference

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

Inheritance diagram for TypeFloat32:



Public Member Functions

- [TypeFloat32](#) ([TypeCode](#) typecode=[FLOAT32_TC](#), int64_t typesize=sizeof(float))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.49.1 Detailed Description

definition of class [TypeFloat32](#), please refer to [BasicType](#), it's same.

4.49.2 Constructor & Destructor Documentation

4.49.2.1 TypeFloat32()

```
TypeFloat32::TypeFloat32 (
    TypeCode typecode = FLOAT32\_TC,
    int64_t typesize = sizeof\(float\) ) [inline]
```

constructor.

4.49.3 Member Function Documentation

4.49.3.1 cmpEQ()

```
bool TypeFloat32::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.49.3.2 cmpGE()

```
bool TypeFloat32::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.49.3.3 cmpGT()

```
bool TypeFloat32::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.49.3.4 cmpLE()

```
bool TypeFloat32::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.49.3.5 cmpLT()

```
bool TypeFloat32::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.49.3.6 copy()

```
int TypeFloat32::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.49.3.7 formatBin()

```
int TypeFloat32::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.49.3.8 formatTxt()

```
int TypeFloat32::formatTxt (
    void * dest,
    void * data ) [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

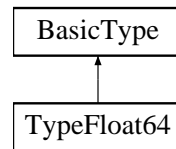
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.50 TypeFloat64 Class Reference

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

Inheritance diagram for TypeFloat64:



Public Member Functions

- [TypeFloat64](#) ([TypeCode](#) typecode=[FLOAT64_TC](#), int64_t typesize=sizeof(double))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.50.1 Detailed Description

definition of class [TypeFloat64](#), please refer to [BasicType](#), it's same.

4.50.2 Constructor & Destructor Documentation

4.50.2.1 TypeFloat64()

```
TypeFloat64::TypeFloat64 (  
    TypeCode typecode = FLOAT64\_TC,  
    int64_t typesize = sizeof(double) ) [inline]
```

constructor.

4.50.3 Member Function Documentation

4.50.3.1 cmpEQ()

```
bool TypeFloat64::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.50.3.2 cmpGE()

```
bool TypeFloat64::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.50.3.3 cmpGT()

```
bool TypeFloat64::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.50.3.4 cmpLE()

```
bool TypeFloat64::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.50.3.5 cmpLT()

```
bool TypeFloat64::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.50.3.6 copy()

```
int TypeFloat64::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.50.3.7 formatBin()

```
int TypeFloat64::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.50.3.8 formatTxt()

```
int TypeFloat64::formatTxt (
    void * dest,
    void * data ) [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

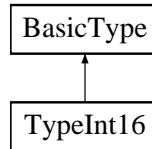
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.51 TypeInt16 Class Reference

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

Inheritance diagram for TypeInt16:



Public Member Functions

- [TypeInt16](#) ([TypeCode](#) typecode=[INT16_TC](#), [int64_t](#) typesize=sizeof([int16_t](#)))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.51.1 Detailed Description

definition of class [TypeInt16](#), please refer to [BasicType](#), it's same.

4.51.2 Constructor & Destructor Documentation

4.51.2.1 TypeInt16()

```
TypeInt16::TypeInt16 (
    TypeCode typecode = INT16\_TC,
    int64\_t typesize = sizeof(int16\_t) ) [inline]
```

constructor.

4.51.3 Member Function Documentation

4.51.3.1 cmpEQ()

```
bool TypeInt16::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.51.3.2 cmpGE()

```
bool TypeInt16::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.51.3.3 cmpGT()

```
bool TypeInt16::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.51.3.4 cmpLE()

```
bool TypeInt16::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.51.3.5 cmpLT()

```
bool TypeInt16::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.51.3.6 copy()

```
int TypeInt16::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.51.3.7 formatBin()

```
int TypeInt16::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.51.3.8 formatTxt()

```
int TypeInt16::formatTxt (
    void * dest,
    void * data ) [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

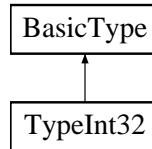
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.52 TypeInt32 Class Reference

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

Inheritance diagram for TypeInt32:



Public Member Functions

- [TypeInt32](#) ([TypeCode](#) typecode=[INT32_TC](#), [int64_t](#) typesize=sizeof([int32_t](#)))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.52.1 Detailed Description

definition of class [TypeInt32](#), please refer to [BasicType](#), it's same.

4.52.2 Constructor & Destructor Documentation

4.52.2.1 TypeInt32()

```
TypeInt32::TypeInt32 (
    TypeCode typecode = INT32\_TC,
    int64\_t typesize = sizeof\(int32\_t\) ) [inline]
```

constructor.

4.52.3 Member Function Documentation

4.52.3.1 cmpEQ()

```
bool TypeInt32::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.52.3.2 cmpGE()

```
bool TypeInt32::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.52.3.3 cmpGT()

```
bool TypeInt32::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.52.3.4 cmpLE()

```
bool TypeInt32::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.52.3.5 cmpLT()

```
bool TypeInt32::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.52.3.6 copy()

```
int TypeInt32::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.52.3.7 formatBin()

```
int TypeInt32::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.52.3.8 formatTxt()

```
int TypeInt32::formatTxt (
    void * dest,
    void * data ) [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

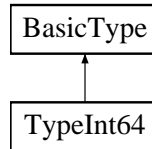
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.53 TypeInt64 Class Reference

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

Inheritance diagram for TypeInt64:



Public Member Functions

- [TypeInt64](#) ([TypeCode](#) typecode=[INT64_TC](#), [int64_t](#) typesize=sizeof([int64_t](#)))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.53.1 Detailed Description

definition of class [TypeInt64](#), please refer to [BasicType](#), it's same.

4.53.2 Constructor & Destructor Documentation

4.53.2.1 TypeInt64()

```
TypeInt64::TypeInt64 (
    TypeCode typecode = INT64\_TC,
    int64\_t typesize = sizeof\(int64\_t\) ) [inline]
```

constructor.

4.53.3 Member Function Documentation

4.53.3.1 cmpEQ()

```
bool TypeInt64::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.53.3.2 cmpGE()

```
bool TypeInt64::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.53.3.3 cmpGT()

```
bool TypeInt64::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.53.3.4 cmpLE()

```
bool TypeInt64::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.53.3.5 cmpLT()

```
bool TypeInt64::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.53.3.6 copy()

```
int TypeInt64::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.53.3.7 formatBin()

```
int TypeInt64::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.53.3.8 formatTxt()

```
int TypeInt64::formatTxt (
    void * dest,
    void * data ) [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

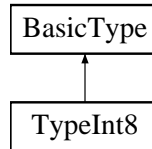
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.54 TypeInt8 Class Reference

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

Inheritance diagram for TypeInt8:



Public Member Functions

- [TypeInt8](#) ([TypeCode](#) typecode=[INT8_TC](#), [int64_t](#) typesize=sizeof([int8_t](#)))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.54.1 Detailed Description

definition of class [TypeInt8](#), please refer to [BasicType](#), it's same.

4.54.2 Constructor & Destructor Documentation

4.54.2.1 TypeInt8()

```
TypeInt8::TypeInt8 (
    TypeCode typecode = INT8\_TC,
    int64\_t typesize = sizeof\(int8\_t\) ) [inline]
```

constructor.

4.54.3 Member Function Documentation

4.54.3.1 cmpEQ()

```
bool TypeInt8::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.54.3.2 cmpGE()

```
bool TypeInt8::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.54.3.3 cmpGT()

```
bool TypeInt8::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.54.3.4 cmpLE()

```
bool TypeInt8::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.54.3.5 cmpLT()

```
bool TypeInt8::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.54.3.6 copy()

```
int TypeInt8::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.54.3.7 formatBin()

```
int TypeInt8::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.54.3.8 formatTxt()

```
int TypeInt8::formatTxt (
    void * dest,
    void * data ) [inline], [virtual]
```

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

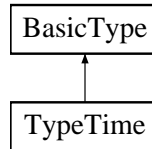
The documentation for this class was generated from the following file:

- [system/datatype.h](#)

4.55 TypeTime Class Reference

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

Inheritance diagram for TypeTime:



Public Member Functions

- [TypeTime](#) ([TypeCode](#) typecode=[TIME_TC](#), int64_t typesize=sizeof(time_t))
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpLT](#) (void *data1, void *data2)
- int [copy](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)

Additional Inherited Members

4.55.1 Detailed Description

definition of class [TypeTime](#), please refer to [BasicType](#), it's same.

4.55.2 Constructor & Destructor Documentation

4.55.2.1 TypeTime()

```
TypeTime::TypeTime (
    TypeCode typecode = TIME_TC,
    int64_t typesize = sizeof(time_t) ) [inline]
```

constructor.

4.55.3 Member Function Documentation

4.55.3.1 cmpEQ()

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

equal to.

Reimplemented from [BasicType](#).

4.55.3.2 cmpGE()

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

greater than or equal to

Reimplemented from [BasicType](#).

4.55.3.3 cmpGT()

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

greater than.

Reimplemented from [BasicType](#).

4.55.3.4 cmpLE()

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

less than or equal to.

Reimplemented from [BasicType](#).

4.55.3.5 cmpLT()

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

less than.

Reimplemented from [BasicType](#).

4.55.3.6 copy()

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

copy from data to dest.

Reimplemented from [BasicType](#).

4.55.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.55.3.8 formatTxt()

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

extract txt format from data(bin) to dest.

Reimplemented from [BasicType](#).

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

- [system/datatype.h](#)

Chapter 5

File Documentation

5.1 system/catalog.cc File Reference

```
#include "catalog.h"
```

Variables

- [Catalog g_catalog](#)

5.1.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.1.2 DESCRIPTION

this file provides an element container of database, which can be described as the stem of a tree with this [Catalog](#), the only one in global system, you can access all elements of system

basic usage:

(1) you use `g_catalog.createXXX[Database,Table,Column,Index]` to create objects. (2) you can get the pointer of `[Database,Table,Column,Index]` by call `g_catalog.getObjById/Name` (3) in `[Database,Table,Column,Index]`, you can define the relation of different objects by adding operation in those object (4) when all relation defined in database, you must call `initDatabase` to actually getting the database prepared to be used. (5) shut will get everything shutup can free `shutDatabase` will only shutup the selected database (6) for more tips, you may learn from `debug_catalog.cc`

5.1.3 Variable Documentation

5.1.3.1 g_catalog

`Catalog g_catalog`

5.2 system/catalog.d File Reference

5.3 system/catalog.h File Reference

```
#include <vector>
#include <unordered_map>
#include "schema.h"
#include "rowtable.h"
#include "hashindex.h"
#include "pbtreeindex.h"
```

Classes

- class [Catalog](#)

Variables

- [Catalog g_catalog](#)

5.3.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.3.2 DESCRIPTION

this file provides an element container of database, which can be described as the stem of a tree with this [Catalog](#), the only one in global system, you can access all elements of system

basic usage:

(1) you use `g_catalog.createXXX[Database,Table,Column,Index]` to create objects. (2) you can get the pointer of `[Database,Table,Column,Index]` by call `g_catalog.getObjById/Name` (3) in `[Database,Table,Column,Index]`, you can define the relation of different objects by adding operation in those object (4) when all relation defined in database, you must call `initDatabase` to actually getting the database prepared to be used. (5) shut will get everything shutup can free `shutDatabase` will only shutup the selected database (6) for more tips, you may learn from `debug_catalog.cc`

5.3.3 Variable Documentation

5.3.3.1 g_catalog

`Catalog g_catalog [extern]`

5.4 catalog.h

[Go to the documentation of this file.](#)

```

1
23 #ifndef _CATALOG_H
24 #define _CATALOG_H
25
26 #include <vector>
27 #include <unordered_map>
28 #include "schema.h"
29 #include "rowtable.h"
30 #include "hashindex.h"
31 #include "pbtreeindex.h"
32
33 class Catalog {
34 private:
35     std::vector<Object*> cl_id_obj;
36     std::unordered_map<std::string, Object*> cl_name_obj;
37 public:
38     void init(void) {
39         cl_id_obj.push_back(NULL);
40     }
41     bool shut(void);
42     bool createDatabase(const char *name, int64_t & d_id);
43     bool createTable(const char *name, TableType type, int64_t & t_id);
44     bool createColumn(const char *name, ColumnType type, int64_t option_size, int64_t & c_id);
45     bool createIndex(const char *name, IndexType type, Key i_key, int64_t & i_id);
46     bool initDatabase(int64_t d_id); // this function will truly init the inherited class
47     bool shutDatabase(int64_t d_id);
48     Object *getObjById(int64_t o_id);
49     Object *getObjByName(char *o_name);
50     void print(void) {
51         for (unsigned int ii = 0; ii < cl_id_obj.size(); ii++) {
52             Object *obj = cl_id_obj[ii];
53             if (obj == NULL)
54                 continue;
55             obj->print();
56         }
57     }
58 private:
59     bool initTable(int64_t t_id);
60     bool initColumn(int64_t c_id);
61     bool initIndex(int64_t i_id, int64_t t_id);
62     int64_t registerObj(Object * obj) {
63         cl_id_obj[obj->getOid()] = obj;
64         cl_name_obj.insert(std::pair< std::string,
65                                 Object *>(obj->getOname(), obj));
66         return obj->getOid();
67     }
68     int64_t obtainId(void) {
69         int64_t id = cl_id_obj.size();
70         cl_id_obj.push_back(NULL);
71         return id;
72     }
73 }; // class Catalog
74
75 extern Catalog g_catalog;
76 #endif

```

5.5 system/datatype.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <time.h>
```

Classes

- class [BasicType](#)
- class [TypeCharN](#)
- class [TypeDate](#)
- class [TypeDateTime](#)
- class [TypeFloat32](#)
- class [TypeFloat64](#)
- class [TypeInt16](#)
- class [TypeInt32](#)
- class [TypeInt64](#)
- class [TypeInt8](#)
- class [TypeTime](#)

Enumerations

- enum [TypeCode](#) {
 [INVID_TC](#) = 0 , [INT8_TC](#) , [INT16_TC](#) , [INT32_TC](#) ,
 [INT64_TC](#) , [FLOAT32_TC](#) , [FLOAT64_TC](#) , [CHARN_TC](#) ,
 [DATE_TC](#) , [TIME_TC](#) , [DATETIME_TC](#) , [MAXTYPE_TC](#) }

5.5.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.5.2 DESCRIPTION

all datatype supported by this system

5.5.3 Enumeration Type Documentation

5.5.3.1 TypeCode

enum [TypeCode](#)

data type code.

Enumerator

INVID_TC	
INT8_TC	int8
INT16_TC	int16
INT32_TC	int32
INT64_TC	int64
FLOAT32_TC	float32
FLOAT64_TC	float64
CHARN_TC	charn
DATE_TC	days from 1970-01-01 till current DATE
TIME_TC	seconds from 00:00:00 till current TIME
DATETIME_TC	seconds from 1970-01-01 00:00:00 till current DATETIME
MAXTYPE_TC	

5.6 datatype.h

[Go to the documentation of this file.](#)

```

1
12 #ifndef _DATATYPE_H
13 #define _DATATYPE_H
14
15 #include <stdio.h>
16 #include <stdint.h>
17 #include <string.h>
18 #include <time.h>
19
21 enum TypeCode {
22     INVID_TC = 0,
23     INT8_TC,
24     INT16_TC,
25     INT32_TC,
26     INT64_TC,
27     FLOAT32_TC,
28     FLOAT64_TC,
29     CHARN_TC,
30     DATE_TC,
31     TIME_TC,
32     DATETIME_TC,
33     MAXTYPE_TC
34 };
35
37 class BasicType {
38     protected:
39         TypeCode b_type_code;
40         int64_t b_type_size;
41     public:
42         BasicType(TypeCode typecode, int64_t typesize) {
43             b_type_code = typecode;
44             b_type_size = typesize;
45         }
46         virtual ~BasicType () {}
47         virtual int copy(void *dest, void *data) {
48             printf("[BasicType][ERROR][copy]: not support!\n");
49             return -1;
50         }
51         virtual bool cmpLT(void *data1, void *data2) {
52             printf("[BasicType][ERROR][cmpLT]: not support!\n");
53             return false;
54         }
55         virtual bool cmpEQ(void *data1, void *data2) {
56             printf("[BasicType][ERROR][cmpEQ]: not support!\n");
57             return false;
58         }
59         virtual bool cmpLE(void *data1, void *data2) {
60             printf("[BasicType][ERROR][cmpLE]: not support!\n");
61             return false;
62         }
63         virtual bool cmpGT(void *data1, void *data2) {

```

```

85     printf("[BasicType][ERROR][cmpGT: not support!\n");
86     return false;
87 }
91 virtual bool cmpGE(void *data1, void *data2) {
92     printf("[BasicType][ERROR][cmpGE]: not support!\n");
93     return false;
94 }
98 virtual int formatTxt(void *dest, void *data) {
99     printf("[BasicType][ERROR][formatTxt]: not support!\n");
100    return -2;
101 }
105 virtual int formatBin(void *dest, void *data) {
106     printf("[BasicType][ERROR][formatBin]: not support!\n");
107     return -3;
108 }
112 virtual int64_t getTypeSize(void) {
113     return b_type_size;
114 }
118 virtual TypeCode getTypeCode(void) {
119     return b_type_code;
120 }
121 };
122
124 class TypeInt8:public BasicType {
125 public:
129     TypeInt8(TypeCode typecode = INT8_TC, int64_t typesize = sizeof(int8_t)):BasicType(typecode,
130         typesize)
131     {
132     }
136     int copy(void *dest, void *data) {
137         *(int8_t *) dest = *(int8_t *) data;
138         return b_type_size;
139     }
140     int formatTxt(void *dest, void *data) {
141         return sprintf((char *) dest, "%d", (int) (*(int8_t *) data));
142     }
143     int formatBin(void *dest, void *data) {
144         int tmp;
145         sscanf((char *) data, "%d", &tmp);
146         if (tmp < 128 && tmp >= -128) {
147             *(int8_t *) dest = tmp;
148             return b_type_size;
149         } else {
150             printf("[TypeInt8][ERROR][formatBin]: data exceed range!\n");
151             return -1;
152         }
153     }
154     bool cmpLT(void *data1, void *data2) {
155         return *(int8_t *) data1 < *(int8_t *) data2;
156     }
157     bool cmpLE(void *data1, void *data2) {
158         return *(int8_t *) data1 <= *(int8_t *) data2;
159     }
160     bool cmpEQ(void *data1, void *data2) {
161         return *(int8_t *) data1 == *(int8_t *) data2;
162     }
163     bool cmpGT(void *data1, void *data2) {
164         return *(int8_t *) data1 > *(int8_t *) data2;
165     }
166     bool cmpGE(void *data1, void *data2) {
167         return *(int8_t *) data1 >= *(int8_t *) data2;
168     }
169 };
170
172 class TypeInt16:public BasicType {
173 public:
177     TypeInt16(TypeCode typecode = INT16_TC, int64_t typesize = sizeof(int16_t)):BasicType(typecode,
178         typesize)
179     {
180     }
184     int copy(void *dest, void *data) {
185         *(int16_t *) dest = *(int16_t *) data;
186         return b_type_size;
187     }
188     int formatTxt(void *dest, void *data) {
189         return sprintf((char *) dest, "%d", (int) (*(int16_t *) data));
190     }
191     int formatBin(void *dest, void *data) {
192         int tmp;
193         sscanf((char *) data, "%d", &tmp);
194         if (tmp < (1 << 16) && tmp >= -(1 << 16)) {
195             *(int16_t *) dest = tmp;
196             return b_type_size;
197         } else {
198             printf("[TypeInt16][ERROR][formatBin]: data exceed range!\n");
199             return -1;
200         }

```



```

201     }
202     bool cmpLT(void *data1, void *data2) {
203         return *(int16_t *) data1 < *(int16_t *) data2;
204     }
205     bool cmpLE(void *data1, void *data2) {
206         return *(int16_t *) data1 <= *(int16_t *) data2;
207     }
208     bool cmpEQ(void *data1, void *data2) {
209         return *(int16_t *) data1 == *(int16_t *) data2;
210     }
211     bool cmpGT(void *data1, void *data2) {
212         return *(int16_t *) data1 > *(int16_t *) data2;
213     }
214     bool cmpGE(void *data1, void *data2) {
215         return *(int16_t *) data1 >= *(int16_t *) data2;
216     }
217 };
218
219 class TypeInt32:public BasicType {
220 public:
221     TypeInt32(TypeCode typecode = INT32_TC, int64_t typesize = sizeof(int32_t)):BasicType(typecode,
222         typesize)
223     {
224     }
225     int copy(void *dest, void *data) {
226         *(int32_t *) dest = *(int32_t *) data;
227         return b_type_size;
228     }
229     int formatTxt(void *dest, void *data) {
230         return sprintf((char *) dest, "%d", (int) (*(int32_t *) data));
231     }
232     int formatBin(void *dest, void *data) {
233         int tmp;
234         sscanf((char *) data, "%d", &tmp);
235         *(int32_t *) dest = tmp;
236         return b_type_size;
237     }
238     bool cmpLT(void *data1, void *data2) {
239         return *(int32_t *) data1 < *(int32_t *) data2;
240     }
241     bool cmpLE(void *data1, void *data2) {
242         return *(int32_t *) data1 <= *(int32_t *) data2;
243     }
244     bool cmpEQ(void *data1, void *data2) {
245         return *(int32_t *) data1 == *(int32_t *) data2;
246     }
247     bool cmpGT(void *data1, void *data2) {
248         return *(int32_t *) data1 > *(int32_t *) data2;
249     }
250     bool cmpGE(void *data1, void *data2) {
251         return *(int32_t *) data1 >= *(int32_t *) data2;
252     }
253 };
254
255 class TypeInt64:public BasicType {
256 public:
257     TypeInt64(TypeCode typecode = INT64_TC, int64_t typesize = sizeof(int64_t)):BasicType(typecode,
258         typesize)
259     {
260     }
261     int copy(void *dest, void *data) {
262         *(int64_t *) dest = *(int64_t *) data;
263         return b_type_size;
264     }
265     int formatTxt(void *dest, void *data) {
266         return sprintf((char *) dest, "%ld",
267             (int64_t) (*(int64_t *) data));
268     }
269     int formatBin(void *dest, void *data) {
270         int64_t tmp;
271         sscanf((char *) data, "%ld", &tmp);
272         *(int64_t *) dest = tmp;
273         return b_type_size;
274     }
275     bool cmpLT(void *data1, void *data2) {
276         return *(int64_t *) data1 < *(int64_t *) data2;
277     }
278     bool cmpLE(void *data1, void *data2) {
279         return *(int64_t *) data1 <= *(int64_t *) data2;
280     }
281     bool cmpEQ(void *data1, void *data2) {
282         return *(int64_t *) data1 == *(int64_t *) data2;
283     }
284     bool cmpGT(void *data1, void *data2) {
285         return *(int64_t *) data1 > *(int64_t *) data2;
286     }
287     bool cmpGE(void *data1, void *data2) {
288         return *(int64_t *) data1 >= *(int64_t *) data2;
289     }
290 };

```

```

302         return *(int64_t *) data1 >= *(int64_t *) data2;
303     }
304 };
305
306 class TypeFloat32:public BasicType {
307 public:
308     TypeFloat32(TypeCode typecode = FLOAT32_TC, int64_t typesize = sizeof(float)):BasicType(typecode,
309         typesize)
310     {
311     }
312     int copy(void *dest, void *data) {
313         *(float *) dest = *(float *) data;
314         return b_type_size;
315     }
316     int formatTxt(void *dest, void *data) {
317         return sprintf((char *) dest, "%f", (float) (*(float *) data));
318     }
319     int formatBin(void *dest, void *data) {
320         float tmp;
321         sscanf((char *) data, "%f", &tmp);
322         *(float *) dest = tmp;
323         return b_type_size;
324     }
325     bool cmpLT(void *data1, void *data2) {
326         return *(float *) data1 < *(float *) data2;
327     }
328     bool cmpLE(void *data1, void *data2) {
329         return *(float *) data1 <= *(float *) data2;
330     }
331     bool cmpEQ(void *data1, void *data2) {
332         return *(float *) data1 == *(float *) data2;
333     }
334     bool cmpGT(void *data1, void *data2) {
335         return *(float *) data1 > *(float *) data2;
336     }
337     bool cmpGE(void *data1, void *data2) {
338         return *(float *) data1 >= *(float *) data2;
339     }
340 };
341
342 class TypeFloat64:public BasicType {
343 public:
344     TypeFloat64(TypeCode typecode = FLOAT64_TC, int64_t typesize = sizeof(double)):BasicType(typecode,
345         typesize)
346     {
347     }
348     int copy(void *dest, void *data) {
349         *(double *) dest = *(double *) data;
350         return b_type_size;
351     }
352     int formatTxt(void *dest, void *data) {
353         return sprintf((char *) dest, "%lf", (double) (*(double *) data));
354     }
355     int formatBin(void *dest, void *data) {
356         double tmp;
357         sscanf((char *) data, "%lf", &tmp);
358         *(double *) dest = tmp;
359         return b_type_size;
360     }
361     bool cmpLT(void *data1, void *data2) {
362         return *(double *) data1 < *(double *) data2;
363     }
364     bool cmpLE(void *data1, void *data2) {
365         return *(double *) data1 <= *(double *) data2;
366     }
367     bool cmpEQ(void *data1, void *data2) {
368         return *(double *) data1 == *(double *) data2;
369     }
370     bool cmpGT(void *data1, void *data2) {
371         return *(double *) data1 > *(double *) data2;
372     }
373     bool cmpGE(void *data1, void *data2) {
374         return *(double *) data1 >= *(double *) data2;
375     }
376 };
377
378 class TypeCharN:public BasicType {
379 public:
380     TypeCharN(int64_t typesize):BasicType(CHARN_TC, typesize) {
381     }
382     TypeCharN(TypeCode typecode = CHARN_TC, int64_t typesize = 32):BasicType(typecode,
383         typesize)
384     {
385     }
386     int copy(void *dest, void *data) {
387         strncpy((char *) dest, (char *) data, b_type_size);
388         return b_type_size;
389     }
390 };

```

```

410     }
411     int formatTxt(void *dest, void *data) {
412         strncpy((char *) dest, (char *) data, b_type_size);
413         return b_type_size;
414     }
415     int formatBin(void *dest, void *data) {
416         strncpy((char *) dest, (char *) data, b_type_size);
417         return b_type_size;
418     }
419     bool cmpLT(void *data1, void *data2) {
420         return strcmp((char *) data1, (char *) data2,
421             b_type_size) < 0 ? true : false;
422     }
423     bool cmpLE(void *data1, void *data2) {
424         return strcmp((char *) data1, (char *) data2,
425             b_type_size) <= 0 ? true : false;
426     }
427     bool cmpEQ(void *data1, void *data2) {
428         return strcmp((char *) data1, (char *) data2,
429             b_type_size) == 0 ? true : false;
430     }
431     bool cmpGT(void *data1, void *data2) {
432         return strcmp((char *) data1, (char *) data2,
433             b_type_size) > 0 ? true : false;
434     }
435     bool cmpGE(void *data1, void *data2) {
436         return strcmp((char *) data1, (char *) data2,
437             b_type_size) >= 0 ? true : false;
438     }
439 };
440
441 class TypeDate:public BasicType {
442 public:
443     TypeDate(TypeCode typecode = DATE_TC, int64_t typesize = sizeof(time_t)):BasicType(typecode,
444         typesize)
445     {
446     }
447     int copy(void *dest, void *data) {
448         *(time_t *) dest = *(time_t *) data;
449         return b_type_size;
450     }
451     int formatTxt(void *dest, void *data) {
452         struct tm *tt = localtime((time_t *) data);
453         return strftime((char *) dest, 32, "%Y-%m-%d", tt);
454     }
455     int formatBin(void *dest, void *data) {
456         struct tm tt = { 0 };
457         strptime((char *) data, "%Y-%m-%d", &tt);
458         time_t et = mktime(&tt);
459         *(time_t *) dest = et;
460         return b_type_size;
461     }
462     bool cmpLT(void *data1, void *data2) {
463         return *(time_t *) data1 < *(time_t *) data2;
464     }
465     bool cmpLE(void *data1, void *data2) {
466         return *(time_t *) data1 <= *(time_t *) data2;
467     }
468     bool cmpEQ(void *data1, void *data2) {
469         return *(time_t *) data1 == *(time_t *) data2;
470     }
471     bool cmpGT(void *data1, void *data2) {
472         return *(time_t *) data1 > *(time_t *) data2;
473     }
474     bool cmpGE(void *data1, void *data2) {
475         return *(time_t *) data1 >= *(time_t *) data2;
476     }
477 };
478
479 class TypeTime:public BasicType {
480 public:
481     TypeTime(TypeCode typecode = TIME_TC, int64_t typesize = sizeof(time_t)):BasicType(typecode,
482         typesize)
483     {
484     }
485     int copy(void *dest, void *data) {
486         *(time_t *) dest = *(time_t *) data;
487         return b_type_size;
488     }
489     int formatTxt(void *dest, void *data) {
490         struct tm *tt = localtime((time_t *) data);
491         return strftime((char *) dest, 32, "%H:%M:%S", tt);
492     }
493     int formatBin(void *dest, void *data) {
494         struct tm tt = { 0 };
495         strptime((char *) data, "%H:%M:%S", &tt);
496         time_t et = mktime(&tt);

```

```

511     *(time_t *) dest = et;
512     return b_type_size;
513 }
514 bool cmpLT(void *data1, void *data2) {
515     return *(time_t *) data1 < *(time_t *) data2;
516 }
517 bool cmpLE(void *data1, void *data2) {
518     return *(time_t *) data1 <= *(time_t *) data2;
519 }
520 bool cmpEQ(void *data1, void *data2) {
521     return *(time_t *) data1 == *(time_t *) data2;
522 }
523 bool cmpGT(void *data1, void *data2) {
524     return *(time_t *) data1 > *(time_t *) data2;
525 }
526 bool cmpGE(void *data1, void *data2) {
527     return *(time_t *) data1 >= *(time_t *) data2;
528 }
529 };
530
531 class TypeDateTime:public BasicType {
532 public:
533     TypeDateTime(TypeCode typecode = DATETIME_TC, int64_t typesize = sizeof(time_t)):BasicType(typecode,
534         typesize)
535     {
536     }
537     int copy(void *dest, void *data) {
538         *(time_t *) dest = *(time_t *) data;
539         return b_type_size;
540     }
541     int formatTxt(void *dest, void *data) {
542         struct tm *tt = localtime((time_t *) data);
543         return strftime((char *) dest, 32, "%Y-%m-%d %H:%M:%S", tt);
544     }
545     int formatBin(void *dest, void *data) {
546         struct tm tt = { 0 };
547         strptime((char *) data, "%Y-%m-%d %H:%M:%S", &tt);
548         time_t et = mktime(&tt);
549         *(time_t *) dest = et;
550         return b_type_size;
551     }
552     bool cmpLT(void *data1, void *data2) {
553         return *(time_t *) data1 < *(time_t *) data2;
554     }
555     bool cmpLE(void *data1, void *data2) {
556         return *(time_t *) data1 <= *(time_t *) data2;
557     }
558     bool cmpEQ(void *data1, void *data2) {
559         return *(time_t *) data1 == *(time_t *) data2;
560     }
561     bool cmpGT(void *data1, void *data2) {
562         return *(time_t *) data1 > *(time_t *) data2;
563     }
564     bool cmpGE(void *data1, void *data2) {
565         return *(time_t *) data1 >= *(time_t *) data2;
566     }
567 };
568 #endif

```

5.7 system/errorlog.cc File Reference

```

#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <stdarg.h>
#include <execinfo.h>
#include <cxxabi.h>
#include "errorlog.h"

```

Macros

- #define `EL_TOTAL_FILES` (sizeof(EL_src_file_name)/sizeof(char*))

Variables

- `const char * EL_src_file_name []`
- `ErrorLog_Thread_local * thread_el = NULL`

5.7.1 Detailed Description

Author

Shimin Chen chensm@ict.ac.cn

Version

0.1

5.7.2 Description

This file provides the error handling and logging utility.

5.7.3 Macro Definition Documentation

5.7.3.1 EL_TOTAL_FILES

```
#define EL_TOTAL_FILES (sizeof(EL_src_file_name)/sizeof(char*))
```

5.7.4 Variable Documentation

5.7.4.1 EL_src_file_name

```
const char* EL_src_file_name[]
```

Initial value:

```
= {  
    "ErrorLog.h",  
    "ErrorLog.cc",  
  
    "schema.h",  
    "schema.cc",  
    "rowtable.h",  
    "rowtable.cc",  
    "cursor.h",  
    "cursor.cc",  
    "hashindex.h",  
    "hashindex.cc",  
    "storeprocedure.h",  
    "storeprocedure.cc",  
    "tpcserver.h",  
    "tpcserver.cc",  
    "debug_Error.cc",  
    NULL  
}
```

5.7.4.2 thread_el

```
ErrorLog _Thread_local* thread_el = NULL
```

5.8 system/errorlog.d File Reference

5.9 system/errorlog.h File Reference

```
#include <pthread.h>
#include <stdio.h>
#include <string>
#include <unordered_map>
#include <time.h>
```

Classes

- class [ErrorLog](#)
an array of source file names

Macros

- `#define _Thread_local __thread`
- `#define EL_ASSERT(t)`
- `#define EL_BAD_FILEID (99999999)`
- `#define EL_DEBUG 1`
- `#define EL_ERRCODE() (thread_el->getErrorCode())`
- `#define EL_ERRMSG() (thread_el->getErrMsg())`
- `#define EL_ERROR 4`
- `#define EL_ERROR_CODE(fileid, lineno) ((fileid)*100000 + (lineno))`
- `#define EL_GET_FILEID(errcode) ((errcode)/100000)`
- `#define EL_GET_FILENAME(errcode) (ErrorLog::id2Name(EL_GET_FILEID(errcode)))`
- `#define EL_GET_LINENO(errcode) ((errcode)%100000)`
- `#define EL_INFO 2`
- `#define EL_LEVEL_COMPILE EL_INFO`
- `#define EL_LOG_DEBUG(...)`
- `#define EL_LOG_ERROR(...) thread_el->log(EL_ERROR, __FILE__, __LINE__, __VA_ARGS__)`
- `#define EL_LOG_INFO(...)`
- `#define EL_LOG_SERIOUS(...) thread_el->log(EL_SERIOUS, __FILE__, __LINE__, __VA_ARGS__)`
- `#define EL_LOG_WARN(...)`
- `#define EL_OK (0)`
- `#define EL_RESET() (thread_el->reset())`
- `#define EL_SERIOUS 5`
- `#define EL_WARN 3`

Variables

- `const char * EL_src_file_name []`
- `_Thread_local ErrorLog * thread_el`

5.9.1 Detailed Description

Author

Shimin Chen chensm@ict.ac.cn

Version

0.1

5.9.2 Description

This file provides the error handling and logging utility.

1. error code The error code is a decimal number with 8 digits:
FFFLLLL

The higher 3 digits show the source file ID, while the lower 5 digits indicate the line number in the source file, where the error occurs.

The following macros are useful:

```
EL_GET_FILEID(err_code)
EL_GET_LINENO(err_code)
EL_GET_FILENAME(err_code)
```

1. initialize

- (1) `main()` must call `ErrorLog::init(int level, const char *logfile);`
- (2) then every thread must new an `ErrorLog` instance:

```
thread_el= new ErrorLog("thread_name");
```

- (3) put source file names into `EL_src_file_name[]` in `ErrorLog.cc`

1. normal use

```
EL_LOG_DEBUG(format, args, ...); EL_LOG_INFO(format, args, ...); EL_LOG_WARN(format, args, ...);
EL_LOG_ERROR(format, args, ...); EL_LOG_SERIOUS(format, args, ...);
```

The message will be written into the specified log file as follows

[thread][level][[file:lineno](#)] message a call stack trace will be shown for error and serious messages.

Moreover, an assertion can be written as:

```
EL_ASSERT(expression);
```

The expression is a test that evaluates to a True or False value. If the value is False, then a debug assertion message will be generated and written to the log.

2. In a worker thread:

- (1) reset and clear the error messages

```
EL_RESET();
```

- (2) then follow the above to log messages

- (3) finally, obtain error code and message as follows

```
int err_code= EL_ERRCODE(); const char *err_msg= EL_ERRMSG();
```

- (4) optionally, flush the log file

```
ErrorLog::flushLog();
```

3. Before exiting, call the following globally once

```
ErrorLog::closeLog();
```

5.9.3 Macro Definition Documentation

5.9.3.1 `_Thread_local`

```
#define _Thread_local __thread
```

5.9.3.2 `EL_ASSERT`

```
#define EL_ASSERT(  
    t )
```

5.9.3.3 `EL_BAD_FILEID`

```
#define EL_BAD_FILEID (99999999)
```

5.9.3.4 `EL_DEBUG`

```
#define EL_DEBUG 1
```

5.9.3.5 `EL_ERRCODE`

```
#define EL_ERRCODE( ) (thread_el->getErrorCode())
```

5.9.3.6 `EL_ERRMSG`

```
#define EL_ERRMSG( ) (thread_el->getErrMsg())
```

5.9.3.7 `EL_ERROR`

```
#define EL_ERROR 4
```


5.9.3.8 EL_ERROR_CODE

```
#define EL_ERROR_CODE(  
    fileid,  
    lineno ) ((fileid)*100000 + (lineno))
```

5.9.3.9 EL_GET_FILEID

```
#define EL_GET_FILEID(  
    errcode ) ((errcode)/100000)
```

5.9.3.10 EL_GET_FILENAME

```
#define EL_GET_FILENAME(  
    errcode ) (ErrorLog::id2Name(EL_GET_FILEID(errcode)))
```

5.9.3.11 EL_GET_LINENO

```
#define EL_GET_LINENO(  
    errcode ) ((errcode)%100000)
```

5.9.3.12 EL_INFO

```
#define EL_INFO 2
```

5.9.3.13 EL_LEVEL_COMPILE

```
#define EL_LEVEL_COMPILE EL\_INFO
```

5.9.3.14 EL_LOG_DEBUG

```
#define EL_LOG_DEBUG(  
    ... )
```

5.9.3.15 EL_LOG_ERROR

```
#define EL_LOG_ERROR(  
    ... )    thread_el->log(EL_ERROR, __FILE__, __LINE__, __VA_ARGS__)
```

5.9.3.16 EL_LOG_INFO

```
#define EL_LOG_INFO(  
    ... )
```

Value:

```
do{ if (EL_INFO>=ErrorLog::el_level) \  
    thread_el->log(EL_INFO, __FILE__, __LINE__, __VA_ARGS__); }while(0)
```

5.9.3.17 EL_LOG_SERIOUS

```
#define EL_LOG_SERIOUS(  
    ... )    thread_el->log(EL_SERIOUS, __FILE__, __LINE__, __VA_ARGS__)
```

5.9.3.18 EL_LOG_WARN

```
#define EL_LOG_WARN(  
    ... )
```

Value:

```
do{ if (EL_WARN>=ErrorLog::el_level) \  
    thread_el->log(EL_WARN, __FILE__, __LINE__, __VA_ARGS__); }while(0)
```

5.9.3.19 EL_OK

```
#define EL_OK (0)
```

5.9.3.20 EL_RESET

```
#define EL_RESET( ) (thread_el->reset())
```

5.9.3.21 EL_SERIOUS

```
#define EL_SERIOUS 5
```

5.9.3.22 EL_WARN

```
#define EL_WARN 3
```

5.9.4 Variable Documentation

5.9.4.1 EL_src_file_name

```
const char* EL_src_file_name[] [extern]
```

5.9.4.2 thread_el

```
_Thread_local ErrorLog* thread_el [extern]
```

5.10 errorlog.h

[Go to the documentation of this file.](#)

```
1
78 #ifndef _ERRORLOG_H
79 #define _ERRORLOG_H
80
81 #include <pthread.h>
82 #include <stdio.h>
83 #include <string>
84 #include <unordered_map>
85 #include <time.h>
86
87 /* ----- */
88 /* Error Logging Macros */
89 /* ----- */
90
91 #define EL_DEBUG          1
92 #define EL_INFO           2
93 #define EL_WARN           3
94 #define EL_ERROR          4
95 #define EL_SERIOUS       5
96
97 // errors < this level will not be compiled into the executable
98 // this can be from EL_DEBUG to EL_ERROR.
99 // Note that we will always report any EL_ERROR and EL_SERIOUS messages.
100
101 #ifdef DEBUG
102 #define EL_LEVEL_COMPILE EL_DEBUG
103 #else
104 #define EL_LEVEL_COMPILE EL_INFO
105 #endif
106
107 #if EL_DEBUG >= EL_LEVEL_COMPILE
108 #define EL_LOG_DEBUG(...) \
```

```

109 do{ if (EL_DEBUG>=ErrorLog::el_level) \
110     thread_el->log(EL_DEBUG,__FILE__,__LINE__,__VA_ARGS__);while(0)
111 #else
112 #define EL_LOG_DEBUG(...)
113 #endif
114
115 #if EL_INFO >= EL_LEVEL_COMPILE
116 #define EL_LOG_INFO(...) \
117 do{ if (EL_INFO>=ErrorLog::el_level) \
118     thread_el->log(EL_INFO,__FILE__,__LINE__,__VA_ARGS__);while(0)
119 #else
120 #define EL_LOG_INFO(...)
121 #endif
122
123 #if EL_WARN >= EL_LEVEL_COMPILE
124 #define EL_LOG_WARN(...) \
125 do{ if (EL_WARN>=ErrorLog::el_level) \
126     thread_el->log(EL_WARN,__FILE__,__LINE__,__VA_ARGS__);while(0)
127 #else
128 #define EL_LOG_WARN(...)
129 #endif
130
131 #define EL_LOG_ERROR(...) \
132     thread_el->log(EL_ERROR,__FILE__,__LINE__,__VA_ARGS__)
133
134 #define EL_LOG_SERIOUS(...) \
135     thread_el->log(EL_SERIOUS,__FILE__,__LINE__,__VA_ARGS__)
136
137 #define EL_RESET()      (thread_el->reset())
138
139 #define EL_ERRCODE()    (thread_el->getErrorCode())
140
141 #define EL_ERRMSG()     (thread_el->getErrorMsg())
142
143 #if EL_DEBUG >= EL_LEVEL_COMPILE
144 #define EL_ASSERT(exp) \
145 do{ if (!(exp)) EL_LOG_DEBUG("Assertion failed: %s\n", #exp);} while(0)
146 #else
147 #define EL_ASSERT(t)
148 #endif
149
150 /* ----- */
151 /* Error Code */
152 /* ----- */
153
154 // Error code consists of 3 digits of file id and 5 digits of line number
155 // line number cannot
156 #define EL_ERROR_CODE(fileid, lineno) ((fileid)*100000 + (lineno))
157 #define EL_GET_FILEID(errcode) ((errcode)/100000)
158 #define EL_GET_LINENO(errcode) ((errcode)%100000)
159
160 #define EL_OK (0)
161 #define EL_BAD_FILEID (99999999)
162
163 #define EL_GET_FILENAME(errcode) \
164     (ErrorLog::id2Name(EL_GET_FILEID(errcode)))
165
166 // source file names
167 extern const char *EL_src_file_name[];
168
169 /* ----- */
170 /* class ErrorLog */
171 /* ----- */
172 class ErrorLog {
173
174     // ---
175     // class static
176     // ---
177
178 public:
179     static int el_level;
180     static const char *el_level_name[EL_SERIOUS + 1];
181
182 private:
183     static pthread_mutex_t el_lock;
184     static char *el_logfile;
185
186     static FILE *el_fp;
187     static std::unordered_map < std::string, int > *el_name_2_id;
188 public:
189     static void init(int level, const char *logfile);
190
191     static void setLevel(int level);
192
193     static void flushLog(void) {
194         if (el_fp)
195             fflush(el_fp);
196     }

```

```

213     } static void closeLog(void) {
217         if (el_fp)
218             fclose(el_fp);
219         el_fp = NULL;
220     }
221
222     static int name2Id(const char *src_name);
223
224     static const char *id2Name(int src_id);
225
226     // ---
227     // instance
228     // ---
229
230 private:
231     char *el_thread_name;
232
233     int el_err_code;
234
235     char *el_msg_buf;
236     int el_msg_cap;
237     char *el_msg_cur;
238
239     char *el_demangle_buf;
240     size_t el_demangle_len;
241     void *el_bt_buffer[256];
242
243     time_t el_tloc;
244     struct tm el_tm;
245
246     int getFuncNameGCC(char *bt_symbol);
247
248 public:
249     ErrorLog(const char *thread_name, int msg_cap = 256 * 1024);
250
251     ~ErrorLog();
252
253     void reset();
254
255     void log(int level, const char *src_name, const int lineno, ...);
256
257     int getErrorCode(void) {
258         return el_err_code;
259     }
260
261     const char *getErrorMsg(void) {
262         return el_msg_buf;
263     }
264
265     // ErrorLog
266 };
267
268 #ifndef _Thread_local
269 #define _Thread_local __thread
270 #endif
271
272 extern _Thread_local ErrorLog *thread_el;
273 #endif

```

5.11 system/executor.cc File Reference

```
#include "executor.h"
```

Functions

- `int64_t allocColBuf` (std::vector< int64_t > &col_id, int64_t &size_want, char *&buf_to_alloc)
- `int64_t easyAlloc` (int64_t size_want, char *&buf_to_alloc)
- `int64_t getTupleSize` (std::vector< int64_t > &col_id)

5.11.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.11.2 DESCRIPTION

definition of executor

5.11.3 Function Documentation

5.11.3.1 allocColBuf()

```
int64_t allocColBuf (
    std::vector< int64_t > & col_id,
    int64_t & size_want,
    char *& buf_to_alloc ) [inline]
```

Set record buffer for an input column

Parameters

<i>col_id</i>	reference of input column ID
<i>size_want</i>	size of buffer, used for return
<i>buf_to_alloc</i>	buffer to alloc, used for return

Return values

>0	actual allocated buffer size
<0	failure

5.11.3.2 easyAlloc()

```
int64_t easyAlloc (
    int64_t size_want,
    char *& buf_to_alloc ) [inline]
```

Author

zyl & dhk

5.11.3.3 getTupleSize()

```
int64_t getTupleSize (
    std::vector< int64_t > & col_id ) [inline]
```

Get size of a tuple according to its column ID

Parameters

<i>col_id</i>	reference of column ID
---------------	------------------------

Return values

<i>size</i>	of this tuple
-------------	---------------

5.12 system/executor.d File Reference

5.13 system/executor.h File Reference

```
#include "catalog.h"
#include "mymemory.h"
#include <map>
#include <algorithm>
#include <float.h>
```

Classes

- struct [AggreCondition](#)
- struct [Condition](#)
- struct [Conditions](#)
- class [Executor](#)
- class [Filter](#)
- class [GrAggRecord](#)
- struct [GroupbyAggre::group_by_hash](#)
- struct [GroupbyAggre::group_by_key](#)
- class [GroupbyAggre](#)
- class [HashJoin](#)
- class [IndexJoin](#)
- class [IndexScan](#)

- class [Join](#)
- class [Operator](#)
- class [Orderby](#)
- class [Project](#)
- struct [RequestColumn](#)
- struct [RequestTable](#)
- class [ResultTable](#)
- class [Scan](#)
- class [SelectQuery](#)

Enumerations

- enum [AggregateMethod](#) {
 [NONE_AM](#) = 0 , [COUNT](#) , [SUM](#) , [AVG](#) ,
 [MAX](#) , [MIN](#) , [MAX_AM](#) }
- enum [CompareMethod](#) {
 [NONE_CM](#) = 0 , [LT](#) , [LE](#) , [EQ](#) ,
 [NE](#) , [GT](#) , [GE](#) , [LINK](#) ,
 [MAX_CM](#) }

Functions

- `int64_t easyAlloc (int64_t size_want, char *&buf_to_alloc)`

5.13.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.13.2 DESCRIPTION

definition of executor

5.13.3 Enumeration Type Documentation

5.13.3.1 AggregateMethod

enum [AggregateMethod](#)

aggregate method.

Enumerator

NONE_AM	none
COUNT	count of rows
SUM	sum of data
AVG	average of data
MAX	maximum of data
MIN	minimum of data
MAX_AM	

5.13.3.2 CompareMethod

enum [CompareMethod](#)

compare method.

Enumerator

NONE_CM	
LT	less than
LE	less than or equal to
EQ	equal to
NE	not equal than
GT	greater than
GE	greater than or equal to
LINK	join
MAX_CM	

5.13.4 Function Documentation

5.13.4.1 easyAlloc()

```
int64_t easyAlloc (
    int64_t size_want,
    char *& buf_to_alloc ) [inline]
```

Wrapped function of alloc

Parameters

<i>size_want</i>	size wanted to alloc
<i>buf_to_alloc</i>	reference of new buffer addr

Return values

>0	actual allocated size
<0	failure

Author

zyl & dhk

5.14 executor.h

[Go to the documentation of this file.](#)

```

1
12 #ifndef _EXECUTOR_H
13 #define _EXECUTOR_H
14
15 #include "catalog.h"
16 #include "mymemory.h"
17
18 #include <map>
19 #include <algorithm>
20 #include <float.h>
21
22 enum AggregateMethod {
23     NONE_AM = 0,
24     COUNT,
25     SUM,
26     AVG,
27     MAX,
28     MIN,
29     MAX_AM
30 };
31
32 enum CompareMethod {
33     NONE_CM = 0,
34     LT,
35     LE,
36     EQ,
37     NE,
38     GT,
39     GE,
40     LINK,
41     MAX_CM
42 };
43
44 struct RequestColumn {
45     char name[128];
46     AggregateMethod aggregate_method;
47 };
48
49 struct RequestTable {
50     char name[128];
51 };
52
53 struct Condition {
54     RequestColumn column;
55     CompareMethod compare;
56     char value[128];
57 };
58
59 struct Conditions {
60     int condition_num;
61     Condition condition[4];
62 };
63
64 struct AggreCondition {
65     int column_rank;
66     AggregateMethod method;
67 };
68
69 class SelectQuery {
70 public:
71     int64_t database_id;
72     int select_number;
73     RequestColumn select_column[4];
74     int from_number;

```

```

83     RequestTable from_table[4];
84     Conditions where;
85     int groupby_number;
86     RequestColumn groupby[4];
87     Conditions having;
88     int orderby_number;
89     RequestColumn orderby[4];
90 }; // class SelectQuery
91
92 class ResultTable {
93 public:
94     int column_number;
95     BasicType **column_type;
96     char *buffer;
97     int64_t buffer_size;
98     int row_length;
99     int row_number;
100    int row_capacity;
101    int *offset;
102    int offset_size;
103
104    int init(BasicType *col_types[],int col_num,int64_t capacity = 1024);
105    char* getRC(int row, int column);
106    int writeRC(int row, int column, void *data);
107    int print(void);
108    int dump(FILE *fp);
109    int shut(void);
110
111    int append(char* src);
112 }; // class ResultTable
113
114 class Operator {
115 public:
116     char * buffer_from_father;
117     public:
118     Operator() {};
119     virtual ~Operator() {};
120
121     inline void setBuffer(char * buffer_allocated) {
122         buffer_from_father = buffer_allocated;
123     }
124     inline char * getBuffer() {
125         return buffer_from_father;
126     }
127
128     virtual bool open () {};
129     virtual bool getNext () {};
130     virtual bool close () {};
131 };
132
133 class Executor {
134 private:
135     SelectQuery *current_query;
136 public:
137     Operator* root;
138 public:
139     Operator* planner(SelectQuery *query);
140
141     int findCol(char* table_name, char* column_name);
142
143     int exec(SelectQuery *query, ResultTable *result);
144
145     int close();
146
147     int64_t getRank(std::vector < int64_t > &vec, int64_t id);
148 };
149
150 inline int64_t easyAlloc(int64_t size_want, char * & buf_to_alloc);
151
152 class Scan : public Operator {
153 private:
154     Table * scan_table;
155     int64_t total_record;
156     int64_t next_record;
157 public:
158     Scan() {};
159     ~Scan() {};
160
161     void setTable (Table * table) {
162         scan_table = table;
163     }
164
165     bool open ();
166     bool getNext ();
167     bool close ();
168 };
169
170 class IndexScan : public Operator {

```

```

311 private:
312     Table * from;
313     Index * index;
314     IndexType i_type;
315     void * info_ptr;
316     void * current_key;
317     bool key_end;
318 public:
319     void setTabIdx(Table * table, Index * index) {
320         this->from = table;
321         this->index = index;
322     }
323     IndexScan() {};
324     IndexScan(Table * table, Index * index) {
325         setTabIdx(table, index);
326     }
327     ~IndexScan() {};
328     bool open();
329     void updateKey(void * search_key);
330     bool getNext();
331     bool close();
332 };
333
334 class Filter : public Operator {
335 private:
336     Operator * child;
337     char * buf_for_child;
338     int64_t child_buf_size;
339     std::vector< int64_t > input_cid;
340     char * filt_pos;
341     int64_t filt_off;
342     CompareMethod cmp_mtd;
343     BasicType * filt_type;
344     bool (*cmp_func) (void *a, void *b, BasicType * data_type);
345     int64_t in_tuple_size;
346     char value[128];
347 private:
348     bool (*cmp_table [MAX_CM])(void * data1, void * data2, BasicType * data_type);
349     static inline bool cmpLT(void *data1, void *data2, BasicType * data_type){
350         return data_type->cmpLT(data1, data2);
351     }
352     static inline bool cmpLE(void *data1, void *data2, BasicType * data_type){
353         return data_type->cmpLE(data1, data2);
354     }
355     static inline bool cmpEQ(void *data1, void *data2, BasicType * data_type){
356         return data_type->cmpEQ(data1, data2);
357     }
358     static inline bool cmpNE(void *data1, void *data2, BasicType * data_type){
359         return !(data_type->cmpEQ(data1, data2));
360     }
361     static inline bool cmpGT(void *data1, void *data2, BasicType * data_type){
362         return data_type->cmpGT(data1, data2);
363     }
364     static inline bool cmpGE(void *data1, void *data2, BasicType * data_type){
365         return data_type->cmpGE(data1, data2);
366     }
367     void setFiltCond(int64_t filt_rank, CompareMethod cmp_mtd, char * value);
368     void initCmpFunc() {
369         cmp_table[NONE_CM] = NULL;
370         cmp_table[LT] = &cmpLT;
371         cmp_table[LE] = &cmpLE;
372         cmp_table[EQ] = &cmpEQ;
373         cmp_table[NE] = &cmpNE;
374         cmp_table[GT] = &cmpGT;
375         cmp_table[GE] = &cmpGE;
376         cmp_table[LINK] = NULL;
377     }
378 public:
379     void setChild(Operator * child) {
380         this->child = child;
381     }
382     bool setColumn(int64_t c_id [], int64_t num_column, int64_t filt_rank, CompareMethod cmp_mtd, char *
383     value) {
384         if (cmp_mtd == NONE_CM || cmp_mtd == LINK || cmp_mtd == MAX_CM) {
385             return false;
386         }
387         for (int64_t i = 0; i < num_column; i++){
388             input_cid.push_back(c_id[i]);
389         }
390         setFiltCond(filt_rank, cmp_mtd, value);
391         return true;
392     }
393     bool setColumn(std::vector< int64_t > input_cid, int64_t filt_rank, CompareMethod cmp_mtd, char *
394     value) {

```

```

520     if (cmp_mtd == NONE_CM || cmp_mtd == LINK || cmp_mtd == MAX_CM) {
521         return false;
522     }
523
524     this->input_cid = input_cid;
525     setFiltCond(filt_rank, cmp_mtd, value);
526
527     return true;
528 }
529
530 Filter() {
531     initCmpFunc();
532 };
533 ~Filter() {};
534 Filter(Operator * child, std::vector < int64_t > input_cid, int64_t filt_rank, CompareMethod
535 cmp_mtd, char * value) {
536     initCmpFunc();
537     setChild(child);
538     setColumn(input_cid, filt_rank, cmp_mtd, value);
539 }
540 Filter(Operator * child, int64_t c_id [], int64_t num_column, int64_t filt_rank, CompareMethod
541 cmp_mtd, char * value) {
542     initCmpFunc();
543     setChild(child);
544     setColumn(c_id, num_column, filt_rank, cmp_mtd, value);
545 }
546
547 bool open();
548 bool getNext();
549 bool close();
550
551 };
552
553 class Join : public Operator {
554 private:
555     Operator * left;
556     Operator * right;
557     std::vector < int64_t > left_cid;
558     std::vector < int64_t > right_cid;
559     int64_t left_rank;
560     int64_t right_rank;
561 public:
562     Join() {}
563     virtual ~Join() {}
564     void setLeftOp(Operator * lchild) {
565         left = lchild;
566     }
567     void setRightOp(Operator * rchild) {
568         right = rchild;
569     }
570     Operator * & getLeftOp() {
571         return left;
572     }
573     Operator * & getRightOp() {
574         return right;
575     }
576     void setJoinCol(std::vector < int64_t > left_cid, std::vector < int64_t > right_cid, int64_t
577 left_rank, int64_t right_rank) {
578         this->left_cid = left_cid;
579         this->right_cid = right_cid;
580         this->left_rank = left_rank;
581         this->right_rank = right_rank;
582     }
583     std::vector <int64_t> & getLeftCol() {
584         return left_cid;
585     }
586     std::vector <int64_t> & getRightCol() {
587         return right_cid;
588     }
589     int64_t getLeftRank() {
590         return left_rank;
591     }
592     int64_t getRightRank() {
593         return right_rank;
594     }
595     virtual bool open() = 0;
596     virtual bool getNext() = 0;
597     virtual bool close() = 0;
598 };
599
600 class IndexJoin : public Join {
601 private:
602     int64_t left_tuple_size;
603     int64_t right_tuple_size;
604     char * left_buf;
605     char * right_buf;
606     int64_t left_buf_size;

```

```

711     int64_t right_buf_size;
715     void * current_key;
716     bool right_has_next;
718 public:
722     IndexJoin() {}
726     ~IndexJoin() {}
727
735     IndexJoin(std::vector < int64_t > left_cid, std::vector < int64_t > right_cid, int64_t left_rank,
int64_t right_rank) {
736         setJoinCol(left_cid, right_cid, left_rank, right_rank);
737     }
738
744     bool open();
750     bool getNext();
756     bool close();
757 };
758
764 class HashJoin : public Join {
765 private:
766     char * left_buf;
767     char * right_buf;
769     std::vector <char *> middle_buf_array;
770     int64_t middle_buf_size;
772     int64_t left_tuple_size;
773     int64_t right_tuple_size;
774     int64_t right_buf_size;
776     bool right_has_next;
778     char * right_key_pos;
779     int64_t left_key_off;
780     BaseType * right_key_type;
781     BaseType * left_key_type;
783     char txt_buf [128];
784     std::multimap <std::string, char *> hash_index;
786     // Note: this should be arranged by lexicographic order
787     std::multimap <std::string, char *> ::iterator last_iter;
788     std::multimap <std::string, char *> ::iterator upper_iter;
790 public:
794     HashJoin() {}
798     ~HashJoin() {}
799     /*setIndex(std::vector < int64_t > left_cid, int64_t left_rank) {
800         // Create a Hash Index
801         //hash_index = new HashIndex()
802     }*/
810     HashJoin(std::vector < int64_t > left_cid, std::vector < int64_t > right_cid, int64_t left_rank,
int64_t right_rank) {
811         setJoinCol(left_cid, right_cid, left_rank, right_rank);
812     }
818     bool open();
824     bool getNext();
830     bool close();
831 };
832
837 class Project : public Operator {
838 private:
839     Operator * child;
840     char * buf_for_child;
842     int64_t self_buf_size;
844     std::vector < int64_t > input_cid;
845     std::vector < int64_t > output_cid;
846     std::vector < int64_t > out_to_in;
848     int64_t in_tuple_size;
849     int64_t in_buf_size;
851     std::vector < BaseType * > input_type;
852     std::vector < int64_t > input_off;
853     std::vector < char * > input_pos;
855     char * output_type;
856     int64_t output_type_size;
857     int64_t output_type_buf_size;
858     bool topid;
860 public:
864     Project() {};
868     ~Project() {
869         g_memory.free(output_type, output_type_buf_size);
870     };
875     void setChild(Operator * child) {
876         this -> child = child;
877     }
882     int64_t getColnum() {
883         return output_cid.size();
884     }
889     BaseType ** getSchema() {
890         return (BaseType **)output_type;
891     }
892
898     void setProjCol(std::vector <int64_t> in_cid, std::vector <int64_t> out_cid) {
899         input_cid = in_cid;
900         output_cid = out_cid;

```

```

901
902     output_type_size = sizeof(BasicType *) * output_cid.size();
903     //printf("set1");
904
905     output_type_buf_size = easyAlloc(output_type_size, output_type);
906     int64_t offset = 0;
907     //printf("set2");
908
909
910     for (int64_t i = 0; i < out_cid.size(); i++) {
911         // Add output's index from input
912         auto iter = std::find(input_cid.begin(), input_cid.end(), out_cid[i]);
913         out_to_in.push_back(std::distance(input_cid.begin(), iter));
914
915         //printf("set (%d)\n", i);
916         BasicType ** out_type = (BasicType **) (output_type + offset);
917         Column * column = (Column *) (g_catalog.getObjById(output_cid[i]));
918         *out_type = column -> getDataType();
919         offset += sizeof(BasicType *);
920     }
921 }
922
923 Project(std::vector<int64_t> in_cid, std::vector<int64_t> out_cid) {
924     setProjCol(in_cid, out_cid);
925 }
926
927 bool open();
928 bool getNext();
929 bool close();
930 bool top();
931 };
932
933 class GrAggRecord {
934 public:
935     char * middle_record;
936     std::vector<int64_t> sum;
937     std::vector<int64_t> count;
938     GrAggRecord(char * middle_record, int64_t num_aggr) {
939         this -> middle_record = middle_record;
940         sum.resize(num_aggr);
941         count.resize(num_aggr);
942     }
943 };
944
945 class GroupbyAggre : public Operator {
946 private:
947     Operator* child;
948     std::vector<int64_t> in_cid;
949     std::vector<int64_t> groupby_rank;
950     std::vector<int64_t> out_cid;
951     std::vector<AggreCondition> conditions;
952     char * buf_for_child;
953     int64_t child_buf_size;
954     int64_t child_tuple_size;
955     int64_t middle_tuple_size;
956     int64_t middle_buf_size;
957     std::vector<char * > middle_buf_array;
958     typedef std::vector<BasicType * > group_by_type_t;
959     group_by_type_t group_by_type;
960     std::vector<char * > group_by_pos;
961     std::vector<int64_t > group_by_size;
962     std::vector<BasicType * > aggr_type;
963     std::vector<char * > aggr_pos;
964     // Specify key type
965     typedef struct group_by_key {
966         group_by_type_t type_array;
967         std::vector<char *> value_array;
968     } group_by_key_t;
969     bool operator == (const group_by_key & k) const {
970         int64_t count = 0;
971         for (int64_t i = 0; i < value_array.size(); i++) {
972             if (k.type_array[i] -> cmpEQ(value_array[i], k.value_array[i])) {
973                 count++;
974             }
975         }
976         return count == value_array.size();
977     }
978     } group_by_key_t;
979     // Specify hash function
980     typedef struct group_by_hash {
981         size_t operator() (const group_by_key_t &key) const {
982             std::hash<std::string> hash_func;
983             size_t hash_val = 0;
984             char cmp_buffer[128];
985             for (int64_t i = 0; i < key.value_array.size(); i++) {
986                 key.type_array[i] -> formatTxt(cmp_buffer, key.value_array[i]);
987                 hash_val ^= hash_func(std::string(cmp_buffer)) + 0xCafeBabe + (hash_val « 3) + (hash_val »
988 1);
989             }
990         }
991     } group_by_hash_t;

```

```

1030         return hash_val;
1031     }
1032     } group_by_hash_t;
1033     std::unordered_map <group_by_key_t, GrAggRecord *, group_by_hash_t> hash_group;
1034     std::unordered_map <group_by_key_t, GrAggRecord *> ::iterator next_iter;
1035     static void sumInt8(void * sum, void * count, void * x) {
1036         *(int8_t *)sum += *(int8_t *)x;
1037     }
1038     static void sumInt16(void * sum, void * count, void * x) {
1039         *(int16_t *)sum += *(int16_t *)x;
1040     }
1041     static void sumInt32(void * sum, void * count, void * x) {
1042         *(int32_t *)sum += *(int32_t *)x;
1043     }
1044     static void sumInt64(void * sum, void * count, void * x) {
1045         *(int64_t *)sum += *(int64_t *)x;
1046     }
1047     static void sumFloat32(void * sum, void * count, void * x) {
1048         *(float *)sum += *(float *)x;
1049     }
1050     static void sumFloat64(void * sum, void * count, void * x) {
1051         *(double *)sum += *(double *)x;
1052     }
1053     void (*sum_table [MAXTYPE_TC])(void * sum, void * count, void * x);
1054
1055     static void avgInt8(void * sum, void * count, void * x) {
1056         sumInt8(sum, count, x);
1057         *(int64_t *)count += 1;
1058     }
1059     static void avgInt16(void * sum, void * count, void * x) {
1060         sumInt16(sum, count, x);
1061         *(int64_t *)count += 1;
1062     }
1063     static void avgInt32(void * sum, void * count, void * x) {
1064         sumInt32(sum, count, x);
1065         *(int64_t *)count += 1;
1066     }
1067     static void avgInt64(void * sum, void * count, void * x) {
1068         sumInt64(sum, count, x);
1069         *(int64_t *)count += 1;
1070     }
1071     static void avgFloat32(void * sum, void * count, void * x) {
1072         sumFloat32(sum, count, x);
1073         *(int64_t *)count += 1;
1074     }
1075     static void avgFloat64(void * sum, void * count, void * x) {
1076         sumFloat64(sum, count, x);
1077         *(int64_t *)count += 1;
1078     }
1079     void (*avg_table [MAXTYPE_TC])(void * sum, void * count, void * x);
1080     static void maxInt8(void * sum, void * count, void * x) {
1081         *(int8_t *)sum = std::max(*(int8_t *)sum, *(int8_t *)x);
1082     }
1083     static void maxInt16(void * sum, void * count, void * x) {
1084         *(int16_t *)sum = std::max(*(int16_t *)sum, *(int16_t *)x);
1085     }
1086     static void maxInt32(void * sum, void * count, void * x) {
1087         *(int32_t *)sum = std::max(*(int32_t *)sum, *(int32_t *)x);
1088     }
1089     static void maxInt64(void * sum, void * count, void * x) {
1090         *(int64_t *)sum = std::max(*(int64_t *)sum, *(int64_t *)x);
1091     }
1092     static void maxFloat32(void * sum, void * count, void * x) {
1093         *(float *)sum = std::max(*(float *)sum, *(float *)x);
1094     }
1095     static void maxFloat64(void * sum, void * count, void * x) {
1096         *(double *)sum = std::max(*(double *)sum, *(double *)x);
1097     }
1098     void (*max_table [MAXTYPE_TC])(void * sum, void * count, void * x);
1099     static void minInt8(void * sum, void * count, void * x) {
1100         *(int8_t *)sum = std::min(*(int8_t *)sum, *(int8_t *)x);
1101     }
1102     static void minInt16(void * sum, void * count, void * x) {
1103         *(int16_t *)sum = std::min(*(int16_t *)sum, *(int16_t *)x);
1104     }
1105     static void minInt32(void * sum, void * count, void * x) {
1106         *(int32_t *)sum = std::min(*(int32_t *)sum, *(int32_t *)x);
1107     }
1108     static void minInt64(void * sum, void * count, void * x) {
1109         *(int64_t *)sum = std::min(*(int64_t *)sum, *(int64_t *)x);
1110     }
1111     static void minFloat32(void * sum, void * count, void * x) {
1112         *(float *)sum = std::min(*(float *)sum, *(float *)x);
1113     }
1114     static void minFloat64(void * sum, void * count, void * x) {
1115         *(double *)sum = std::min(*(double *)sum, *(double *)x);
1116     }
1117 }

```



```

1266 void (*min_table [MAXTYPE_TC])(void * sum, void * count, void * x);
1273 static void count(void * sum, void * count, void * x) {
1274     *(int64_t *)count += 1;
1275 }
1276
1277 void (*aggr_method [4])(void * sum, void * count, void * x);
1284 static void initSum(void * sum, void * count) {
1285     *(int64_t *)sum = 0;
1286 }
1292 static void initCount(void * sum, void * count) {
1293     *(int64_t *)count = 0;
1294 }
1300 static void initAvg(void * sum, void * count) {
1301     initSum(sum, count);
1302     *(int64_t *)sum = 0;
1303 }
1309 static void initInt8Min(void * sum, void * count) {
1310     *(int64_t *)sum = INT8_MAX;
1311 }
1317 static void initInt16Min(void * sum, void * count) {
1318     *(int64_t *)sum = INT16_MAX;
1319 }
1325 static void initInt32Min(void * sum, void * count) {
1326     *(int64_t *)sum = INT32_MAX;
1327 }
1333 static void initInt64Min(void * sum, void * count) {
1334     *(int64_t *)sum = INT64_MAX;
1335 }
1341 static void initFloat32Min(void * sum, void * count) {
1342     *(float *)sum = FLT_MAX;
1343 }
1349 static void initFloat64Min(void * sum, void * count) {
1350     *(double *)sum = DBL_MAX;
1351 }
1353 void (*init_min_table [MAXTYPE_TC])(void * sum, void * count);
1359 static void initInt8Max(void * sum, void * count) {
1360     *(int64_t *)sum = INT8_MIN;
1361 }
1367 static void initInt16Max(void * sum, void * count) {
1368     *(int64_t *)sum = INT16_MIN;
1369 }
1375 static void initInt32Max(void * sum, void * count) {
1376     *(int64_t *)sum = INT32_MIN;
1377 }
1383 static void initInt64Max(void * sum, void * count) {
1384     *(int64_t *)sum = INT64_MIN;
1385 }
1391 static void initFloat32Max(void * sum, void * count) {
1392     *(float *)sum = FLT_MIN;
1393 }
1399 static void initFloat64Max(void * sum, void * count) {
1400     *(double *)sum = DBL_MIN;
1401 }
1403 void (*init_max_table [MAXTYPE_TC])(void * sum, void * count);
1404
1405 void (*init_method [4])(void * sum, void * count);
1413 static void finalInt8Sum(void * sum, void * count, void * result) {
1414     *(int8_t *)result = *(int8_t *)sum;
1415 }
1422 static void finalInt16Sum(void * sum, void * count, void * result) {
1423     *(int16_t *)result = *(int16_t *)sum;
1424 }
1430 static void finalInt32Sum(void * sum, void * count, void * result) {
1431     *(int32_t *)result = *(int32_t *)sum;
1432 }
1439 static void finalInt64Sum(void * sum, void * count, void * result) {
1440     *(int64_t *)result = *(int64_t *)sum;
1441 }
1447 static void finalFloat32Sum(void * sum, void * count, void * result) {
1448     *(float *)result = *(float *)sum;
1449 }
1456 static void finalFloat64Sum(void * sum, void * count, void * result) {
1457     *(double *)result = *(double *)sum;
1458 }
1460 void (*final_sum_table [MAXTYPE_TC])(void * sum, void * count, void * result);
1467 static void finalCount(void * sum, void * count, void * result) {
1468     *(int64_t *)result = *(int64_t *)count;
1469 }
1476 static void finalIntAvg(void * sum, void * count, void * result) {
1477     int64_t c = *(int64_t *)count;
1478     int64_t s = *(int64_t *)sum;
1479     *(double *)result = (double)s / (double)c;
1480 }
1487 static void finalFloat32Avg(void * sum, void * count, void * result) {
1488     int64_t c = *(int64_t *)count;
1489     float s = *(float *)sum;
1490     *(double *)result = (double)s / (double)c;

```

```

1491     }
1492     static void finalFloat64Avg(void * sum, void * count, void * result) {
1493         int64_t c = *(int64_t *)count;
1494         double s = *(double *)sum;
1495         *(double *)result = (double)s / (double)c;
1496     }
1497     void (*final_avg_table [MAXTYPE_TC])(void * sum, void * count, void * result);
1498
1499     void (*final_method [4])(void * sum, void * count, void * result);
1500 public:
1501     GroupbyAggre() {
1502         sum_table[INT8_TC] = &sumInt8;
1503         sum_table[INT16_TC] = &sumInt16;
1504         sum_table[INT32_TC] = &sumInt32;
1505         sum_table[INT64_TC] = &sumInt64;
1506         sum_table[FLOAT32_TC] = &sumFloat32;
1507         sum_table[FLOAT64_TC] = &sumFloat64;
1508
1509         avg_table[INT8_TC] = &avgInt8;
1510         avg_table[INT16_TC] = &avgInt16;
1511         avg_table[INT32_TC] = &avgInt32;
1512         avg_table[FLOAT32_TC] = &avgFloat32;
1513         avg_table[FLOAT64_TC] = &avgFloat64;
1514
1515         max_table[INT8_TC] = &maxInt8;
1516         max_table[INT16_TC] = &maxInt16;
1517         max_table[INT32_TC] = &maxInt32;
1518         max_table[INT64_TC] = &maxInt64;
1519         max_table[FLOAT32_TC] = &maxFloat32;
1520         max_table[FLOAT64_TC] = &maxFloat64;
1521
1522         min_table[INT8_TC] = &minInt8;
1523         min_table[INT16_TC] = &minInt16;
1524         min_table[INT32_TC] = &minInt32;
1525         min_table[INT64_TC] = &minInt64;
1526         min_table[FLOAT32_TC] = &minFloat32;
1527         min_table[FLOAT64_TC] = &minFloat64;
1528
1529         init_min_table[INT8_TC] = &initInt8Min;
1530         init_min_table[INT16_TC] = &initInt16Min;
1531         init_min_table[INT32_TC] = &initInt32Min;
1532         init_min_table[FLOAT32_TC] = &initFloat32Min;
1533         init_min_table[FLOAT64_TC] = &initFloat64Min;
1534
1535         init_max_table[INT8_TC] = &initInt8Max;
1536         init_max_table[INT16_TC] = &initInt16Max;
1537         init_max_table[INT32_TC] = &initInt32Max;
1538         init_max_table[FLOAT32_TC] = &initFloat32Max;
1539         init_max_table[FLOAT64_TC] = &initFloat64Max;
1540
1541         final_sum_table[INT8_TC] = &finalInt8Sum;
1542         final_sum_table[INT16_TC] = &finalInt16Sum;
1543         final_sum_table[INT32_TC] = &finalInt32Sum;
1544         final_sum_table[INT64_TC] = &finalInt64Sum;
1545         final_sum_table[FLOAT32_TC] = &finalFloat32Sum;
1546         final_sum_table[FLOAT64_TC] = &finalFloat64Sum;
1547
1548         final_avg_table[INT8_TC] = &finalIntAvg;
1549         final_avg_table[INT16_TC] = &finalIntAvg;
1550         final_avg_table[INT32_TC] = &finalIntAvg;
1551         final_avg_table[INT64_TC] = &finalIntAvg;
1552         final_avg_table[FLOAT32_TC] = &finalFloat32Avg;
1553         final_avg_table[FLOAT64_TC] = &finalFloat64Avg;
1554     };
1555     ~GroupbyAggre() {};
1556
1557     void set (std::vector<int64_t> input_colid, std::vector<int64_t> groupby_rank,
1558         std::vector<AggreCondition> conditions, std::vector<int64_t> output_colid) {
1559         this->in_cid = input_colid;
1560         this->groupby_rank = groupby_rank;
1561         this->conditions = conditions;
1562         this->out_cid = output_colid;
1563     }
1564     void setChild(Operator * child) {
1565         this->child = child;
1566     }
1567     bool open ();
1568     bool getNext ();
1569     bool close ();
1570 };
1571
1572 class Orderby : public Operator {
1573 private:
1574     Operator* child;
1575     std::vector<int64_t> colid;
1576     std::vector<int> colrank;

```

```

1618     std::vector<int> coloff;
1619     std::vector<BasicType*> coltype;
1620     int orderby_num;
1621     int64_t tuple_size;
1622     int64_t middle_buf_size;
1623     char* child_buffer;
1624     std::vector<char*> middle_buf_array;
1625     int arrayid;
1626     int64_t self_buf_size;
1627 public:
1631     Orderby() {};
1632
1636     ~Orderby() {};
1637
1643     void set (std::vector<int64_t> input_colid, std::vector<int> orderby_rank) {
1644         colid = input_colid;
1645         colrank = orderby_rank;
1646         orderby_num = colrank.size();
1647     }
1652     void setChild(Operator * child) {
1653         this -> child = child;
1654     }
1660     bool open    ();
1666     bool getNext ();
1672     bool close   ();
1673 };
1674
1675
1676
1677 #endif

```

5.15 system/gcc_pf_p3.h File Reference

Macros

- #define `pfld(mem_var) prefetcht0(mem_var)`
- #define `pfldnta(mem_var) prefetchnta(mem_var)`
- #define `pfst(mem_var) prefetcht0(mem_var)`
- #define `pfstnta(mem_var) prefetchnta(mem_var)`
- #define `prefetchnta(mem_var) __asm__ __volatile__ ("prefetchnta %0": : "m"(mem_var))`
- #define `prefetcht0(mem_var) __asm__ __volatile__ ("prefetcht0 %0": : "m"(mem_var))`
- #define `prefetcht1(mem_var) __asm__ __volatile__ ("prefetcht1 %0": : "m"(mem_var))`
- #define `ptouch(mem_var) __asm__ __volatile__ ("movl %0, %%eax": : "m"(mem_var):"%eax")`

5.15.1 Macro Definition Documentation

5.15.1.1 `pfld`

```

#define pfld(
    mem_var ) prefetcht0(mem_var)

```

5.15.1.2 `pfldnta`

```

#define pfldnta(
    mem_var ) prefetchnta(mem_var)

```

5.15.1.3 pfst

```
#define pfst(  
    mem_var ) prefetcht0(mem_var)
```

5.15.1.4 pfstnta

```
#define pfstnta(  
    mem_var ) prefetchnta(mem_var)
```

5.15.1.5 prefetchnta

```
#define prefetchnta(  
    mem_var ) __asm__ __volatile__ ("prefetchnta %0": : "m" (mem_var))
```

5.15.1.6 prefetcht0

```
#define prefetcht0(  
    mem_var ) __asm__ __volatile__ ("prefetcht0 %0": : "m" (mem_var))
```

5.15.1.7 prefetcht1

```
#define prefetcht1(  
    mem_var ) __asm__ __volatile__ ("prefetcht1 %0": : "m" (mem_var))
```

5.15.1.8 ptouch

```
#define ptouch(  
    mem_var ) __asm__ __volatile__ ("movl %0, %%eax": : "m" (mem_var) : "%eax")
```

5.16 gcc_pf_p3.h

[Go to the documentation of this file.](#)

```

1 /* File Name: gcc_pf_p3.h
2  * Author:    Shimin Chen
3  */
4
5 #ifndef _GCC_PREFETCH_P3
6 #define _GCC_PREFETCH_P3
7
8 #define prefetcht0(mem_var) \
9     __asm__ __volatile__ ("prefetcht0 %0": : "m" (mem_var))
10 #define prefetcht1(mem_var) \
11     __asm__ __volatile__ ("prefetcht1 %0": : "m" (mem_var))
12 // #define prefetcht2(mem_var) \
13 //     __asm__ __volatile__ ("prefetcht2 %0": : "m" (mem_var))
14 #define prefetchnta(mem_var) \
15     __asm__ __volatile__ ("prefetchnta %0": : "m" (mem_var))
16
17 /*
18     T0 (temporal data):
19         prefetch data into all levels of the cache hierarchy.
20         Pentium III processor 1st- or 2nd-level cache.
21         Pentium 4 and Intel Xeon processors 2nd-level cache.
22
23     T1 (temporal data with respect to first level cache)
24         prefetch data into level 2 cache and higher.
25         Pentium III processor 2nd-level cache.
26         Pentium 4 and Intel Xeon processors 2nd-level cache.
27
28     T2 (temporal data with respect to second level cache)
29         prefetch data into level 2 cache and higher.
30         Pentium III processor 2nd-level cache.
31         Pentium 4 and Intel Xeon processors 2nd-level cache.
32
33     NTA (non-temporal data with respect to all cache levels)
34         prefetch data into non-temporal cache structure and
35         into a location close to the processor, minimizing cache pollution.
36         Pentium III processor 1st-level cache
37         Pentium 4 and Intel Xeon processors 2nd-level cache
38 */
39
40 #define pfld(mem_var)      prefetcht0(mem_var)
41 #define pfst(mem_var)      prefetcht0(mem_var)
42 #define pfldnta(mem_var)  prefetchnta(mem_var)
43 #define pfstnta(mem_var)  prefetchnta(mem_var)
44
45 #define ptouch(mem_var) \
46     __asm__ __volatile__ ("movl %0, %%eax": : "m" (mem_var) : "%eax")
47
48 #endif /* _GCC_PREFETCH_P3 */

```

5.17 system/global.cc File Reference

```
#include "global.h"
```

Functions

- int [global_init](#) ()
- int [global_shut](#) ()

5.17.1 Function Documentation

5.17.1.1 `global_init()`

```
int global_init ( )
```

init memory and catalog.

5.17.1.2 `global_shut()`

```
int global_shut ( )
```

shut down catalog and memory.

5.18 `system/global.d` File Reference

5.19 `system/global.h` File Reference

```
#include "memory.h"  
#include "catalog.h"
```

Macros

- `#define BNODE_POINTERS_NUM` (16)
- `#define GLOBAL_MEMORY_MINIMUM` (1L<< 3)
- `#define GLOBAL_MEMORY_SIZE` (1L<<31)

Functions

- int `global_init` ()
- int `global_shut` ()

Variables

- Catalog `g_catalog`
- Memory `g_memory`

5.19.1 Macro Definition Documentation

5.19.1.1 `BNODE_POINTERS_NUM`

```
#define BNODE_POINTERS_NUM (16)
```

5.19.1.2 GLOBAL_MEMORY_MINIMUM

```
#define GLOBAL_MEMORY_MINIMUM (1L<< 3)
```

5.19.1.3 GLOBAL_MEMORY_SIZE

```
#define GLOBAL_MEMORY_SIZE (1L<<31)
```

5.19.2 Function Documentation

5.19.2.1 global_init()

```
int global_init ( )
```

init memory and catalog.

5.19.2.2 global_shut()

```
int global_shut ( )
```

shut down catalog and memory.

5.19.3 Variable Documentation

5.19.3.1 g_catalog

```
Catalog g_catalog [extern]
```

5.19.3.2 g_memory

```
Memory g_memory [extern]
```

5.20 global.h

[Go to the documentation of this file.](#)

```
1 #include "memory.h"
2 #include "catalog.h"
3
4 #define GLOBAL_MEMORY_SIZE      (1L<<31)
5 #define GLOBAL_MEMORY_MINIMUM  (1L<< 3)
6
7 #define BNODE_POINTERS_NUM      (16)          // shoule be 2*m, it's a default value and strongly advised
8
9 extern Memory g_memory;
10 extern Catalog g_catalog;
11
15 int global_init();
16
20 int global_shut();
```

5.21 system/hashindex.cc File Reference

```
#include "hashindex.h"
```

5.21.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.21.2 DESCRIPTION

hash index, here we support all type data. and you can use multi-keys, and I will guarantee it's right the value accessed at Element is the pointer of related recored. this implementation permits duplicated key with different value to be inserted, but not element with same key and value del operation will only del the first one which meet requirement, if you want delete all, you can call it many times till it returns false

@basic usage:

for each insert,del,look,scan, this file provides 2 same name method to handle 2 type data format you can use (1) for lookup, you should all use set_Is to set [HashInfo](#) with proper value, the first param is valid, leave the second to be NULL, [HashInfo](#) can help you iterately get values you need. (2) call lookup to get the value iterately.

5.22 system/hashindex.d File Reference

5.23 system/hashindex.h File Reference

```
#include "schema.h"
#include "hashtable.h"
```


Classes

- class [HashIndex](#)
- struct [HashInfo](#)

Macros

- `#define` [HASHINFO_CAPICITY](#) (8)

5.23.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.23.2 DESCRIPTION

hash index, here we support all type data. and you can use multi-keys, and I will guarantee it's right the value accessed at Element is the pointer of related recored. this implementation permits duplicated key with different value to be inserted, but not element with same key and value del operation will only del the first one which meet requirement, if you want delete all, you can call it many times till it returns false

@basic usage:

for each insert,del,look,scan, this file provides 2 same name method to handle 2 type data format you can use (1) for lookup, you should all use set_Is to set [HashInfo](#) with proper value, the first param is valid, leave the second to be NULL, [HashInfo](#) can help you iterately get values you need. (2) call lookup to get the value iterately.

5.23.3 Macro Definition Documentation

5.23.3.1 HASHINFO_CAPICITY

```
#define HASHINFO_CAPICITY (8)
```

5.24 hashindex.h

[Go to the documentation of this file.](#)

```

1
22 #ifndef _HASHINDEX_H
23 #define _HASHINDEX_H
24
25 #include "schema.h"
26 #include "hashtable.h"
27
28 // support INT and CHARN, id type data
29 #define HASHINFO_CAPACITY (8)
30
31 struct HashInfo {
32     char *result[HASHINFO_CAPACITY];
33     int rnum;
34     int ppos;
35     int last;
36     int64_t hash;
37 };
38
39
40 class HashIndex:public Index {
41 private:
42     HashTable *ih_hashtable;
43     int64_t ih_cell_capbits;
44     int64_t *ih_hash_bits;
45     BaseType **ih_datatype;
46     int64_t ih_column_num;
47     int64_t ih_column_cap;
48     int64_t *ih_table_offset;
49 public:
50     HashIndex(int64_t h_id, const char *i_name, Key & i_key)
51         :Index(h_id, i_name, HASHINDEX, i_key)
52     {
53     }
54     // init process bool init(void);
55     bool init (void);
56     void setCellCap(int64_t cell_capbits) { // cell size is power of cell_capbits by 2
57         this->ih_cell_capbits = cell_capbits;
58     }
59     bool addIndexDTpye(BaseType * i_dt, int64_t offset);
60     bool finish(void);
61     bool shut(void);
62
63     // data operator
64     bool insert(void *i_data, void *p_in);
65     bool insert(void *i_data[], void *p_in);
66     bool set_ls(void *i_data1, void *i_data2, void *info);
67     bool set_ls(void *i_data1[], void *i_data2[], void *info);
68     bool lookup(void *i_data, void *info, void *&result);
69     bool lookup(void *i_data[], void *info, void *&result);
70     bool del(void *i_data);
71     bool del(void *i_data[]);
72     void print(void);
73
74 private:
75     int64_t tranToInt64(void *i_data);
76     int64_t tranToInt64(void *i_data[]);
77     int64_t hash(char *str, int64_t maxlen);
78     bool cmpEQ(void *i_data[], void *result);
79     bool cmpEQ(void *i_data, void *result);
80 };
81 #endif

```

5.25 system/hashtable.cc File Reference

```
#include "hashtable.h"
```

Macros

- #define [ESTIMATE_ERROR](#) (1024)

5.25.1 Macro Definition Documentation

5.25.1.1 ESTIMATE_ERROR

```
#define ESTIMATE_ERROR (1024)
```

5.26 system/hashtable.d File Reference

5.27 system/hashtable.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <unordered_map>
#include "mymemory.h"
```

Classes

- class [HashCell](#)
- class [Hashcode_Ptr](#)
- class [HashTable](#)

Macros

- #define [hc_capacity](#) hc_union.num_2_or_more.capacity
- #define [hc_ent](#) hc_union.ent
- #define [hc_ents](#) hc_union.num_2_or_more.ents

5.27.1 Macro Definition Documentation

5.27.1.1 hc_capacity

```
#define hc_capacity hc_union.num_2_or_more.capacity
```

5.27.1.2 hc_ent

```
#define hc_ent hc_union.ent
```

5.27.1.3 hc_ents

```
#define hc_ents hc_union.num_2_or_more.ents
```

5.28 hashtable.h

[Go to the documentation of this file.](#)

```
1
15 #ifndef _HASH_TABLE_H
16 #define _HASH_TABLE_H
17 /* ----- */
18
19 #include <stdio.h>
20 #include <stdint.h>
21 #include <stdlib.h>
22 #include <string.h>
23 #include <unordered_map>
24 #include "mymemory.h"
25
26 /* ----- */
27
28 class Hashcode_Ptr {
29 public:
30     int64_t hash_code;
31     char *tuple;
32 }; // class Hashcode_Ptr
33
34 class HashCell {
35 public:
36     int hc_num;
37     union {
38         Hashcode_Ptr ent;
39         struct { // not hit, search a array, not a link-list, decrease cache miss
40             int capacity;
41             Hashcode_Ptr *ents;
42         } num_2_or_more;
43     } hc_union;
44 }; // class HashCell
45
46 #define hc_ent hc_union.ent
47 #define hc_capacity hc_union.num_2_or_more.capacity
48 #define hc_ents hc_union.num_2_or_more.ents
49
50 class HashTable {
51 public:
52     int estimated_num_distinct_keys;
53     double estimated_duplicates_per_key;
54     int initial_array_size;
55     char *begin;
56     Hashcode_Ptr *free_header[16];
57
58     // allocate the array
59     // free_header[0]: all of initial_array_size
60     // free_header[1]: 2*initial_array_size
61     // ...
62     // free_header[k]: 2^k * initial_array_size
63
64     Hashcode_Ptr *avail;
65     Hashcode_Ptr *end;
66     HashCell *table;
67     int table_size;
68     int more_allocated;
69 private:
70     std::unordered_map<void*, int> pointer2size;
71     void *allocate(int size);
72     void free(void *mem);
73     int size_to_slot(int array_size);
74 public:
75     HashTable(int estimatedNumDistinctKeys, double estimatedDupPerKey,
76               int num_partitions);
77     ~HashTable();
```

```

88
96     bool add(int64_t hashCode, char *tup);
104     bool del(int64_t hashCode, char *tup);
105
106     // return num matched
107     // <0: means capacity has been reached, there could be more
108     // -ret is the last position probed
109     int probe(int64_t hashCode, char *match[], int capacity);
110     int probe_contd(int64_t hashCode, int last, char *match[],
111                     int capacity);
112     void utilization();
113     void show();
114
115 };
116                                     //HashTable
117
118 /* ----- */
119 #endif                                     /* _HASH_TABLE_H */

```

5.29 system/mymemory.cc File Reference

```
#include "mymemory.h"
```

Variables

- [Memory g_memory](#)

5.29.1 Detailed Description

@author liugang(liugang@ict.ac.cn)

Version

0.1

5.29.2 DESCRIPTION

[Memory](#) is system own manager to alloc and free slab of different size the minmux size is sizeof(void*), maxsize is given by m_array_list size

interface: int64_t alloc (char *&p, int64_t size) int64_t free (char *p, int64_t size) you should put down the size of memory allocated, when you free back, you need this data

5.29.3 Variable Documentation

5.29.3.1 g_memory

[Memory](#) g_memory

5.30 system/mymemory.d File Reference

5.31 system/mymemory.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <vector>
#include <sys/mman.h>
```

Classes

- class [Memory](#)

Macros

- #define [MEMORY_OK](#) 0
- #define [NON_TABLE_MEMORY_ADDR](#) ((void*)0x600000000000)
- #define [TABLE_MEMORY_ALLOC_INC](#) (1L<<28)
- #define [TABLE_MEMORY_ALLOC_MAX](#) (1L<<34)
- #define [TABLE_MEMORY_INIT_ADDR](#) ((void*)0x700000000000)
- #define [TABLE_MEMORY_MAX_ADDR](#) ((void*)0x7f0000000000)

Variables

- [Memory](#) [g_memory](#)

5.31.1 Detailed Description

@author liugang(liugang@ict.ac.cn)

Version

0.1

5.31.2 DESCRIPTION

[Memory](#) is system own manager to alloc and free slab of different size the minmux size is sizeof(void*), maxsize is given by m_array_list size

interface: int64_t alloc (char *&p, int64_t size) int64_t free (char *p, int64_t size) you should put down the size of memory allocated, when you free back, you need this data

5.31.3 Macro Definition Documentation

5.31.3.1 MEMORY_OK

```
#define MEMORY_OK 0
```

5.31.3.2 NON_TABLE_MEMORY_ADDR

```
#define NON_TABLE_MEMORY_ADDR ((void*)0x600000000000)
```

5.31.3.3 TABLE_MEMORY_ALLOC_INC

```
#define TABLE_MEMORY_ALLOC_INC (1L<<28)
```

5.31.3.4 TABLE_MEMORY_ALLOC_MAX

```
#define TABLE_MEMORY_ALLOC_MAX (1L<<34)
```

5.31.3.5 TABLE_MEMORY_INIT_ADDR

```
#define TABLE_MEMORY_INIT_ADDR ((void*)0x700000000000)
```

5.31.3.6 TABLE_MEMORY_MAX_ADDR

```
#define TABLE_MEMORY_MAX_ADDR ((void*)0x7f0000000000)
```

5.31.4 Variable Documentation

5.31.4.1 g_memory

```
Memory g_memory [extern]
```

5.32 mymemory.h

[Go to the documentation of this file.](#)

```

1
16 #ifndef _MYMEMORY_H
17 #define _MYMEMORY_H
18
19 #include <stdint.h>
20 #include <stdio.h>
21 #include <vector>
22 #include <sys/mman.h>
23
24 #define MEMORY_OK 0
25
26 #define NON_TABLE_MEMORY_ADDR ((void*)0x600000000000)
27 #define TABLE_MEMORY_INIT_ADDR ((void*)0x700000000000)
28 #define TABLE_MEMORY_MAX_ADDR ((void*)0x7f0000000000)
29 #define TABLE_MEMORY_ALLOC_MAX (1L<34) // maximum 16 GB per table
30 #define TABLE_MEMORY_ALLOC_INC (1L<28) // fixed 256 MB per increment
31
32 class Memory {
33 private:
34     char *m_head;
35     char *m_curr;
36     char *m_tail;
37     char **m_array_list;
38     int64_t m_total;
39     int64_t m_mins;
40     char *m_table_addr;
41
42 public:
43     int init(int64_t total, int64_t mins);
44     int64_t alloc(char *&p, int64_t size);
45     int64_t free(char *p, int64_t size);
46     int shut(void);
47     int print(void);
48     int allocTableAddr (char *&p);
49
50 private:
51     unsigned int slot(int64_t size);
52     int64_t alloc_default(char *&p, int64_t size);
53 }; // class Memory
54
55 extern Memory g_memory;
56 #endif

```

5.33 system/nodepref.h File Reference

```
#include "gcc_pf_p3.h"
```

Macros

- #define [AREA_LINE_NUM](#) (([NODE_LINE_NUM](#)+1)/2)
- #define [BNODE_SIZE](#) ([CACHE_LINE_SIZE](#) * [NODE_LINE_NUM](#))
- #define [CACHE_LINE_SIZE](#) [L3_CACHE_LINE](#)
- #define [ITEM_SIZE](#) (8)
- #define [L3_CACHE_LINE](#) (64)
- #define [LEAF_PREF](#) [NODE_PREF](#)
- #define [LEAF_PREF_ST](#) [NODE_PREF_ST](#)
- #define [NODE_LINE_NUM](#) (4)

5.33.1 Macro Definition Documentation

5.33.1.1 AREA_LINE_NUM

```
#define AREA_LINE_NUM ((NODE_LINE_NUM+1)/2)
```

5.33.1.2 BNODE_SIZE

```
#define BNODE_SIZE (CACHE_LINE_SIZE * NODE_LINE_NUM)
```

5.33.1.3 CACHE_LINE_SIZE

```
#define CACHE_LINE_SIZE L3_CACHE_LINE
```

5.33.1.4 ITEM_SIZE

```
#define ITEM_SIZE ( 8)
```

—X86_64 prefetch file—

5.33.1.5 L3_CACHE_LINE

```
#define L3_CACHE_LINE (64)
```

5.33.1.6 LEAF_PREF

```
#define LEAF_PREF NODE_PREF
```

5.33.1.7 LEAF_PREF_ST

```
#define LEAF_PREF_ST NODE_PREF_ST
```

5.33.1.8 NODE_LINE_NUM

```
#define NODE_LINE_NUM ( 4)
```

5.34 nodepref.h

[Go to the documentation of this file.](#)

```

1
15 #ifndef _PBTREE_NODEPREF_H
16 #define _PBTREE_NODEPREF_H
17 /* ----- */
19 #include "gcc_pf_p3.h"
20
21 #define ITEM_SIZE      ( 8)
22 #define NODE_LINE_NUM ( 4)
23 #define L3_CACHE_LINE (64)
24 // node size
25 #define CACHE_LINE_SIZE      L3_CACHE_LINE
26 #define BNODE_SIZE           (CACHE_LINE_SIZE * NODE_LINE_NUM)
27 #define AREA_LINE_NUM       ((NODE_LINE_NUM+1)/2)
28
29 static void inline NODE_PREF(register void *bbp)
30 {
31     pfld (* ((char *)bbp));
32     #   if NODE_LINE_NUM >= 2
33     pfld (* ((char *)bbp + CACHE_LINE_SIZE));
34     #   endif
35     #   if NODE_LINE_NUM >= 3
36     pfld (* ((char *)bbp + CACHE_LINE_SIZE*2));
37     #   endif
38     #   if NODE_LINE_NUM >= 4
39     pfld (* ((char *)bbp + CACHE_LINE_SIZE*3));
40     #   endif
41     #   if NODE_LINE_NUM >= 5
42     pfld (* ((char *)bbp + CACHE_LINE_SIZE*4));
43     #   endif
44     #   if NODE_LINE_NUM >= 6
45     pfld (* ((char *)bbp + CACHE_LINE_SIZE*5));
46     #   endif
47     #   if NODE_LINE_NUM >= 7
48     pfld (* ((char *)bbp + CACHE_LINE_SIZE*6));
49     #   endif
50     #   if NODE_LINE_NUM >= 8
51     pfld (* ((char *)bbp + CACHE_LINE_SIZE*7));
52     #   endif
53     #   if NODE_LINE_NUM >= 9
54     pfld (* ((char *)bbp + CACHE_LINE_SIZE*8));
55     #   endif
56     #   if NODE_LINE_NUM >= 10
57     pfld (* ((char *)bbp + CACHE_LINE_SIZE*9));
58     #   endif
59     #   if NODE_LINE_NUM >= 11
60     pfld (* ((char *)bbp + CACHE_LINE_SIZE*10));
61     #   endif
62     #   if NODE_LINE_NUM >= 12
63     pfld (* ((char *)bbp + CACHE_LINE_SIZE*11));
64     #   endif
65     #   if NODE_LINE_NUM >= 13
66     pfld (* ((char *)bbp + CACHE_LINE_SIZE*12));
67     #   endif
68     #   if NODE_LINE_NUM >= 14
69     pfld (* ((char *)bbp + CACHE_LINE_SIZE*13));
70     #   endif
71     #   if NODE_LINE_NUM >= 15
72     pfld (* ((char *)bbp + CACHE_LINE_SIZE*14));
73     #   endif
74     #   if NODE_LINE_NUM >= 16
75     pfld (* ((char *)bbp + CACHE_LINE_SIZE*15));
76     #   endif
77     #   if NODE_LINE_NUM >= 17
78     pfld (* ((char *)bbp + CACHE_LINE_SIZE*16));
79     #   endif
80     #   if NODE_LINE_NUM >= 18
81     pfld (* ((char *)bbp + CACHE_LINE_SIZE*17));
82     #   endif
83     #   if NODE_LINE_NUM >= 19
84     pfld (* ((char *)bbp + CACHE_LINE_SIZE*18));
85     #   endif
86     #   if NODE_LINE_NUM >= 20
87     pfld (* ((char *)bbp + CACHE_LINE_SIZE*19));
88     #   endif
89     #   if NODE_LINE_NUM >= 21
90     pfld (* ((char *)bbp + CACHE_LINE_SIZE*20));
91     #   endif
92     #   if NODE_LINE_NUM >= 22
93     pfld (* ((char *)bbp + CACHE_LINE_SIZE*21));
94     #   endif
95     #   if NODE_LINE_NUM >= 23
96     pfld (* ((char *)bbp + CACHE_LINE_SIZE*22));

```

```

97 #      endif
98 #      if NODE_LINE_NUM >= 24
99 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*23));
100 #      endif
101 #      if NODE_LINE_NUM >= 25
102 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*24));
103 #      endif
104 #      if NODE_LINE_NUM >= 26
105 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*25));
106 #      endif
107 #      if NODE_LINE_NUM >= 27
108 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*26));
109 #      endif
110 #      if NODE_LINE_NUM >= 28
111 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*27));
112 #      endif
113 #      if NODE_LINE_NUM >= 29
114 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*28));
115 #      endif
116 #      if NODE_LINE_NUM >= 30
117 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*29));
118 #      endif
119 #      if NODE_LINE_NUM >= 31
120 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*30));
121 #      endif
122 #      if NODE_LINE_NUM >= 32
123 #      pfld (* ((char *)bbp + CACHE_LINE_SIZE*31));
124 #      endif
125 #      if NODE_LINE_NUM >= 33
126 #      error "NODE_LINE_NUM must be <= 32!"
127 #      endif
128 }
129
130 static void inline NODE_PREF_ST(register void *bbp)
131 {
132     pfst (* ((char *)bbp));
133 #     if NODE_LINE_NUM >= 2
134 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE));
135 #     endif
136 #     if NODE_LINE_NUM >= 3
137 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*2));
138 #     endif
139 #     if NODE_LINE_NUM >= 4
140 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*3));
141 #     endif
142 #     if NODE_LINE_NUM >= 5
143 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*4));
144 #     endif
145 #     if NODE_LINE_NUM >= 6
146 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*5));
147 #     endif
148 #     if NODE_LINE_NUM >= 7
149 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*6));
150 #     endif
151 #     if NODE_LINE_NUM >= 8
152 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*7));
153 #     endif
154 #     if NODE_LINE_NUM >= 9
155 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*8));
156 #     endif
157 #     if NODE_LINE_NUM >= 10
158 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*9));
159 #     endif
160 #     if NODE_LINE_NUM >= 11
161 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*10));
162 #     endif
163 #     if NODE_LINE_NUM >= 12
164 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*11));
165 #     endif
166 #     if NODE_LINE_NUM >= 13
167 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*12));
168 #     endif
169 #     if NODE_LINE_NUM >= 14
170 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*13));
171 #     endif
172 #     if NODE_LINE_NUM >= 15
173 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*14));
174 #     endif
175 #     if NODE_LINE_NUM >= 16
176 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*15));
177 #     endif
178 #     if NODE_LINE_NUM >= 17
179 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*16));
180 #     endif
181 #     if NODE_LINE_NUM >= 18
182 #     pfst (* ((char *)bbp + CACHE_LINE_SIZE*17));
183 #     endif

```

```

184 #   if NODE_LINE_NUM >= 19
185     pfst (* ((char *)bbp + CACHE_LINE_SIZE*18));
186 #   endif
187 #   if NODE_LINE_NUM >= 20
188     pfst (* ((char *)bbp + CACHE_LINE_SIZE*19));
189 #   endif
190 #   if NODE_LINE_NUM >= 21
191     pfst (* ((char *)bbp + CACHE_LINE_SIZE*20));
192 #   endif
193 #   if NODE_LINE_NUM >= 22
194     pfst (* ((char *)bbp + CACHE_LINE_SIZE*21));
195 #   endif
196 #   if NODE_LINE_NUM >= 23
197     pfst (* ((char *)bbp + CACHE_LINE_SIZE*22));
198 #   endif
199 #   if NODE_LINE_NUM >= 24
200     pfst (* ((char *)bbp + CACHE_LINE_SIZE*23));
201 #   endif
202 #   if NODE_LINE_NUM >= 25
203     pfst (* ((char *)bbp + CACHE_LINE_SIZE*24));
204 #   endif
205 #   if NODE_LINE_NUM >= 26
206     pfst (* ((char *)bbp + CACHE_LINE_SIZE*25));
207 #   endif
208 #   if NODE_LINE_NUM >= 27
209     pfst (* ((char *)bbp + CACHE_LINE_SIZE*26));
210 #   endif
211 #   if NODE_LINE_NUM >= 28
212     pfst (* ((char *)bbp + CACHE_LINE_SIZE*27));
213 #   endif
214 #   if NODE_LINE_NUM >= 29
215     pfst (* ((char *)bbp + CACHE_LINE_SIZE*28));
216 #   endif
217 #   if NODE_LINE_NUM >= 30
218     pfst (* ((char *)bbp + CACHE_LINE_SIZE*29));
219 #   endif
220 #   if NODE_LINE_NUM >= 31
221     pfst (* ((char *)bbp + CACHE_LINE_SIZE*30));
222 #   endif
223 #   if NODE_LINE_NUM >= 32
224     pfst (* ((char *)bbp + CACHE_LINE_SIZE*31));
225 #   endif
226 #   if NODE_LINE_NUM >= 33
227 #       error "NODE_LINE_NUM must be <= 32!"
228 #   endif
229 }
230
231 #define LEAF_PREF          NODE_PREF
232 #define LEAF_PREF_ST      NODE_PREF_ST
233
234 // prefetch AREA_LINE_NUM cache lines starting from &(area[i])
235 static void inline AREA_PREF(void * area[], int i, int total)
236 {
237     if ((i) + AREA_LINE_NUM*CACHE_LINE_SIZE/ITEM_SIZE <= (total)) {
238         register char *p = (char *)&(area[i]);
239         pfst (* p);
240 #       if AREA_LINE_NUM >= 2
241         pfst (* (p + CACHE_LINE_SIZE));
242 #       endif
243 #       if AREA_LINE_NUM >= 3
244         pfst (* (p + CACHE_LINE_SIZE*2));
245 #       endif
246 #       if AREA_LINE_NUM >= 4
247         pfst (* (p + CACHE_LINE_SIZE*3));
248 #       endif
249 #       if AREA_LINE_NUM >= 5
250         pfst (* (p + CACHE_LINE_SIZE*4));
251 #       endif
252 #       if AREA_LINE_NUM >= 6
253         pfst (* (p + CACHE_LINE_SIZE*5));
254 #       endif
255 #       if AREA_LINE_NUM >= 7
256         pfst (* (p + CACHE_LINE_SIZE*6));
257 #       endif
258 #       if AREA_LINE_NUM >= 8
259         pfst (* (p + CACHE_LINE_SIZE*7));
260 #       endif
261 #       if AREA_LINE_NUM >= 9
262         pfst (* (p + CACHE_LINE_SIZE*8));
263 #       endif
264 #       if AREA_LINE_NUM >= 10
265         pfst (* (p + CACHE_LINE_SIZE*9));
266 #       endif
267 #       if AREA_LINE_NUM >= 11
268         pfst (* (p + CACHE_LINE_SIZE*10));
269 #       endif
270 #       if AREA_LINE_NUM >= 12

```

```

271     pfst (* (p + CACHE_LINE_SIZE*11));
272 #     endif
273 #     if AREA_LINE_NUM >= 13
274     pfst (* (p + CACHE_LINE_SIZE*12));
275 #     endif
276 #     if AREA_LINE_NUM >= 14
277     pfst (* (p + CACHE_LINE_SIZE*13));
278 #     endif
279 #     if AREA_LINE_NUM >= 15
280     pfst (* (p + CACHE_LINE_SIZE*14));
281 #     endif
282 #     if AREA_LINE_NUM >= 16
283     pfst (* (p + CACHE_LINE_SIZE*15));
284 #     endif
285 #     if AREA_LINE_NUM >= 17
286 #         error "AREA_LINE_NUM must be <= 16!"
287 #     endif
288     }
289 }
290
291 /* ----- */
292 #endif /* _PBTREE_NODEPREF_H */

```

5.35 system/pbtree.cc File Reference

```
#include "pbtree.h"
```

Macros

- #define `LEFT_KEY_NUM` $((\text{LEAF_KEY_NUM}+1)/2)$
- #define `LEFT_KEY_NUM` $((\text{NON_LEAF_KEY_NUM})/2)$
- #define `RIGHT_KEY_NUM` $((\text{LEAF_KEY_NUM}+1) - \text{LEFT_KEY_NUM})$
- #define `RIGHT_KEY_NUM` $((\text{NON_LEAF_KEY_NUM}) - \text{LEFT_KEY_NUM})$

5.35.1 Detailed Description

Author

Shimin Chen shimin.chen@gmail.com

Version

1.0

5.35.2 LICENSE

TBD

5.35.3 DESCRIPTION

The pbtree class implements prefetching B+-tree (without jump pointer arrays). with NO_PREFETCH, this becomes an B+-tree implementation.

5.35.4 Macro Definition Documentation

5.35.4.1 LEFT_KEY_NUM [1/2]

```
#define LEFT_KEY_NUM ((LEAF_KEY_NUM+1)/2)
```

5.35.4.2 LEFT_KEY_NUM [2/2]

```
#define LEFT_KEY_NUM ((NON_LEAF_KEY_NUM)/2)
```

5.35.4.3 RIGHT_KEY_NUM [1/2]

```
#define RIGHT_KEY_NUM ((LEAF_KEY_NUM+1) - LEFT_KEY_NUM)
```

5.35.4.4 RIGHT_KEY_NUM [2/2]

```
#define RIGHT_KEY_NUM ((NON_LEAF_KEY_NUM) - LEFT_KEY_NUM)
```

5.36 system/pbtree.d File Reference

5.37 system/pbtree.h File Reference

```
#include <assert.h>
#include <stdlib.h>
#include "mymemory.h"
#include "nodepref.h"
#include <stdio.h>
```

Classes

- class [bnode](#)
- class [pbtree](#)
- class [Pbtree](#)
- class [Pointer8B](#)

Macros

- #define `BKEY_NUM` (`BNODE_SIZE`/`(KEY_SIZE+POINTER8B_SIZE)` - 1)
- #define `bleaf` `bnode`
- #define `bnext`(`ptr`) (((`bleaf` *)(`ptr`))->`ch`(0))
- #define `bnum`(`ptr`) (((`bleaf` *)(`ptr`))->`k`(0))
- #define `KEY_SIZE` 8
- #define `LEAF_KEY_NUM` `BKEY_NUM`
- #define `MAX_KEY` ((`key_type`)(0x7fffffffffffffff))
- #define `MIN_KEY` ((`key_type`)(0x8000000000000000))
- #define `NON_LEAF_KEY_NUM` `BKEY_NUM`
- #define `POINTER8B_SIZE` 8
- #define `POINTER_SIZE` 8

Typedefs

- typedef long long `key_type`

5.37.1 Detailed Description

Author

Shimin Chen shimin.chen@gmail.com

Version

1.0

5.37.2 LICENSE

TBD

5.37.3 DESCRIPTION

The pbtree class implements prefetching B+-tree (without jump pointer arrays). with NO_PREFETCH, this becomes an B+-tree implementation.

5.37.4 Macro Definition Documentation

5.37.4.1 BKEY_NUM

```
#define BKEY_NUM (BNODE_SIZE/(KEY_SIZE+POINTER8B_SIZE) - 1)
```

5.37.4.2 bleaf

```
#define bleaf bnode
```

5.37.4.3 bnext

```
#define bnext(  
    ptr ) (((bleaf *) (ptr))->ch(0))
```

5.37.4.4 bnum

```
#define bnum(  
    ptr ) (((bleaf *) (ptr))->k(0))
```

5.37.4.5 KEY_SIZE

```
#define KEY_SIZE 8
```

5.37.4.6 LEAF_KEY_NUM

```
#define LEAF_KEY_NUM BKEY_NUM
```

5.37.4.7 MAX_KEY

```
#define MAX_KEY (((key_type) (0x7fffffffffffffff))
```

5.37.4.8 MIN_KEY

```
#define MIN_KEY (((key_type) (0x8000000000000000))
```


5.37.4.9 NON_LEAF_KEY_NUM

```
#define NON_LEAF_KEY_NUM BKEY_NUM
```

5.37.4.10 POINTER8B_SIZE

```
#define POINTER8B_SIZE 8
```

5.37.4.11 POINTER_SIZE

```
#define POINTER_SIZE 8
```

5.37.5 Typedef Documentation

5.37.5.1 key_type

```
typedef long long key_type
```

5.38 pbtree.h

[Go to the documentation of this file.](#)

```
1
16 /* B+ tree:
17
18     non_leaf nodes:
19     - structure:
20         k[0] k[1] k[2] ... k[N-1]
21         ch[0] ch[1] ch[2] ... ch[N-1]
22
23     - property
24         k[0] is the actual number of keys in the node
25         k[1] < k[2] < ... < k[N-1]
26         ch[] are pointers to subtrees.
27
28         can hold at most (N-1) keys and N pointers.
29
30     - implementation specific:
31         assert(k[0]>=1)
32
33     leaf nodes:
34     - structure:
35         k[0]    k[1]    k[2] ... k[N-1]
36         ch[0]   ch[1] ch[2]... ch[N-1]
37
38     - property:
39         (k[i], ch[i]) 1<=i<=N-1, are index entries
40         k[0] is the actual number of entries in the node.
41         ch[0] is leaf sibling pointer.
42
43         can hold at most (N-1) index entries.
44
45     - implementation specific:
46         assert(k[0]>=1)
```

```

47  */
48
49 #ifndef _PBTREE_PBTREE_H
50 #define _PBTREE_PBTREE_H
51
52 #include <assert.h>
53 #include <stdlib.h>
54 #include "mymemory.h"
55
56 #include "nodepref.h"
57 /* ----- */
58 // #include "tree.h" based on x86_64 prefetch
59 //-----from tree.h-----
60 #include <stdio.h>
61
62 typedef long long key_type;
63 #define KEY_SIZE 8
64 #define POINTER_SIZE 8
65 #define POINTER8B_SIZE 8
66 #define MAX_KEY ((key_type) (0x7fffffffffffffff))
67 #define MIN_KEY ((key_type) (0x8000000000000000))
68
69 /* ----- */
70
71 // BKEY_NUM = N-1
72 #define BKEY_NUM (BNODE_SIZE/(KEY_SIZE+POINTER8B_SIZE) - 1)
73 #define NON_LEAF_KEY_NUM BKEY_NUM
74 #define LEAF_KEY_NUM BKEY_NUM
75
76 /* ----- */
77 class Pointer8B {
78 public:
79     // unsigned int num;
80     unsigned long long value;
81
82 public:
83     Pointer8B & operator= (const void * ptr)
84     {
85         value= (unsigned long long)ptr;
86         return *this;
87     }
88
89     Pointer8B & operator= (const Pointer8B & p)
90     {
91         value= p.value;
92         return *this;
93     }
94
95     operator void*() { return (void *)value; }
96     operator char*() { return (char *)value; }
97     operator struct bnode *() { return (struct bnode *)value; }
98     operator struct bleaf *() { return (struct bleaf *)value; }
99     operator unsigned long long () { return value; }
100
101     void print(void) {
102         if (value & (1UL<<63)) {
103             void *p = (void*)(value & ((1UL<<63)-1));
104             int num = *(int*) p;
105             void** vptrs = (void**)((char*)p+2*sizeof(int));
106             printf ("[");
107             for (int ii=0; ii< num-1; ii++)
108                 printf ("%p,", vptrs[ii]);
109             printf ("%p]\n", vptrs[num-1]);
110         }
111         else {
112             printf("[%p]\n", (void*)value);
113         }
114     }
115
116 }; // Pointer8B
117
118 #ifdef UNIFIED_IDX_ENTRY
119 typedef struct IdxEntry{
120     key_type k;
121     Pointer8B ch;
122 } IdxEntry;
123
124 class bnode {
125 public:
126     IdxEntry ent[BKEY_NUM + 1];
127 public:
128     key_type & k(int idx) { return ent[idx].k; }
129     Pointer8B & ch(int idx) { return ent[idx].ch; }
130     char * chEndAddr(int idx)
131     {return (char *)&(ent[idx].ch)+sizeof(Pointer8B)-1;}
132 }; // bnode
133 #else

```

```

134 class bnode {
135 public:
136     key_type    key[BKEY_NUM + 1];
137     Pointer8B   child[BKEY_NUM + 1];
138 public:
139     key_type & k(int idx) { return key[idx]; }
140     Pointer8B & ch(int idx) { return child[idx]; }
141     char * chEndAddr(int idx)
142     { return (char *)&(child[idx])+sizeof(Pointer8B)-1; }
143 }; // bnode
144 #endif
145
146 #define bleaf bnode
147 #define bnum(ptr) ((bleaf *) (ptr))->k(0)
148 #define bnext(ptr) ((bleaf *) (ptr))->ch(0)
149
150 /* ----- */
151 class pbtree {
152 private: // tree nodes definition
153
154     bnode * get_node ()
155     {
156         char *p = NULL;
157         int64_t sz = g_memory.alloc (p, BNODE_SIZE);
158         return sz == BNODE_SIZE? (bnode*)p: NULL;
159     }
160
161     void delete_node (void *p)
162     {
163         g_memory.free((char*)p, BNODE_SIZE);
164     }
165
166 public: // root and level
167
168     bnode * tree_root;
169     int root_level; // leaf: level 0, parent of leaf: level 1
170
171 private:
172     /* don't need to be prefetched!!! */
173     bnode * parray[32];
174     int ppos[32];
175
176 public:
177
178     int init (void);
179     int shut (void);
180     // given a search key, perform the search operation
181     // return the leaf node pointer and the position with leaf node
182     void * lookup (key_type key, int *pos);
183
184     void * get_recptr (void *p, int pos)
185     {
186         if (p == NULL) return NULL;
187         return ((bleaf *)p)->ch(pos);
188     }
189
190     key_type get_key (void *p, int pos)
191     {
192         return ((bleaf *)p)->k(pos);
193     }
194
195     void** get_recptr_sp (void *p, int pos)
196     {
197         if (p == NULL) return NULL;
198         return (void**)&(((bleaf *)p)->ch(pos));
199     }
200
201     // insert (key, ptr)
202     void insert (key_type key, void *ptr);
203
204     // delete key
205     void del (key_type key);
206
207     // range scan
208     // Input: *p is the starting leaf node
209     //         *spos is the starting position
210     //         endkey is the end key
211     //         *num is the buffer size of area[]
212     // Operation: store record pointers into area[]
213     // Output:
214     //         return 0: endkey is not reached
215     //                 should call again with (*p, *spos)
216     //                 *num is the number of pointers retrieved into area[]
217     //         return non-0: endkey is reached
218     //                 *num is the number of pointers retrieved into area[]
219     int scan (void **p, int *spos, key_type startkey, key_type endkey,
220              void * area[], int *num);

```

```

221
222 private:
223     void print (bnode *p, int level);
224     void check (bnode *p, int level, key_type *start, key_type *end, void **ptr);
225
226 public:
227     void print ()
228     {
229         printf("Printing pbtree\n");
230         print (tree_root, root_level);
231     }
232
233     void check (key_type *start, key_type *end)
234     {
235         assert (sizeof(bnode)==BNODE_SIZE);
236         assert (sizeof(bleaf)==BNODE_SIZE);
237         void * ptr = NULL;
238         check (tree_root, root_level, start, end, &ptr);
239     }
240
241     int level () {return root_level;}
242
243 public:
244     void save (char *filename)
245     {
246         FILE *fp = fopen (filename, "w");
247         if (fp == NULL) {
248             perror (filename); exit (1);
249         }
250         fprintf (fp, "tree_root=%p\n", tree_root);
251         fprintf (fp, "root_level=%d\n\n", root_level);
252         fclose (fp);
253     }
254
255     void load (char *filename)
256     {
257         FILE *fp = fopen (filename, "r");
258         if (fp == NULL) {
259             perror (filename); exit (1);
260         }
261         if ((fscanf (fp, "tree_root=%p\n", &tree_root) != 1) ||
262             (fscanf (fp, "root_level=%d\n", &root_level) != 1)) {
263
264             fprintf (stderr, "%s format error!\n", filename);
265             exit (1);
266         }
267         fclose (fp);
268     }
269
270 }; // pbtree
271
272 /* ----- */
273 class Pbtree { // support duplicated elements
274 private:
275     pbtree p_pbtree;
276     char* p_free_header[16]; // minimum size is 32 Byte
277
278 public:
279     void init (void);
280     bool insert (key_type key, void* ptr);
281     bool del (key_type key, void* ptr);
282     void* lookup (key_type);
283     int get_recptr (void* p, void* match[], int capacity, int &pos);
284
285     void* lookup_s (key_type key, int *pos);
286     int scan (void **p, int *spos, key_type startkey, key_type endkey,
287              void * area[], int *num);
288
289     void shut (void);
290     void print (void);
291
292 public:
293     bool allocate (char *&p,int leve) {
294         if (p_free_header[leve]) {
295             p = p_free_header[leve];
296             p_free_header[leve] = *(char**)p_free_header[leve];
297             *(int*)(p) = 0;
298             *(int*)(p+sizeof(int)) = leve2cap (leve);
299             return true;
300         }
301         else {
302             int64_t size = leve2size (leve);
303             int64_t sz = g_memory.alloc (p, size);
304             if (sz != size) {
305                 printf ("[Pbtree][ERROR][allocate]: g_memory is not enough! 296\n");
306                 return false;
307             }

```

```

308         *(int*) (p) = 0;
309         *(int*) (p+sizeof(int)) = leve2cap (leve);
310         return true;
311     }
312     return false;
313 }
314 bool free (char *p, int leve) {
315     *(char**) (p) = p_free_header[leve];
316     p_free_header[leve] = p;
317     return true;
318 }
319 int cap2leve(int cap) {
320     int t = ((cap+1)>>2);
321     int leve = 0;
322     while (t>1) {
323         leve ++;
324         t = t>1;
325     }
326     return leve;
327 }
328 int leve2cap(int leve) {
329     return (1<<(leve+2))-1;
330 }
331 int leve2size (int leve) {
332     return 1<<(leve+5);
333 }
334 int size2leve (int size) {
335     int leve = 0;
336     int sz = size;
337     while (sz > (1<<5)) {
338         leve ++;
339         sz = sz >> 1;
340     }
341     return leve;
342 }
343 };
344
345 /* ----- */
346 #endif /* _PBTREE_H */

```

5.39 system/pbtreeindex.cc File Reference

```
#include "pbtreeindex.h"
```

5.40 system/pbtreeindex.d File Reference

5.41 system/pbtreeindex.h File Reference

```
#include "schema.h"
#include "pbtree.h"
```

Classes

- class [PbtreeIndex](#)
- struct [PbtreeInfo](#)

Macros

- #define [PBTREEINFO_CAPICITY](#) (16)

5.41.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.41.2 DESCRIPTION

pbtrees index, here we support ID data type(INT8/16/32/64/DATE/TIME/DATETIME) the value accessed at Element is the pointer of related record. this implementation permits duplicated key with different value to be inserted, but not element with same key and value del operation will only del the first one which meet requirement, if you want delete all, you can call it many times till it returns false

@basic usage:

for each insert,del,look,scan, this file provides 2 same name method to handle 2 type data format you can use (1) for lookup, you should all use set_Is to set BptreeInfo with proper value, the first param is valid, leave the second to be NULL, BptreeInfo can help you iterately get values you need. (2) call lookup to get the value iterately. (3) scan you can scan to iterately get the value you need

5.41.3 Macro Definition Documentation

5.41.3.1 PBTRREEINFO_CAPACITY

```
#define PBTRREEINFO_CAPACITY (16)
```

5.42 pbtreesindex.h

[Go to the documentation of this file.](#)

```
1
23 #ifndef _PBTRREEINDEX_H
24 #define _PBTRREEINDEX_H
25
26 #include "schema.h"
27 #include "pbtrees.h"
28
29 // support INT, id type data
30 #define PBTRREEINFO_CAPACITY (16)
31
32 struct PbtreesInfo {
33     key_type left;
34     key_type right;
35     void *l_ptr;
36     void *s_ptr;
37     int s_pos;
38     int s_num;
39     int s_end;
40     void *area[PBTRREEINFO_CAPACITY];
41     void *result[PBTRREEINFO_CAPACITY];
42     int cr_area;
43     int cr_resu;
```

```

45     int    pos_resu;
46     int    le_resu;
47 };
48
49 class PbtreeIndex : public Index {
50 private:
51     Pbtree    pi_pbtree;
52     BasicType *pi_datatype;
53 public:
54     PbtreeIndex (int64_t pi_id, const char *i_name, Key &i_key)
55         :Index (pi_id, i_name, BPTREEINDEX, i_key)
56     {
57     }
58     bool init (void);
59     bool setIndexDTpye(BasicType * i_dt);
60     bool shut(void);
61
62     // data operator
63     bool insert(void *i_data, void *p_in);
64     bool set_ls(void *i_data1, void *i_data2, void *info);
65     bool lookup(void *i_data, void *info, void *&result);
66     bool scan (void *info, void *&result);
67     bool del(void *i_data, void *p_del);
68     void print(void);
69 };
70
71 #endif

```

5.43 system/rowtable.cc File Reference

```
#include "rowtable.h"
```

5.43.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.43.2 DESCRIPTION

rowtable implementation, this file implement all interface required by class table data space is managed by g_malloc memory, which decreases malloc overhead when create rowtable, I will make one more column to represent its validation. if delete a record, put down the label to set it "invalid" for index in this table, delete the entry inside index

5.44 system/rowtable.d File Reference

5.45 system/rowtable.h File Reference

```

#include "mymemory.h"
#include "schema.h"

```

Classes

- class [MStorage](#)
- class [RowTable](#)
- class [RPattern](#)

Variables

- [Memory g_memory](#)

5.45.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.45.2 DESCRIPTION

rowtable implementation, this file implement all interface required by class table data space is managed by g_memory, which decreases malloc overhead when create rowtable, I will make one more column to represent its validation. if delete a record, put down the label to set it "invalid" for index in this table, delete the entry inside index

basic usage: using rowtable interface surrounded by "//-----" will be enough for you

5.45.3 Variable Documentation

5.45.3.1 g_memory

[Memory](#) g_memory [extern]

5.46 rowtable.h

[Go to the documentation of this file.](#)

```

1
19 #ifndef _ROWTABLE_H
20 #define _ROWTABLE_H
21
22 #include "mymemory.h"
23 #include "schema.h"
24
25 extern Memory g_memory;
26
27 class RPattern {
28 private:
29     int64_t rp_colnum;
30     int64_t *rp_offset;
31     BasicType **rp_dtype;
32     char *rp_memory;
33     int64_t rp_mem_sz;
34     int64_t rp_current;
35     int64_t rp_row_sz;
36     char par[128 - 3 * sizeof(void *) - 4 * sizeof(int64_t)];
37 public:
38     bool init(int64_t col_num) {
39         rp_row_sz = 0L;
40         rp_current = 0L;
41         rp_colnum = col_num;
42         int64_t alloc_size =
43             rp_colnum * (sizeof(BasicType *) + sizeof(int64_t));
44         for (rp_mem_sz = 8L; rp_mem_sz < alloc_size;
45              rp_mem_sz = rp_mem_sz << 1);
46         alloc_size = g_memory.alloc(rp_memory, rp_mem_sz);
47         if (alloc_size != rp_mem_sz) {
48             printf("[RPattern][ERROR][init]: alloc memory error! -1\n");
49             return false;
50         }
51         rp_offset = (int64_t *) rp_memory;
52         rp_dtype = (BasicType **) (rp_memory + rp_mem_sz / 2);
53         return true;
54     }
55     bool addColumn(BasicType * col_type) {
56         if (rp_current >= rp_colnum)
57             return false;
58         rp_dtype[rp_current] = col_type;
59         rp_offset[rp_current] = rp_row_sz;
60         rp_row_sz += col_type->getTypeSize();
61         rp_current++;
62         return true;
63     }
64     int64_t getColumnOffset(int64_t col_rank) {
65         return col_rank < rp_colnum ? rp_offset[col_rank] : -1;
66     }
67     BasicType *getColumnType(int64_t col_rank) {
68         return col_rank < rp_colnum ? rp_dtype[col_rank] : NULL;
69     }
70     void reset(void) {
71         rp_row_sz = 0L;
72         rp_current = 0L;
73     }
74     void shut(void) {
75         g_memory.free(rp_memory, rp_mem_sz);
76     }
77     int64_t getRowSize(void) {
78         return rp_row_sz;
79     }
80     int64_t print(char *r_ptr) {
81         int64_t sz = 0;
82         for (int64_t ii = 0; ii < rp_colnum - 1; ii++) {
83             char buf[1024];
84             sz += rp_dtype[ii]->formatTxt(buf, r_ptr + rp_offset[ii]);
85             printf("%s\t", buf);
86         }
87         printf("%c", *(r_ptr + rp_row_sz - 1)); // label of validation
88         return sz;
89     }
90 }; // class RPattern
91
92 class MStorage {
93 private:
94     int64_t ms_record_size;
95     int64_t ms_record_num;
96     int64_t ms_record_max;
97     char *ms_memory;
98 }

```

```

141     char      *ms_memory_cur;
142     int64_t    ms_memory_size;
143     char      pad[128-4*sizeof(int64_t)-sizeof(void *)];
144 public:
145     bool init(int64_t record_size) {
146         ms_record_num = 0L;
147         ms_memory_size = 0L;
148         ms_record_max = 0L;
149         ms_record_size = record_size;
150         if(g_memory.allocTableAddr (ms_memory)) {
151             printf ("[Mstorage][ERROR][init]: allocTableAddr error!\n");
152             return false;
153         }
154         ms_memory_cur = ms_memory;
155         return true;
156     }
157     int64_t allocRow(char *&pointer) {
158         if (ms_record_num >= ms_record_max) {
159             if (!expand()) {
160                 printf ("[Mstorage][ERROR][allowRow]: expend error! -172\n");
161                 return -1;
162             }
163         }
164         pointer = ms_memory_cur;
165         ms_memory_cur += ms_record_size;
166         return ms_record_num++;
167     }
168     char *getRow(int64_t record_rank) {
169         if (record_rank >= ms_record_num) {
170             printf ("[Mstorage][ERROR][getRow]: record_rank exceed! -4\n");
171             return NULL;
172         }
173         return ms_memory + record_rank*ms_record_size;
174     }
175     void shut(void) {
176         for (char *mm= ms_memory; mm< ms_memory_cur; mm+= TABLE_MEMORY_ALLOC_INC) {
177             if (munmap (mm, TABLE_MEMORY_ALLOC_INC)) {
178                 printf ("[Mstorage][ERROR][shut]: munmap non-table error!-50\n");
179             }
180         }
181     }
182     int64_t getRecordNum(void) {
183         return ms_record_num;
184     }
185 private:
186     bool expand(void) {
187         if (ms_memory_size > TABLE_MEMORY_ALLOC_MAX) {
188             printf ("[Mstorage][ERROR][expand]: exceed maximux mmap fail! -218\n");
189             return false;
190         }
191         void* addr = mmap (ms_memory+ms_memory_size, TABLE_MEMORY_ALLOC_INC, PROT_READ|PROT_WRITE,
192             MAP_PRIVATE|MAP_ANONYMOUS|MAP_FIXED, -1, 0);
193         if (addr == MAP_FAILED) {
194             printf ("[Mstorage][ERROR][expand]: mmap fail! \n");
195             return false;
196         }
197         ms_memory_size += TABLE_MEMORY_ALLOC_INC;
198         ms_record_max = ms_memory_size / ms_record_size;
199         return true;
200     }
201 }; // class MStorage
202
203 class RowTable:public Table {
204 private:
205     RPattern r_pattern;
206     MStorage r_storage;
207 public:
208     RowTable(int64_t r_id, const char *r_name)
209         :Table(r_id, r_name,ROWTABLE) {
210     }
211
212     // schema operating method, if you call finish, you must not call init and add Column
213
214     bool init(void);
215     bool finish(void);
216     bool shut(void);
217
218     // data    operating method
219
220     //-----
221     // select
222     // get data by record_rank, mainly for OLAP to scan
223     bool selectCol(int64_t record_rank, int64_t column_rank, char *dest);
224     bool selectCols(int64_t record_rank, int64_t column_total,
225         int64_t * column_ranks, char *dest);
226     bool select(int64_t record_rank, char *dest);
227
228
229
230
231

```

```

294 // if you know the pointer of row by index, for OLTP to copy out the data to dest
303 bool selectCol(char *row_pointer, int64_t column_rank, char *dest);
313 bool selectCols(char *row_pointer, int64_t column_total,
314                 int64_t * column_ranks, char *dest);
322 bool select(char *row_pointer, char *dest);
323
324 // update
333 bool updateCol(char *row_pointer, int64_t column_rank, char *source);
342 bool updateCol(int64_t record_rank, int64_t column_rank, char *source);
352 bool updateCols(int64_t record_rank, int64_t column_total,
353                 int64_t * column_ranks, char *source);
363 bool updateCols(char *row_pointer, int64_t column_total,
364                 int64_t * column_ranks, char *source);
374 bool updateCols(int64_t record_rank, int64_t column_total,
375                 int64_t * column_ranks, char *source[]);
385 bool updateCols(char *row_pointer, int64_t column_total,
386                 int64_t * column_ranks, char *source[]);
387
388 // del
389
396 bool del(int64_t record_rank);
403 bool del(char *row_pointer);
404
405 // insert
406
413 bool insert(char *source);
420 bool insert(char *columns[]);
421
422 //-----
426 bool printData(void);
430 bool loadData(const char *filename);
434 RPattern & getRPattern(void) {
435     return r_pattern;
436 }
440 MStorage & getMStorage(void) {
441     return r_storage;
442 }
446 int64_t getRecordNum(void) {
447     return r_storage.getRecordNum();
448 }
455 void *getRecordPtr(int64_t row_rank) {
456     char *ptr = NULL;
457     return access(row_rank, ptr) ? ptr : NULL;
458 }
459
460 private:
468 bool access(int64_t record_rank, char *&pointer);
477 bool accessCol(int64_t record_rank, int64_t column_rank,
478                char *&pointer);
485 bool isValid(char *record_ptr) {
486     return *(record_ptr + r_pattern.getRowSize() - 1) ==
487            'Y' ? true : false;
488 }
495 bool invalid(char *record_ptr) {
496     if (*(record_ptr + r_pattern.getRowSize() - 1) == 'N')
497         return false;
498     *(record_ptr + r_pattern.getRowSize() - 1) = 'N';
499     return true;
500 }
501 }; // class RowTable
502
503 #endif

```

5.47 system/runaimdb.cc File Reference

```

#include "global.h"
#include "executor.h"
#include <stdio.h>
#include <stdlib.h>
#include <vector>

```

Functions

- int **load_data** (const char *tablename[], const char *data_dir, int number)

- int `load_schema` (const char *filename)
- int `main` (int argc, char *argv[])
- int `test` (void)
- int `testOne` (int which)

Variables

- int `print_flag` = false
- `SelectQuery queries` [22]
 -----*tpch test*-----
- const char * `table_name` []

5.47.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.47.2 DESCRIPTION

the main entrance of AIMDB

5.47.3 Function Documentation

5.47.3.1 `load_data()`

```
int load_data (
    const char * tablename[],
    const char * data_dir,
    int number )
```

load table data from txt files

Parameters

<i>tablename</i>	names of tables
<i>data_dir</i>	the dir of data files corresponding to the table, end with '/'
<i>number</i>	number of tables to load data

Return values

0	success
<0	faliure

naming rule of data files

each data file should be named "table_name.tab" and meet the following requirement

data format

- (1) split by one '\t', a row ends with '
- ' (2) no empty row

5.47.3.2 load_schema()

```
int load_schema (
    const char * filename )
```

load a database schema from a txt file.

Parameters

<i>filename</i>	name of schema file, the schema must meet the folowing condition
-----------------	--

Return values

0	success #retval <0 faliure
---	----------------------------

schema format (1) split by one '\t', a row ends with '
' (2) claim [Database](#), [Table](#), column, index in order (3) no empty row

5.47.3.3 main()

```
int main (
    int argc,
    char * argv[] )
```

5.47.3.4 test()

```
int test (
    void )
```

5.47.3.5 testOne()

```
int testOne (
    int which )
```

5.47.4 Variable Documentation

5.47.4.1 print_flag

```
int print_flag = false
```

5.47.4.2 querys

```
SelectQuery querys[22]
```

```
-----tpch test-----
```

5.47.4.3 table_name

```
const char* table_name[]
```

Initial value:

```
= {
    "part",
    "supplier",
    "partsupp",
    "customer",
    "nation",
    "lineitem",
    "region",
    "orders"
}
```

5.48 system/runaimdb.d File Reference

5.49 system/schema.h File Reference

```
#include <string.h>
#include <vector>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include "datatype.h"
```

Classes

- class [Column](#)
- class [Database](#)
- class [Index](#)
- class [Key](#)
- class [Object](#)
- class [Table](#)

Macros

- `#define` [OBJ_NAME_MAX](#) (128)

Enumerations

- enum [ColumnType](#) {
 [INVID_C](#) = 0 , [INT8](#) , [INT16](#) , [INT32](#) ,
 [INT64](#) , [FLOAT32](#) , [FLOAT64](#) , [CHARN](#) ,
 [DATE](#) , [TIME](#) , [DATETIME](#) , [MAXTYPE_C](#) }
- enum [IndexType](#) {
 [INVID_I](#) = 0 , [HASHINDEX](#) , [BPTREEINDEX](#) , [ARTTREEINDEX](#) ,
 [MAXTYPE_I](#) }
- enum [ObjectType](#) {
 [INVID_O](#) = 0 , [DATABASE](#) , [TABLE](#) , [COLUMN](#) ,
 [INDEX](#) , [MAXTYPE_O](#) }
- enum [TableType](#) { [INVID_T](#) = 0 , [ROWTABLE](#) , [COLTABLE](#) , [MAXTYPE_T](#) }

5.49.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.49.2 DESCRIPTION

this file defines the abstract class of four primary elements of database system, these abstract classes provide uniform interface for upper application

basic interface: init,finish,shut,select,insert,update,del,selectCol,lookup,scan

notice: for insert,del,update, input data requires to be processed by [BasicType](#) method formatBin, then call above function to actually put the into table example: char date[10] = 1970-01-01 [TypeDate](#) type; char buff[10]; type.format(buff, date); // in buff, it's stored as time_t with 4 Byte then, you can perform insert ()

5.49.3 Macro Definition Documentation

5.49.3.1 OBJ_NAME_MAX

```
#define OBJ_NAME_MAX (128)
```

5.49.4 Enumeration Type Documentation

5.49.4.1 ColumnType

enum [ColumnType](#)

an enum for column.

Enumerator

INVID_C	
INT8	int8
INT16	int16
INT32	int32
INT64	int64
FLOAT32	float32
FLOAT64	float64
CHARN	charn, fixed length string
DATE	days from 1970-01-01 till current DATE
TIME	seconds from 00:00:00 till current TIME
DATETIME	seconds from 1970-01-01 00:00:00 till current DATETIME
MAXTYPE_C	

5.49.4.2 IndexType

enum [IndexType](#)

an enum for [Index](#).

Enumerator

INVID_I	
---------	--

Enumerator

HASHINDEX	hash index
BPTREEINDEX	bptree index
ARTTREEINDEX	art tree index
MAXTYPE_I	

5.49.4.3 ObjectType

enum [ObjectType](#)

an enum for ObjectType label.

Enumerator

INVID_O	
DATABASE	database
TABLE	table
COLUMN	column
INDEX	index
MAXTYPE_O	

5.49.4.4 TableType

enum [TableType](#)

an enum for [Table](#).

Enumerator

INVID_T	
ROWTABLE	row table
COLTABLE	column table
MAXTYPE↔ _T	

5.50 schema.h

[Go to the documentation of this file.](#)

```

1
26 #ifndef _SCHEMA_H
27 #define _SCHEMA_H
28
29 #include <string.h>
30 #include <vector>
31 #include <stdint.h>
32 #include <stdio.h>
33 #include <stdlib.h>
34 #include "datatype.h"
35
36 enum ObjectType {
37     INVID_O = 0,
38     DATABASE,
39     TABLE,
40     COLUMN,
41     INDEX,
42     MAXTYPE_O
43 };
44
45 enum IndexType {
46     INVID_I = 0,
47     HASHINDEX,
48     BPTREEINDEX,
49     ARTTREEINDEX,
50     MAXTYPE_I
51 };
52
53 enum TableType {
54     INVID_T = 0,
55     ROWTABLE,
56     COLTABLE,
57     MAXTYPE_T
58 };
59
60 enum ColumnType {
61     INVID_C = 0,
62     INT8,
63     INT16,
64     INT32,
65     INT64,
66     FLOAT32,
67     FLOAT64,
68     CHARN,
69     DATE,
70     TIME,
71     DATETIME,
72     MAXTYPE_C
73 };
74
75 #define OBJ_NAME_MAX (128)
76
77 class Object {
78 private:
79     int64_t o_id;
80     ObjectType o_type;
81     char o_name[OBJ_NAME_MAX];
82 public:
83     Object(int64_t o_id, ObjectType o_type, const char *o_name) {
84         this->o_id = o_id;
85         this->o_type = o_type;
86         strncpy(this->o_name, o_name, OBJ_NAME_MAX - 1);
87         this->o_name[OBJ_NAME_MAX - 1] = '\0';
88     }
89     virtual bool shut (void) { return false; }
90     virtual void print(void) {
91         printf("Object- o_id: %ld o_type: %d, o_name: %s\n", o_id, o_type,
92             o_name);
93     }
94     int64_t getOid(void) {
95         return o_id;
96     }
97     ObjectType getOtype(void) {
98         return o_type;
99     }
100     char *getOname(void) {
101         return o_name;
102     }
103     bool changeName(char *o_name) {
104         if (strlen(o_name) >= OBJ_NAME_MAX - 1) {
105             printf
106                 ("[Object][ERROR][changeName]: o_name exceed length! -1\n");

```

```

137         return false;
138     }
139     strncpy(this->o_name, o_name, OBJ_NAME_MAX);
140     return true;
141 }
142 }; // class Object
143
144 class Column: public Object {
145 private:
146     ColumnType c_type;
147     int64_t c_size;
148     BaseType *c_datatype;
149     int64_t c_offset;
150 public:
151     Column(int64_t c_id, const char *c_name, ColumnType c_type, int64_t c_size = 0)
152         :Object(c_id, COLUMN, c_name)
153     {
154         this->c_type = c_type;
155         this->c_size = c_size;
156         this->c_datatype = NULL;
157         this->c_offset = 0;
158     }
159     virtual ~Column(void) {
160         delete c_datatype;
161     }
162     ColumnType getCType(void) {
163         return c_type;
164     }
165     int64_t getCSize(void) {
166         return c_size;
167     }
168     int64_t getCOffset(void) {
169         return c_offset;
170     }
171     int64_t setCOffset(int64_t offset) {
172         return this->c_offset = offset;
173     }
174     virtual bool init(void) {
175         switch (c_type) {
176             case INVID_C:
177                 printf("[Column][ERROR][init]: invid type! -1\n");
178                 return false;
179             case INT8:
180                 c_datatype = new TypeInt8();
181                 break;
182             case INT16:
183                 c_datatype = new TypeInt16();
184                 break;
185             case INT32:
186                 c_datatype = new TypeInt32();
187                 break;
188             case INT64:
189                 c_datatype = new TypeInt64();
190                 break;
191             case FLOAT32:
192                 c_datatype = new TypeFloat32();
193                 break;
194             case FLOAT64:
195                 c_datatype = new TypeFloat64();
196                 break;
197             case CHARN:
198                 c_datatype = new TypeCharN(c_size);
199                 break;
200             case DATE:
201                 c_datatype = new TypeDate();
202                 break;
203                 // days from 1970-01-01 till current DATE
204             case TIME:
205                 c_datatype = new TypeTime();
206                 break;
207                 // seconds from 00:00:00 till current TIME
208             case DATETIME:
209                 c_datatype = new TypeDateTime();
210                 break;
211                 // seconds from 1970-01-01 00:00:00 till current DATETIME
212             case MAXTYPE_C:
213                 printf("[Column][ERROR][init]: invid type! -1\n");
214                 return false;
215         }
216         c_size = c_datatype->getTypeSize();
217         return true;
218     }
219     virtual bool finish(void) {
220         return true;
221     }
222     virtual bool shut(void) {
223         delete c_datatype;
224         return true;
225     }
226     virtual void print(void) {

```

```

259         printf("Column- c_id: %ld c_type: %d c_size: %ld c_name: %s c_offset: %ld\n",
260                getOid(), c_type, c_size, getName(), getOffset());
261     }
262     BasicType *getDataType(void) {
263         return c_datatype;
264     }
265     // HACK, if you want to implement COLTABLE, then add code here, now is for ROWTABLE
266 }; // class Column
267
268 // Note: the table provides no key definition (primary key & foreign key)
269
270 class Table:public Object {
271 private:
272     TableType t_type;
273     std::vector < int64_t > t_columns;
274     std::vector < int64_t > t_index;
275 public:
276     TableType getTtype(void) {
277         return t_type;
278     }
279     virtual ~Table(void) {}
280     std::vector < int64_t > &getColumns(void) {
281         return t_columns;
282     }
283     std::vector < int64_t > &getIndexs(void) {
284         return t_index;
285     }
286     int64_t getColumnRank(int64_t c_id) {
287         return getRank(t_columns, c_id);
288     }
289     int64_t getIndexRank(int64_t i_id) {
290         return getRank(t_index, i_id);
291     }
292     int64_t getRank(std::vector < int64_t > &vec, int64_t id) {
293         for (unsigned int ii = 0; ii < vec.size(); ii++)
294             if (vec[ii] == id)
295                 return ii;
296         return -1;
297     }
298 public:
299     Table(int64_t t_id, const char *t_name, TableType t_type):Object(t_id, TABLE,
300         t_name)
301     {
302         this->t_type = t_type;
303     }
304     virtual void print(void) {
305         printf("Table- t_id: %ld t_type: %d t_name: %s columns: {\n",
306                getOid(), getTtype(), getName());
307         unsigned int ii = 0;
308         for (; ii < t_columns.size() - 1; ii++) {
309             printf("%ld, ", t_columns[ii]);
310         }
311         printf("%ld} index: {\n", t_columns[ii]);
312         if (t_index.size() > 0) {
313             for (ii = 0; ii < t_index.size() - 1; ii++) {
314                 printf("%ld, ", t_index[ii]);
315             }
316             printf("%ld", t_index[ii]);
317         }
318         printf("}\n");
319     }
320
321     // schema operating method, if you call finish, you must not call init and add Column
322
323     virtual bool init(void) {
324         return false;
325     }
326     virtual bool addColumn(int64_t column_id) {
327         t_columns.push_back(column_id);
328         return true;
329     }
330     virtual bool addIndex(int64_t index_id) {
331         t_index.push_back(index_id);
332         return true;
333     }
334     virtual bool finish(void) {
335         return false;
336     }
337     // call actual storage system, to actual inilizer
338     virtual bool shut(void) {
339         return false;
340     }
341
342     // data operating method
343
344     // select
345     // get data by record_rank, mainly for OLAP to sacn
346     virtual bool selectCol(int64_t record_rank, int64_t column_rank,

```

```

415         char *dest) {
416     return false;
417 }
427 virtual bool selectCols(int64_t record_rank, int64_t column_total,
428                         int64_t * column_ranks, char *dest) {
429     return false;
430 }
438 virtual bool select(int64_t record_rank, char *dest) {
439     return false;
440 }
441 // if you know the pointer of row by index, for OLTP to copy out the data to dest
450 virtual bool selectCol(char *row_pointer, int64_t column_rank,
451                       char *dest) {
452     return false;
453 }
463 virtual bool selectCols(char *row_pointer, int64_t column_total,
464                         int64_t * column_ranks, char *dest) {
465     return false;
466 }
474 virtual bool select(char *row_pointer, char *dest) {
475     return false;
476 }
477
478 // update
487 virtual bool updateCol(int64_t record_rank, int64_t column_rank,
488                        char *source) {
489     return false;
490 }
499 virtual bool updateCol(char *row_pointer, int64_t column_rank,
500                        char *source) {
501     return false;
502 }
503 /* update several column data.
504  * @param record_rank the n th row in the table storage
505  * @param column_total total number of columns to select
506  * @param column_ranks array of column_rank, column_rank is the n th column in table pattern
507  * @param source        buffer to store data to change for
508  * @retval true         success
509  * @retval false        failure
510  */
511 virtual bool updateCols(int64_t record_rank, int64_t column_total,
512                         int64_t * column_ranks, char *source) {
513     return false;
514 }
524 virtual bool updateCols(char *row_pointer, int64_t column_total,
525                         int64_t * column_ranks, char *source) {
526     return false;
527 }
537 virtual bool updateCols(int64_t record_rank, int64_t column_total,
538                         int64_t * column_ranks, char *source[]) {
539     return false;
540 }
550 virtual bool updateCols(char *row_pointer, int64_t column_total,
551                         int64_t * column_ranks, char *source[]) {
552     return false;
553 }
554
555 // del
562 virtual bool del(int64_t record_rank) {
563     return false;
564 }
571 virtual bool del(char *row_pointer) {
572     return false;
573 }
580 virtual bool del(char *columns[]) {
581     return false;
582 }
583 // this isn't support because SQL condition
584
585 // insert
591 virtual bool insert(char *source) {
592     return false;
593 }
594 // the insert data are arranged in a buffer(source), in the order of and
595 fixed-length pattern
600 virtual bool insert(char *columns[]) {
601     return false;
602 }
603 // the insert data are arranged in different space, and we know its
604 pointer
606 virtual int64_t getRecordNum(void) {
607     return -1;
608 }
612 virtual void *getRecordPtr(int64_t row_rank) {
613     return NULL;
614 }
618 virtual bool loadData(const char *filename) {
619     return false;
620 }
624 virtual bool printData(void) {

```

```

625         return false;
626     }
627 }; // class Table
628
629 class Database:public Object {
630 private:
631     std::vector < int64_t > d_table;
632 public:
633     Database(int64_t d_id, const char *d_name)
634         :Object(d_id, DATABASE,d_name) {
635     }
636     virtual ~Database(void) {}
637     virtual bool init(void) {
638         return false;
639     }
640     virtual bool addTable(int64_t table_id) {
641         d_table.push_back(table_id);
642         return true;
643     }
644     virtual bool finish(void) {
645         return false;
646     } // call actual storage system, to actual inilizer
647     virtual bool shut(void) {
648         return false;
649     }
650     virtual void print(void) {
651         printf("Database- d_id: %ld d_name: %s table: {" , getOid(),
652             getName());
653         unsigned int ii = 0;
654         for (; ii < d_table.size() - 1; ii++) {
655             printf("%ld," , d_table[ii]);
656         }
657         printf("%ld}\n", d_table[ii]);
658     }
659     std::vector < int64_t > &getTables(void) {
660         return d_table;
661     }
662     // insert
663     virtual bool insert(int64_t table_id, char *source) {
664         return false;
665     }
666     virtual bool insert(int64_t table_id, char *columns[]) {
667         return false;
668     }
669     virtual bool loadData(int64_t table_id, const char *filename) {
670         return false;
671     }
672 }; // class Database
673
674 class Key {
675 private:
676     std::vector < int64_t > key;
677 public:
678     Key(void) {
679     }
680     void set(std::vector < int64_t > &in_key) {
681         for (unsigned int ii = 0; ii < in_key.size(); ii++)
682             key.push_back(in_key[ii]);
683     }
684     bool contain(int64_t col_id) {
685         for (unsigned int ii = 0; ii < key.size(); ii++) {
686             if (key[ii] == col_id)
687                 return true;
688         }
689         return false;
690     }
691     Key & operator=(const Key & p) {
692         key.clear();
693         for (unsigned int ii = 0; ii < p.key.size(); ii++)
694             key.push_back(p.key[ii]);
695         return *this;
696     }
697     void print(void) {
698         printf("(");
699         unsigned int ii = 0;
700         for (; ii < key.size() - 1; ii++)
701             printf("%ld," , key[ii]);
702         printf("%ld)", key[ii]);
703     }
704     std::vector < int64_t > &getKey(void) {
705         return key;
706     }
707 }; // class Key
708
709 class Index:public Object {
710 protected:
711     IndexType i_type;

```

```

788     Key i_key;
789     int64_t i_t_id;
790 public:
791     Index(int64_t i_id, const char *i_name, IndexType i_type,
792           Key & i_key):Object(i_id, INDEX, i_name) {
793         this->i_type = i_type;
794         this->i_key = i_key;
795         this->i_t_id = 0;
796     }
797     virtual ~Index(void) {}
798     virtual bool init(void) {
799         return false;
800     }
801     virtual bool finish(void) {
802         return false;
803     }
804     virtual bool shut(void) {
805         return false;
806     }
807     virtual bool insert(void *i_data, void *p_in) {
808         return false;
809     }
810     virtual bool insert(void *i_data[], void *p_in) {
811         return false;
812     }
813     virtual bool del(void *i_data) {
814         return false;
815     }
816     virtual bool del(void *i_data[]) {
817         return false;
818     }
819     virtual bool del(void *i_data, void *p_del) {
820         return false;
821     }
822     virtual bool del(void *i_data[], void *p_del) {
823         return false;
824     }
825     virtual bool update(void *i_data, void *p_in) {
826         return false;
827     }
828     virtual bool update(void *i_data[], void *p_in) {
829         return false;
830     }
831     // the following function can pull one by one
832     virtual bool set_ls(void *i_data1, void *i_data2, void *info) {
833         return false;
834     }
835     virtual bool set_ls(void *i_data1[], void *i_data2[], void *info) {
836         return false;
837     }
838     virtual bool lookup(void *i_data, void *&result) {
839         return false;
840     }
841     virtual bool lookup(void *i_data[], void *&result) {
842         return false;
843     }
844     virtual bool lookup(void *i_data, void *info, void *&result) {
845         return false;
846     }
847     virtual bool lookup(void *i_data[], void *info, void *&result) {
848         return false;
849     }
850     virtual bool scan (void *info, void *&result) {
851         return false;
852     }
853     virtual bool scan_1(void *i_left, void *info) {
854         return false;
855     }
856     virtual bool scan_1(void *i_left[], void *info) {
857         return false;
858     }
859     virtual bool scan_2(void *i_right, void *info, void *&result) {
860         return false;
861     }
862     virtual bool scan_2(void *i_right[], void *info, void *&result) {
863         return false;
864     }
865     virtual int64_t tranToInt64(void *i_data) {
866         return -1;
867     }
868     virtual int64_t tranToInt64(void *i_data[]) {
869         return -1;
870     }
871     virtual void print(void) {
872         printf("Index- i_id: %ld type: %d i_name: %s columns: ", getOid(),
873               i_type, getName());
874     }

```

```
1031         i_key.print();
1032         printf("\n");
1033     }
1037     IndexType getIType(void) {
1038         return i_type;
1039     }
1043     Key & getIKey(void) {
1044         return i_key;
1045     }
1049     virtual void setIndexTid (int64_t tid) {
1050         this->i_t_id = tid;
1051     }
1055     virtual int64_t getIndexTid (void) {
1056         return this->i_t_id;
1057     }
1058 }; // class Index
1059
1060 #endif
```


Index

- _Thread_local
 - errorlog.h, [260](#)
- ~BasicType
 - BasicType, [9](#)
- ~Column
 - Column, [21](#)
- ~Database
 - Database, [26](#)
- ~ErrorLog
 - ErrorLog, [31](#)
- ~Filter
 - Filter, [42](#)
- ~GroupbyAggre
 - GroupbyAggre, [56](#)
- ~HashJoin
 - HashJoin, [93](#)
- ~HashTable
 - HashTable, [98](#)
- ~Index
 - Index, [104](#)
- ~IndexJoin
 - IndexJoin, [117](#)
- ~IndexScan
 - IndexScan, [121](#)
- ~Join
 - Join, [125](#)
- ~Operator
 - Operator, [143](#)
- ~Orderby
 - Orderby, [146](#)
- ~Project
 - Project, [167](#)
- ~Scan
 - Scan, [198](#)
- ~Table
 - Table, [203](#)
- access
 - RowTable, [181](#)
- accessCol
 - RowTable, [181](#)
- add
 - HashTable, [98](#)
- addColumn
 - RPattern, [193](#)
 - Table, [204](#)
- addIndex
 - Table, [204](#)
- addIndexDTpye
 - HashIndex, [82](#)
- addTable
 - Database, [27](#)
- aggr_method
 - GroupbyAggre, [74](#)
- aggr_pos
 - GroupbyAggre, [74](#)
- aggr_type
 - GroupbyAggre, [74](#)
- AggreCondition, [7](#)
 - column_rank, [7](#)
 - method, [7](#)
- aggregate_method
 - RequestColumn, [174](#)
- AggregateMethod
 - executor.h, [268](#)
- alloc
 - Memory, [132](#)
- alloc_default
 - Memory, [133](#)
- allocate
 - HashTable, [99](#)
 - Pbtree, [150](#), [153](#)
- allocColBuf
 - executor.cc, [266](#)
- allocRow
 - MStorage, [137](#)
- allocTableAddr
 - Memory, [133](#)
- append
 - ResultTable, [175](#)
- area
 - PbtreeInfo, [163](#)
- AREA_LINE_NUM
 - nodepref.h, [292](#)
- arrayid
 - Orderby, [147](#)
- ARTTREEINDEX
 - schema.h, [317](#)
- avail
 - HashTable, [101](#)
- AVG
 - executor.h, [269](#)
- avg_table
 - GroupbyAggre, [74](#)
- avgFloat32
 - GroupbyAggre, [56](#)
- avgFloat64
 - GroupbyAggre, [56](#)
- avgInt16

- GroupbyAggre, 57
- avgInt32
 - GroupbyAggre, 57
- avgInt64
 - GroupbyAggre, 57
- avgInt8
 - GroupbyAggre, 58
- b_type_code
 - BasicType, 11
- b_type_size
 - BasicType, 11
- BasicType, 8
 - ~BasicType, 9
 - b_type_code, 11
 - b_type_size, 11
 - BasicType, 9
 - cmpEQ, 9
 - cmpGE, 9
 - cmpGT, 9
 - cmpLE, 10
 - cmpLT, 10
 - copy, 10
 - formatBin, 10
 - formatTxt, 11
 - getTypeCode, 11
 - getTypeSize, 11
- begin
 - HashTable, 101
- BKEY_NUM
 - pbtree.h, 299
- bleaf
 - pbtree.h, 299
- bnext
 - pbtree.h, 300
- bnode, 12
 - ch, 12
 - chEndAddr, 12
 - child, 13
 - k, 12
 - key, 13
- BNODE_POINTERS_NUM
 - global.h, 282
- BNODE_SIZE
 - nodepref.h, 293
- bnum
 - pbtree.h, 300
- BPTREEINDEX
 - schema.h, 317
- buf_for_child
 - Filter, 48
 - GroupbyAggre, 74
 - Project, 171
- buffer
 - ResultTable, 178
- buffer_from_father
 - Operator, 144
- buffer_size
 - ResultTable, 178
- c_datatype
 - Column, 23
- c_offset
 - Column, 23
- c_size
 - Column, 23
- c_type
 - Column, 23
- CACHE_LINE_SIZE
 - nodepref.h, 293
- cap2leve
 - Pbtree, 150, 153
- capacity
 - HashCell, 78
- Catalog, 13
 - cl_id_obj, 20
 - cl_name_obj, 20
 - createColumn, 14
 - createDatabase, 14
 - createIndex, 15
 - createTable, 15
 - getObjById, 16
 - getObjByName, 16
 - init, 16
 - initColumn, 16
 - initDatabase, 17
 - initIndex, 17
 - initTable, 18
 - obtainId, 18
 - print, 18
 - registerObj, 19
 - shut, 19
 - shutDatabase, 19
- catalog.cc
 - g_catalog, 248
- catalog.h
 - g_catalog, 249
- ch
 - bnode, 12
- changeName
 - Object, 140
- CHARN
 - schema.h, 316
- CHARN_TC
 - datatype.h, 251
- chEndAddr
 - bnode, 12
- child
 - bnode, 13
 - Filter, 48
 - GroupbyAggre, 75
 - Orderby, 148
 - Project, 171
- child_buf_size
 - Filter, 48
 - GroupbyAggre, 75
- child_buffer
 - Orderby, 148

- child_tuple_size
 - GroupbyAggre, [75](#)
- cl_id_obj
 - Catalog, [20](#)
- cl_name_obj
 - Catalog, [20](#)
- close
 - Executor, [38](#)
 - Filter, [43](#)
 - GroupbyAggre, [58](#)
 - HashJoin, [93](#)
 - IndexJoin, [118](#)
 - IndexScan, [121](#)
 - Join, [125](#)
 - Operator, [143](#)
 - Orderby, [146](#)
 - Project, [168](#)
 - Scan, [198](#)
- closeLog
 - ErrorLog, [31](#)
- cmp_func
 - Filter, [48](#)
- cmp_mtd
 - Filter, [49](#)
- cmp_table
 - Filter, [49](#)
- cmpEQ
 - BasicType, [9](#)
 - Filter, [43](#)
 - HashIndex, [82](#)
 - TypeCharN, [217](#)
 - TypeDate, [219](#)
 - TypeDateTime, [222](#)
 - TypeFloat32, [225](#)
 - TypeFloat64, [228](#)
 - TypeInt16, [231](#)
 - TypeInt32, [234](#)
 - TypeInt64, [237](#)
 - TypeInt8, [240](#)
 - Time, [243](#)
- cmpGE
 - BasicType, [9](#)
 - Filter, [44](#)
 - TypeCharN, [217](#)
 - TypeDate, [220](#)
 - TypeDateTime, [223](#)
 - TypeFloat32, [226](#)
 - TypeFloat64, [229](#)
 - TypeInt16, [232](#)
 - TypeInt32, [235](#)
 - TypeInt64, [238](#)
 - TypeInt8, [241](#)
 - Time, [244](#)
- cmpGT
 - BasicType, [9](#)
 - Filter, [44](#)
 - TypeCharN, [217](#)
 - TypeDate, [220](#)
- TypeDateTime, [223](#)
- TypeFloat32, [226](#)
- TypeFloat64, [229](#)
- TypeInt16, [232](#)
- TypeInt32, [235](#)
- TypeInt64, [238](#)
- TypeInt8, [241](#)
- Time, [244](#)
- cmpLE
 - BasicType, [10](#)
 - Filter, [44](#)
 - TypeCharN, [217](#)
 - TypeDate, [220](#)
 - TypeDateTime, [223](#)
 - TypeFloat32, [226](#)
 - TypeFloat64, [229](#)
 - TypeInt16, [232](#)
 - TypeInt32, [235](#)
 - TypeInt64, [238](#)
 - TypeInt8, [241](#)
 - Time, [244](#)
- cmpLT
 - BasicType, [10](#)
 - Filter, [45](#)
 - TypeCharN, [217](#)
 - TypeDate, [220](#)
 - TypeDateTime, [223](#)
 - TypeFloat32, [226](#)
 - TypeFloat64, [229](#)
 - TypeInt16, [232](#)
 - TypeInt32, [235](#)
 - TypeInt64, [238](#)
 - TypeInt8, [241](#)
 - Time, [244](#)
- cmpNE
 - Filter, [45](#)
- colid
 - Orderby, [148](#)
- coloff
 - Orderby, [148](#)
- colrank
 - Orderby, [148](#)
- COLTABLE
 - schema.h, [317](#)
- coltype
 - Orderby, [148](#)
- COLUMN
 - schema.h, [317](#)
- Column, [20](#)
 - ~Column, [21](#)
 - c_datatype, [23](#)
 - c_offset, [23](#)
 - c_size, [23](#)
 - c_type, [23](#)
 - Column, [21](#)
 - finish, [21](#)
 - getOffset, [22](#)
 - getCSize, [22](#)

- getCType, [22](#)
 - getDataType, [22](#)
 - init, [22](#)
 - print, [22](#)
 - setCoffset, [22](#)
 - shut, [23](#)
- column
 - Condition, [24](#)
- column_number
 - ResultTable, [178](#)
- column_rank
 - AggreCondition, [7](#)
- column_type
 - ResultTable, [178](#)
- ColumnType
 - schema.h, [316](#)
- compare
 - Condition, [24](#)
- CompareMethod
 - executor.h, [269](#)
- Condition, [24](#)
 - column, [24](#)
 - compare, [24](#)
 - value, [24](#)
- condition
 - Conditions, [25](#)
- condition_num
 - Conditions, [25](#)
- Conditions, [24](#)
 - condition, [25](#)
 - condition_num, [25](#)
- conditions
 - GroupbyAggre, [75](#)
- contain
 - Key, [130](#)
- copy
 - BasicType, [10](#)
 - TypeCharN, [218](#)
 - TypeDate, [221](#)
 - TypeDateTime, [224](#)
 - TypeFloat32, [227](#)
 - TypeFloat64, [230](#)
 - TypeInt16, [233](#)
 - TypeInt32, [236](#)
 - TypeInt64, [239](#)
 - TypeInt8, [242](#)
 - TypeTime, [245](#)
- COUNT
 - executor.h, [269](#)
- count
 - GrAggRecord, [51](#)
 - GroupbyAggre, [58](#)
- cr_area
 - PbtreeInfo, [163](#)
- cr_resu
 - PbtreeInfo, [163](#)
- createColumn
 - Catalog, [14](#)
- createDatabase
 - Catalog, [14](#)
- createIndex
 - Catalog, [15](#)
- createTable
 - Catalog, [15](#)
- current_key
 - IndexJoin, [119](#)
 - IndexScan, [123](#)
- current_query
 - Executor, [40](#)
- d_table
 - Database, [29](#)
- DATABASE
 - schema.h, [317](#)
- Database, [25](#)
 - ~Database, [26](#)
 - addTable, [27](#)
 - d_table, [29](#)
 - Database, [26](#)
 - finish, [27](#)
 - getTables, [27](#)
 - init, [27](#)
 - insert, [27](#), [28](#)
 - loadData, [28](#)
 - print, [29](#)
 - shut, [29](#)
- database_id
 - SelectQuery, [200](#)
- datatype.h
 - CHARN_TC, [251](#)
 - DATE_TC, [251](#)
 - DATETIME_TC, [251](#)
 - FLOAT32_TC, [251](#)
 - FLOAT64_TC, [251](#)
 - INT16_TC, [251](#)
 - INT32_TC, [251](#)
 - INT64_TC, [251](#)
 - INT8_TC, [251](#)
 - INVID_TC, [251](#)
 - MAXTYPE_TC, [251](#)
 - TIME_TC, [251](#)
 - TypeCode, [250](#)
- DATE
 - schema.h, [316](#)
- DATE_TC
 - datatype.h, [251](#)
- DATETIME
 - schema.h, [316](#)
- DATETIME_TC
 - datatype.h, [251](#)
- del
 - HashIndex, [83](#)
 - HashTable, [99](#)
 - Index, [104–106](#)
 - Pbtree, [150](#), [153](#)
 - PbtreeIndex, [157](#)
 - RowTable, [182](#)

- Table, [204](#), [205](#)
- dump
 - ResultTable, [176](#)
- easyAlloc
 - executor.cc, [266](#)
 - executor.h, [269](#)
- EL_ASSERT
 - errorlog.h, [260](#)
- EL_BAD_FILEID
 - errorlog.h, [260](#)
- el_bt_buffer
 - ErrorLog, [35](#)
- EL_DEBUG
 - errorlog.h, [260](#)
- el_demangle_buf
 - ErrorLog, [35](#)
- el_demangle_len
 - ErrorLog, [35](#)
- el_err_code
 - ErrorLog, [35](#)
- EL_ERRCODE
 - errorlog.h, [260](#)
- EL_ERRMSG
 - errorlog.h, [260](#)
- EL_ERROR
 - errorlog.h, [260](#)
- EL_ERROR_CODE
 - errorlog.h, [260](#)
- el_fp
 - ErrorLog, [35](#)
- EL_GET_FILEID
 - errorlog.h, [261](#)
- EL_GET_FILENAME
 - errorlog.h, [261](#)
- EL_GET_LINENO
 - errorlog.h, [261](#)
- EL_INFO
 - errorlog.h, [261](#)
- el_level
 - ErrorLog, [35](#)
- EL_LEVEL_COMPILE
 - errorlog.h, [261](#)
- el_level_name
 - ErrorLog, [36](#)
- el_lock
 - ErrorLog, [36](#)
- EL_LOG_DEBUG
 - errorlog.h, [261](#)
- EL_LOG_ERROR
 - errorlog.h, [261](#)
- EL_LOG_INFO
 - errorlog.h, [262](#)
- EL_LOG_SERIOUS
 - errorlog.h, [262](#)
- EL_LOG_WARN
 - errorlog.h, [262](#)
- el_logfile
 - ErrorLog, [36](#)
- el_msg_buf
 - ErrorLog, [36](#)
- el_msg_cap
 - ErrorLog, [36](#)
- el_msg_cur
 - ErrorLog, [36](#)
- el_name_2_id
 - ErrorLog, [37](#)
- EL_OK
 - errorlog.h, [262](#)
- EL_RESET
 - errorlog.h, [262](#)
- EL_SERIOUS
 - errorlog.h, [262](#)
- EL_src_file_name
 - errorlog.cc, [257](#)
 - errorlog.h, [263](#)
- el_thread_name
 - ErrorLog, [37](#)
- el_tloc
 - ErrorLog, [37](#)
- el_tm
 - ErrorLog, [37](#)
- EL_TOTAL_FILES
 - errorlog.cc, [257](#)
- EL_WARN
 - errorlog.h, [263](#)
- end
 - HashTable, [101](#)
- ent
 - HashCell, [78](#)
- ents
 - HashCell, [79](#)
- EQ
 - executor.h, [269](#)
- ErrorLog, [29](#)
 - ~ErrorLog, [31](#)
 - closeLog, [31](#)
 - el_bt_buffer, [35](#)
 - el_demangle_buf, [35](#)
 - el_demangle_len, [35](#)
 - el_err_code, [35](#)
 - el_fp, [35](#)
 - el_level, [35](#)
 - el_level_name, [36](#)
 - el_lock, [36](#)
 - el_logfile, [36](#)
 - el_msg_buf, [36](#)
 - el_msg_cap, [36](#)
 - el_msg_cur, [36](#)
 - el_name_2_id, [37](#)
 - el_thread_name, [37](#)
 - el_tloc, [37](#)
 - el_tm, [37](#)
 - ErrorLog, [31](#)
 - flushLog, [32](#)
 - getErrorCode, [32](#)
 - getErrorMsg, [32](#)

- getFuncNameGCC, 32
- id2Name, 33
- init, 33
- log, 33
- name2Id, 34
- reset, 34
- setLevel, 34
- errorlog.cc
 - EL_src_file_name, 257
 - EL_TOTAL_FILES, 257
 - thread_el, 257
- errorlog.h
 - _Thread_local, 260
 - EL_ASSERT, 260
 - EL_BAD_FILEID, 260
 - EL_DEBUG, 260
 - EL_ERRCODE, 260
 - EL_ERRMSG, 260
 - EL_ERROR, 260
 - EL_ERROR_CODE, 260
 - EL_GET_FILEID, 261
 - EL_GET_FILENAME, 261
 - EL_GET_LINENO, 261
 - EL_INFO, 261
 - EL_LEVEL_COMPILE, 261
 - EL_LOG_DEBUG, 261
 - EL_LOG_ERROR, 261
 - EL_LOG_INFO, 262
 - EL_LOG_SERIOUS, 262
 - EL_LOG_WARN, 262
 - EL_OK, 262
 - EL_RESET, 262
 - EL_SERIOUS, 262
 - EL_src_file_name, 263
 - EL_WARN, 263
 - thread_el, 263
- ESTIMATE_ERROR
 - hashtable.cc, 287
- estimated_duplicates_per_key
 - HashTable, 101
- estimated_num_distinct_keys
 - HashTable, 101
- exec
 - Executor, 38
- Executor, 37
 - close, 38
 - current_query, 40
 - exec, 38
 - findCol, 39
 - getRank, 39
 - planner, 40
 - root, 40
- executor.cc
 - allocColBuf, 266
 - easyAlloc, 266
 - getTupleSize, 267
- executor.h
 - AggregateMethod, 268
 - AVG, 269
 - CompareMethod, 269
 - COUNT, 269
 - easyAlloc, 269
 - EQ, 269
 - GE, 269
 - GT, 269
 - LE, 269
 - LINK, 269
 - LT, 269
 - MAX, 269
 - MAX_AM, 269
 - MAX_CM, 269
 - MIN, 269
 - NE, 269
 - NONE_AM, 269
 - NONE_CM, 269
 - SUM, 269
- expand
 - MStorage, 137
- filt_off
 - Filter, 49
- filt_pos
 - Filter, 49
- filt_type
 - Filter, 49
- Filter, 40
 - ~Filter, 42
 - buf_for_child, 48
 - child, 48
 - child_buf_size, 48
 - close, 43
 - cmp_func, 48
 - cmp_mtd, 49
 - cmp_table, 49
 - cmpEQ, 43
 - cmpGE, 44
 - cmpGT, 44
 - cmpLE, 44
 - cmpLT, 45
 - cmpNE, 45
 - filt_off, 49
 - filt_pos, 49
 - filt_type, 49
 - Filter, 42
 - getNext, 46
 - in_tuple_size, 49
 - initCmpFunc, 46
 - input_cid, 49
 - open, 46
 - setChild, 46
 - setColumn, 47
 - setFiltCond, 48
 - value, 49
- final_avg_table
 - GroupbyAggre, 75
- final_method
 - GroupbyAggre, 75

- final_sum_table
 - GroupbyAggre, [75](#)
- finalCount
 - GroupbyAggre, [59](#)
- finalFloat32Avg
 - GroupbyAggre, [59](#)
- finalFloat32Sum
 - GroupbyAggre, [59](#)
- finalFloat64Avg
 - GroupbyAggre, [60](#)
- finalFloat64Sum
 - GroupbyAggre, [60](#)
- finalInt16Sum
 - GroupbyAggre, [60](#)
- finalInt32Sum
 - GroupbyAggre, [61](#)
- finalInt64Sum
 - GroupbyAggre, [61](#)
- finalInt8Sum
 - GroupbyAggre, [61](#)
- finalIntAvg
 - GroupbyAggre, [62](#)
- findCol
 - Executor, [39](#)
- finish
 - Column, [21](#)
 - Database, [27](#)
 - HashIndex, [84](#)
 - Index, [106](#)
 - RowTable, [182](#)
 - Table, [205](#)
- FLOAT32
 - schema.h, [316](#)
- FLOAT32_TC
 - datatype.h, [251](#)
- FLOAT64
 - schema.h, [316](#)
- FLOAT64_TC
 - datatype.h, [251](#)
- flushLog
 - ErrorLog, [32](#)
- formatBin
 - BasicType, [10](#)
 - TypeCharN, [218](#)
 - TypeDate, [221](#)
 - TypeDateTime, [224](#)
 - TypeFloat32, [227](#)
 - TypeFloat64, [230](#)
 - TypeInt16, [233](#)
 - TypeInt32, [236](#)
 - TypeInt64, [239](#)
 - TypeInt8, [242](#)
 - TypeTime, [245](#)
- formatTxt
 - BasicType, [11](#)
 - TypeCharN, [218](#)
 - TypeDate, [221](#)
 - TypeDateTime, [224](#)
 - TypeFloat32, [227](#)
 - TypeFloat64, [230](#)
 - TypeInt16, [233](#)
 - TypeInt32, [236](#)
 - TypeInt64, [239](#)
 - TypeInt8, [242](#)
 - TypeTime, [245](#)
- free
 - HashTable, [99](#)
 - Memory, [133](#)
 - Pbtree, [150](#), [153](#)
- free_header
 - HashTable, [102](#)
- from
 - IndexScan, [123](#)
- from_number
 - SelectQuery, [200](#)
- from_table
 - SelectQuery, [201](#)
- g_catalog
 - catalog.cc, [248](#)
 - catalog.h, [249](#)
 - global.h, [283](#)
- g_memory
 - global.h, [283](#)
 - mymemory.cc, [289](#)
 - mymemory.h, [291](#)
 - rowtable.h, [308](#)
- gcc_pf_p3.h
 - pfid, [279](#)
 - pfidnta, [279](#)
 - pfst, [279](#)
 - pfstnta, [280](#)
 - prefetchnta, [280](#)
 - prefetcht0, [280](#)
 - prefetcht1, [280](#)
 - ptouch, [280](#)
- GE
 - executor.h, [269](#)
- get_recptr
 - Pbtree, [150](#), [154](#)
- getBuffer
 - Operator, [143](#)
- getCoffset
 - Column, [22](#)
- getColumn
 - Project, [168](#)
- getColumnOffset
 - RPattern, [193](#)
- getColumnRank
 - Table, [205](#)
- getColumns
 - Table, [206](#)
- getColumnType
 - RPattern, [194](#)
- getCSize
 - Column, [22](#)
- getCType

- Column, [22](#)
- getDataType
 - Column, [22](#)
- getErrorCode
 - ErrorLog, [32](#)
- getErrorMsg
 - ErrorLog, [32](#)
- getFuncNameGCC
 - ErrorLog, [32](#)
- getIKey
 - Index, [106](#)
- getIndexRank
 - Table, [206](#)
- getIndxs
 - Table, [206](#)
- getIndexTid
 - Index, [106](#)
- getIType
 - Index, [107](#)
- getKey
 - Key, [131](#)
- getLeftCol
 - Join, [125](#)
- getLeftOp
 - Join, [126](#)
- getLeftRank
 - Join, [126](#)
- getMStorage
 - RowTable, [183](#)
- getNext
 - Filter, [46](#)
 - GroupbyAggre, [62](#)
 - HashJoin, [94](#)
 - IndexJoin, [118](#)
 - IndexScan, [122](#)
 - Join, [126](#)
 - Operator, [143](#)
 - Orderby, [146](#)
 - Project, [169](#)
 - Scan, [198](#)
- getObjById
 - Catalog, [16](#)
- getObjByName
 - Catalog, [16](#)
- getOid
 - Object, [141](#)
- getOname
 - Object, [141](#)
- getOtype
 - Object, [141](#)
- getRank
 - Executor, [39](#)
 - Table, [207](#)
- getRC
 - ResultTable, [176](#)
- getRecordNum
 - MStorage, [137](#)
 - RowTable, [183](#)
- Table, [207](#)
- getRecordPtr
 - RowTable, [183](#)
- Table, [207](#)
- getRightCol
 - Join, [127](#)
- getRightOp
 - Join, [127](#)
- getRightRank
 - Join, [127](#)
- getRow
 - MStorage, [137](#)
- getRowSize
 - RPattern, [194](#)
- getRPattern
 - RowTable, [183](#)
- getSchema
 - Project, [169](#)
- getTables
 - Database, [27](#)
- getType
 - Table, [207](#)
- getTupleSize
 - executor.cc, [267](#)
- getTypeCode
 - BasicType, [11](#)
- getTypeSize
 - BasicType, [11](#)
- global.cc
 - global_init, [281](#)
 - global_shut, [282](#)
- global.h
 - BNODE_POINTERS_NUM, [282](#)
 - g_catalog, [283](#)
 - g_memory, [283](#)
 - global_init, [283](#)
 - GLOBAL_MEMORY_MINIMUM, [282](#)
 - GLOBAL_MEMORY_SIZE, [283](#)
 - global_shut, [283](#)
- global_init
 - global.cc, [281](#)
 - global.h, [283](#)
- GLOBAL_MEMORY_MINIMUM
 - global.h, [282](#)
- GLOBAL_MEMORY_SIZE
 - global.h, [283](#)
- global_shut
 - global.cc, [282](#)
 - global.h, [283](#)
- GrAggRecord, [50](#)
 - count, [51](#)
 - GrAggRecord, [50](#)
 - middle_record, [51](#)
 - sum, [51](#)
- group_by_hash_t
 - GroupbyAggre, [55](#)
- group_by_key_t
 - GroupbyAggre, [55](#)

- group_by_pos
 - GroupbyAggre, 75
- group_by_size
 - GroupbyAggre, 76
- group_by_type
 - GroupbyAggre, 76
- group_by_type_t
 - GroupbyAggre, 55
- groupby
 - SelectQuery, 201
- groupby_number
 - SelectQuery, 201
- groupby_rank
 - GroupbyAggre, 76
- GroupbyAggre, 53
 - ~GroupbyAggre, 56
 - aggr_method, 74
 - aggr_pos, 74
 - aggr_type, 74
 - avg_table, 74
 - avgFloat32, 56
 - avgFloat64, 56
 - avgInt16, 57
 - avgInt32, 57
 - avgInt64, 57
 - avgInt8, 58
 - buf_for_child, 74
 - child, 75
 - child_buf_size, 75
 - child_tuple_size, 75
 - close, 58
 - conditions, 75
 - count, 58
 - final_avg_table, 75
 - final_method, 75
 - final_sum_table, 75
 - finalCount, 59
 - finalFloat32Avg, 59
 - finalFloat32Sum, 59
 - finalFloat64Avg, 60
 - finalFloat64Sum, 60
 - finalInt16Sum, 60
 - finalInt32Sum, 61
 - finalInt64Sum, 61
 - finalInt8Sum, 61
 - finalIntAvg, 62
 - getNext, 62
 - group_by_hash_t, 55
 - group_by_key_t, 55
 - group_by_pos, 75
 - group_by_size, 76
 - group_by_type, 76
 - group_by_type_t, 55
 - groupby_rank, 76
 - GroupbyAggre, 55
 - hash_group, 76
 - in_cid, 76
 - init_max_table, 76
 - init_method, 76
 - init_min_table, 77
 - initAvg, 62
 - initCount, 62
 - initFloat32Max, 63
 - initFloat32Min, 63
 - initFloat64Max, 63
 - initFloat64Min, 64
 - initInt16Max, 64
 - initInt16Min, 64
 - initInt32Max, 65
 - initInt32Min, 65
 - initInt64Max, 65
 - initInt64Min, 66
 - initInt8Max, 66
 - initInt8Min, 66
 - initSum, 66
 - max_table, 77
 - maxFloat32, 67
 - maxFloat64, 67
 - maxInt16, 67
 - maxInt32, 68
 - maxInt64, 68
 - maxInt8, 69
 - middle_buf_array, 77
 - middle_buf_size, 77
 - middle_tuple_size, 77
 - min_table, 77
 - minFloat32, 69
 - minFloat64, 69
 - minInt16, 70
 - minInt32, 70
 - minInt64, 70
 - minInt8, 71
 - next_iter, 77
 - open, 71
 - out_cid, 77
 - set, 71
 - setChild, 72
 - sum_table, 78
 - sumFloat32, 72
 - sumFloat64, 72
 - sumInt16, 72
 - sumInt32, 73
 - sumInt64, 73
 - sumInt8, 74
- GroupbyAggre::group_by_hash, 51
 - operator(), 51
- GroupbyAggre::group_by_key, 52
 - operator==, 52
 - type_array, 52
 - value_array, 52
- GT
 - executor.h, 269
- hash
 - HashIndex, 84
 - HashInfo, 91
- hash_code

- Hashcode_Ptr, [80](#)
- hash_group
 - GroupbyAggre, [76](#)
- hash_index
 - HashJoin, [94](#)
- HashCell, [78](#)
 - capacity, [78](#)
 - ent, [78](#)
 - ents, [79](#)
 - hc_num, [79](#)
 - hc_union, [79](#)
 - num_2_or_more, [79](#)
- Hashcode_Ptr, [79](#)
 - hash_code, [80](#)
 - tuple, [80](#)
- HASHINDEX
 - schema.h, [317](#)
- HashIndex, [80](#)
 - addIndexDTpye, [82](#)
 - cmpEQ, [82](#)
 - del, [83](#)
 - finish, [84](#)
 - hash, [84](#)
 - HashIndex, [81](#)
 - ih_cell_capbits, [89](#)
 - ih_column_cap, [89](#)
 - ih_column_num, [90](#)
 - ih_datatype, [90](#)
 - ih_hash_bits, [90](#)
 - ih_hashtable, [90](#)
 - ih_table_offset, [90](#)
 - init, [84](#)
 - insert, [85](#)
 - lookup, [86](#)
 - print, [87](#)
 - set_Is, [87](#)
 - setCellCap, [88](#)
 - shut, [88](#)
 - tranToInt64, [88](#), [89](#)
- hashindex.h
 - HASHINFO_CAPICITY, [285](#)
- HashInfo, [90](#)
 - hash, [91](#)
 - last, [91](#)
 - ppos, [91](#)
 - result, [91](#)
 - rnum, [91](#)
- HASHINFO_CAPICITY
 - hashindex.h, [285](#)
- HashJoin, [92](#)
 - ~HashJoin, [93](#)
 - close, [93](#)
 - getNext, [94](#)
 - hash_index, [94](#)
 - HashJoin, [93](#)
 - last_iter, [95](#)
 - left_buf, [95](#)
 - left_key_off, [95](#)
 - left_key_type, [95](#)
 - left_tuple_size, [95](#)
 - middle_buf_array, [95](#)
 - middle_buf_size, [95](#)
 - open, [94](#)
 - right_buf, [95](#)
 - right_buf_size, [96](#)
 - right_has_next, [96](#)
 - right_key_pos, [96](#)
 - right_key_type, [96](#)
 - right_tuple_size, [96](#)
 - txt_buf, [96](#)
 - upper_iter, [96](#)
- HashTable, [97](#)
 - ~HashTable, [98](#)
 - add, [98](#)
 - allocate, [99](#)
 - avail, [101](#)
 - begin, [101](#)
 - del, [99](#)
 - end, [101](#)
 - estimated_duplicates_per_key, [101](#)
 - estimated_num_distinct_keys, [101](#)
 - free, [99](#)
 - free_header, [102](#)
 - HashTable, [98](#)
 - initial_array_size, [102](#)
 - more_allocated, [102](#)
 - pointer2size, [102](#)
 - probe, [99](#)
 - probe_contd, [100](#)
 - show, [100](#)
 - size_to_slot, [101](#)
 - table, [102](#)
 - table_size, [102](#)
 - utilization, [101](#)
- hashtable.cc
 - ESTIMATE_ERROR, [287](#)
- hashtable.h
 - hc_capacity, [287](#)
 - hc_ent, [287](#)
 - hc_ents, [288](#)
- having
 - SelectQuery, [201](#)
- hc_capacity
 - hashtable.h, [287](#)
- hc_ent
 - hashtable.h, [287](#)
- hc_ents
 - hashtable.h, [288](#)
- hc_num
 - HashCell, [79](#)
- hc_union
 - HashCell, [79](#)
- i_key
 - Index, [116](#)
- i_t_id
 - Index, [116](#)

- i_type
 - Index, 116
 - IndexScan, 123
- id2Name
 - ErrorLog, 33
- ih_cell_capbits
 - HashIndex, 89
- ih_column_cap
 - HashIndex, 89
- ih_column_num
 - HashIndex, 90
- ih_datatype
 - HashIndex, 90
- ih_hash_bits
 - HashIndex, 90
- ih_hashtable
 - HashIndex, 90
- ih_table_offset
 - HashIndex, 90
- in_buf_size
 - Project, 171
- in_cid
 - GroupbyAggre, 76
- in_tuple_size
 - Filter, 49
 - Project, 172
- INDEX
 - schema.h, 317
- Index, 103
 - ~Index, 104
 - del, 104–106
 - finish, 106
 - getIKey, 106
 - getIndexTid, 106
 - getIType, 107
 - i_key, 116
 - i_t_id, 116
 - i_type, 116
 - Index, 104
 - init, 107
 - insert, 107
 - lookup, 108, 109
 - print, 110
 - scan, 110
 - scan_1, 110, 112
 - scan_2, 112, 113
 - set_Is, 113, 114
 - setIndexTid, 114
 - shut, 114
 - tranToInt64, 114, 115
 - update, 115
- index
 - IndexScan, 123
- IndexJoin, 116
 - ~IndexJoin, 117
 - close, 118
 - current_key, 119
 - getNext, 118
 - IndexJoin, 117
 - left_buf, 119
 - left_buf_size, 119
 - left_tuple_size, 119
 - open, 118
 - right_buf, 119
 - right_buf_size, 119
 - right_has_next, 119
 - right_tuple_size, 119
- IndexScan, 120
 - ~IndexScan, 121
 - close, 121
 - current_key, 123
 - from, 123
 - getNext, 122
 - i_type, 123
 - index, 123
 - IndexScan, 121
 - info_ptr, 123
 - key_end, 123
 - open, 122
 - setTabIdx, 122
 - updateKey, 122
- IndexType
 - schema.h, 316
- info_ptr
 - IndexScan, 123
- init
 - Catalog, 16
 - Column, 22
 - Database, 27
 - ErrorLog, 33
 - HashIndex, 84
 - Index, 107
 - Memory, 134
 - MStorage, 138
 - Pbtree, 150, 154
 - PbtreeIndex, 158
 - ResultTable, 176
 - RowTable, 184
 - RPattern, 194
 - Table, 207
- init_max_table
 - GroupbyAggre, 76
- init_method
 - GroupbyAggre, 76
- init_min_table
 - GroupbyAggre, 77
- initAvg
 - GroupbyAggre, 62
- initCmpFunc
 - Filter, 46
- initColumn
 - Catalog, 16
- initCount
 - GroupbyAggre, 62
- initDatabase
 - Catalog, 17

- initFloat32Max
 - GroupbyAggre, [63](#)
- initFloat32Min
 - GroupbyAggre, [63](#)
- initFloat64Max
 - GroupbyAggre, [63](#)
- initFloat64Min
 - GroupbyAggre, [64](#)
- initial_array_size
 - HashTable, [102](#)
- initIndex
 - Catalog, [17](#)
- initInt16Max
 - GroupbyAggre, [64](#)
- initInt16Min
 - GroupbyAggre, [64](#)
- initInt32Max
 - GroupbyAggre, [65](#)
- initInt32Min
 - GroupbyAggre, [65](#)
- initInt64Max
 - GroupbyAggre, [65](#)
- initInt64Min
 - GroupbyAggre, [66](#)
- initInt8Max
 - GroupbyAggre, [66](#)
- initInt8Min
 - GroupbyAggre, [66](#)
- initSum
 - GroupbyAggre, [66](#)
- initTable
 - Catalog, [18](#)
- input_cid
 - Filter, [49](#)
 - Project, [172](#)
- input_off
 - Project, [172](#)
- input_pos
 - Project, [172](#)
- input_type
 - Project, [172](#)
- insert
 - Database, [27](#), [28](#)
 - HashIndex, [85](#)
 - Index, [107](#)
 - Pbtree, [151](#), [154](#)
 - PbtreeIndex, [158](#)
 - RowTable, [184](#)
 - Table, [208](#)
- INT16
 - schema.h, [316](#)
- INT16_TC
 - datatype.h, [251](#)
- INT32
 - schema.h, [316](#)
- INT32_TC
 - datatype.h, [251](#)
- INT64
 - schema.h, [316](#)
- INT64_TC
 - datatype.h, [251](#)
- INT8
 - schema.h, [316](#)
- INT8_TC
 - datatype.h, [251](#)
- invalid
 - RowTable, [185](#)
- INVID_C
 - schema.h, [316](#)
- INVID_I
 - schema.h, [316](#)
- INVID_O
 - schema.h, [317](#)
- INVID_T
 - schema.h, [317](#)
- INVID_TC
 - datatype.h, [251](#)
- isValid
 - RowTable, [185](#)
- ITEM_SIZE
 - nodepref.h, [293](#)
- Join, [124](#)
 - ~Join, [125](#)
 - close, [125](#)
 - getLeftCol, [125](#)
 - getLeftOp, [126](#)
 - getLeftRank, [126](#)
 - getNext, [126](#)
 - getRightCol, [127](#)
 - getRightOp, [127](#)
 - getRightRank, [127](#)
 - Join, [125](#)
 - left, [129](#)
 - left_cid, [129](#)
 - left_rank, [129](#)
 - open, [127](#)
 - right, [129](#)
 - right_cid, [129](#)
 - right_rank, [129](#)
 - setJoinCol, [128](#)
 - setLeftOp, [128](#)
 - setRightOp, [128](#)
- k
 - bnode, [12](#)
- Key, [130](#)
 - contain, [130](#)
 - getKey, [131](#)
 - Key, [130](#)
 - key, [131](#)
 - operator=, [131](#)
 - print, [131](#)
 - set, [131](#)
- key
 - bnode, [13](#)
 - Key, [131](#)

- key_end
 - IndexScan, 123
- KEY_SIZE
 - pbtrees.h, 300
- key_type
 - pbtrees.h, 301
- L3_CACHE_LINE
 - nodepref.h, 293
- l_ptr
 - PbtreesInfo, 163
- last
 - HashInfo, 91
- last_iter
 - HashJoin, 95
- LE
 - executor.h, 269
- le_resu
 - PbtreesInfo, 163
- LEAF_KEY_NUM
 - pbtrees.h, 300
- LEAF_PREF
 - nodepref.h, 293
- LEAF_PREF_ST
 - nodepref.h, 293
- left
 - Join, 129
 - PbtreesInfo, 163
- left_buf
 - HashJoin, 95
 - IndexJoin, 119
- left_buf_size
 - IndexJoin, 119
- left_cid
 - Join, 129
- LEFT_KEY_NUM
 - pbtrees.cc, 298
- left_key_off
 - HashJoin, 95
- left_key_type
 - HashJoin, 95
- left_rank
 - Join, 129
- left_tuple_size
 - HashJoin, 95
 - IndexJoin, 119
- leve2cap
 - Pbtrees, 151, 154
- leve2size
 - Pbtrees, 151, 154
- LINK
 - executor.h, 269
- load_data
 - runaimdb.cc, 312
- load_schema
 - runaimdb.cc, 313
- loadData
 - Database, 28
 - RowTable, 185
- Table, 208
- log
 - ErrorLog, 33
- lookup
 - HashIndex, 86
 - Index, 108, 109
 - Pbtrees, 151, 154
 - PbtreesIndex, 159
- lookup_s
 - Pbtrees, 151, 155
- LT
 - executor.h, 269
- m_array_list
 - Memory, 135
- m_curr
 - Memory, 135
- m_head
 - Memory, 135
- m_mins
 - Memory, 135
- m_table_addr
 - Memory, 135
- m_tail
 - Memory, 135
- m_total
 - Memory, 136
- main
 - runaimdb.cc, 313
- MAX
 - executor.h, 269
- MAX_AM
 - executor.h, 269
- MAX_CM
 - executor.h, 269
- MAX_KEY
 - pbtrees.h, 300
- max_table
 - GroupbyAggre, 77
- maxFloat32
 - GroupbyAggre, 67
- maxFloat64
 - GroupbyAggre, 67
- maxInt16
 - GroupbyAggre, 67
- maxInt32
 - GroupbyAggre, 68
- maxInt64
 - GroupbyAggre, 68
- maxInt8
 - GroupbyAggre, 69
- MAXTYPE_C
 - schema.h, 316
- MAXTYPE_I
 - schema.h, 317
- MAXTYPE_O
 - schema.h, 317
- MAXTYPE_T
 - schema.h, 317

- MAXTYPE_TC
 - datatype.h, 251
- Memory, 132
 - alloc, 132
 - alloc_default, 133
 - allocTableAddr, 133
 - free, 133
 - init, 134
 - m_array_list, 135
 - m_curr, 135
 - m_head, 135
 - m_mins, 135
 - m_table_addr, 135
 - m_tail, 135
 - m_total, 136
 - print, 134
 - shut, 134
 - slot, 135
- MEMORY_OK
 - mymemory.h, 290
- method
 - AggreCondition, 7
- middle_buf_array
 - GroupbyAggre, 77
 - HashJoin, 95
 - Orderby, 148
- middle_buf_size
 - GroupbyAggre, 77
 - HashJoin, 95
 - Orderby, 148
- middle_record
 - GrAggRecord, 51
- middle_tuple_size
 - GroupbyAggre, 77
- MIN
 - executor.h, 269
- MIN_KEY
 - pbtrees.h, 300
- min_table
 - GroupbyAggre, 77
- minFloat32
 - GroupbyAggre, 69
- minFloat64
 - GroupbyAggre, 69
- minInt16
 - GroupbyAggre, 70
- minInt32
 - GroupbyAggre, 70
- minInt64
 - GroupbyAggre, 70
- minInt8
 - GroupbyAggre, 71
- more_allocated
 - HashTable, 102
- ms_memory
 - MStorage, 138
- ms_memory_cur
 - MStorage, 138
- ms_memory_size
 - MStorage, 139
- ms_record_max
 - MStorage, 139
- ms_record_num
 - MStorage, 139
- ms_record_size
 - MStorage, 139
- MStorage, 136
 - allocRow, 137
 - expand, 137
 - getRecordNum, 137
 - getRow, 137
 - init, 138
 - ms_memory, 138
 - ms_memory_cur, 138
 - ms_memory_size, 139
 - ms_record_max, 139
 - ms_record_num, 139
 - ms_record_size, 139
 - pad, 139
 - shut, 138
- mymemory.cc
 - g_memory, 289
- mymemory.h
 - g_memory, 291
 - MEMORY_OK, 290
 - NON_TABLE_MEMORY_ADDR, 291
 - TABLE_MEMORY_ALLOC_INC, 291
 - TABLE_MEMORY_ALLOC_MAX, 291
 - TABLE_MEMORY_INIT_ADDR, 291
 - TABLE_MEMORY_MAX_ADDR, 291
- name
 - RequestColumn, 174
 - RequestTable, 174
- name2Id
 - ErrorLog, 34
- NE
 - executor.h, 269
- next_iter
 - GroupbyAggre, 77
- next_record
 - Scan, 199
- NODE_LINE_NUM
 - nodepref.h, 293
- nodepref.h
 - AREA_LINE_NUM, 292
 - BNODE_SIZE, 293
 - CACHE_LINE_SIZE, 293
 - ITEM_SIZE, 293
 - L3_CACHE_LINE, 293
 - LEAF_PREF, 293
 - LEAF_PREF_ST, 293
 - NODE_LINE_NUM, 293
- NON_LEAF_KEY_NUM
 - pbtrees.h, 300
- NON_TABLE_MEMORY_ADDR
 - mymemory.h, 291

NONE_AM
 executor.h, 269
 NONE_CM
 executor.h, 269
 num_2_or_more
 HashCell, 79

 o_id
 Object, 141
 o_name
 Object, 142
 o_type
 Object, 142
 OBJ_NAME_MAX
 schema.h, 316
 Object, 139
 changeName, 140
 getOid, 141
 getOname, 141
 getOtype, 141
 o_id, 141
 o_name, 142
 o_type, 142
 Object, 140
 print, 141
 shut, 141
 ObjectType
 schema.h, 317
 obtainId
 Catalog, 18
 offset
 ResultTable, 178
 offset_size
 ResultTable, 178
 open
 Filter, 46
 GroupbyAggre, 71
 HashJoin, 94
 IndexJoin, 118
 IndexScan, 122
 Join, 127
 Operator, 144
 Orderby, 146
 Project, 169
 Scan, 198
 Operator, 142
 ~Operator, 143
 buffer_from_father, 144
 close, 143
 getBuffer, 143
 getNext, 143
 open, 144
 Operator, 143
 setBuffer, 144
 operator char *
 Pointer8B, 165
 operator struct bleaf *
 Pointer8B, 165
 operator struct bnode *
 Pointer8B, 165
 operator unsigned long long
 Pointer8B, 165
 operator void *
 Pointer8B, 165
 operator()
 GroupbyAggre::group_by_hash, 51
 operator=
 Key, 131
 Pointer8B, 166
 operator==
 GroupbyAggre::group_by_key, 52
 Orderby, 145
 ~Orderby, 146
 arrayid, 147
 child, 148
 child_buffer, 148
 close, 146
 colid, 148
 coloff, 148
 colrank, 148
 coltype, 148
 getNext, 146
 middle_buf_array, 148
 middle_buf_size, 148
 open, 146
 Orderby, 145
 orderby_num, 149
 self_buf_size, 149
 set, 147
 setChild, 147
 tuple_size, 149
 orderby
 SelectQuery, 201
 orderby_num
 Orderby, 149
 orderby_number
 SelectQuery, 201
 out_cid
 GroupbyAggre, 77
 out_to_in
 Project, 172
 output_cid
 Project, 172
 output_type
 Project, 172
 output_type_buf_size
 Project, 173
 output_type_size
 Project, 173

 p_free_header
 Pbtree, 152, 156
 p_pbtrees
 Pbtree, 152, 156
 pad
 MStorage, 139
 par
 RPattern, 196

- Pbtree, 149, 152
 - allocate, 150, 153
 - cap2leve, 150, 153
 - del, 150, 153
 - free, 150, 153
 - get_recptr, 150, 154
 - init, 150, 154
 - insert, 151, 154
 - leve2cap, 151, 154
 - leve2size, 151, 154
 - lookup, 151, 154
 - lookup_s, 151, 155
 - p_free_header, 152, 156
 - p_pbtree, 152, 156
 - print, 151, 155
 - scan, 151, 155
 - shut, 152, 155
 - size2leve, 152, 155
- pbtree.cc
 - LEFT_KEY_NUM, 298
 - RIGHT_KEY_NUM, 298
- pbtree.h
 - BKEY_NUM, 299
 - bleaf, 299
 - bnext, 300
 - bnum, 300
 - KEY_SIZE, 300
 - key_type, 301
 - LEAF_KEY_NUM, 300
 - MAX_KEY, 300
 - MIN_KEY, 300
 - NON_LEAF_KEY_NUM, 300
 - POINTER8B_SIZE, 301
 - POINTER_SIZE, 301
- PbtreeIndex, 156
 - del, 157
 - init, 158
 - insert, 158
 - lookup, 159
 - PbtreeIndex, 157
 - pi_datatype, 162
 - pi_pbtree, 162
 - print, 159
 - scan, 159
 - set_ls, 160
 - setIndexDTpye, 161
 - shut, 161
- pbtreeindex.h
 - PBTREEINFO_CAPICITY, 306
- PbtreeInfo, 162
 - area, 163
 - cr_area, 163
 - cr_resu, 163
 - l_ptr, 163
 - le_resu, 163
 - left, 163
 - pos_resu, 163
 - result, 163
 - right, 164
 - s_end, 164
 - s_num, 164
 - s_pos, 164
 - s_ptr, 164
- PBTREEINFO_CAPICITY
 - pbtreeindex.h, 306
- pfld
 - gcc_pf_p3.h, 279
- pfldnta
 - gcc_pf_p3.h, 279
- pfst
 - gcc_pf_p3.h, 279
- pfstnta
 - gcc_pf_p3.h, 280
- pi_datatype
 - PbtreeIndex, 162
- pi_pbtree
 - PbtreeIndex, 162
- planner
 - Executor, 40
- pointer2size
 - HashTable, 102
- Pointer8B, 164
 - operator char *, 165
 - operator struct bleaf *, 165
 - operator struct bnode *, 165
 - operator unsigned long long, 165
 - operator void *, 165
 - operator=, 166
 - print, 166
 - value, 166
- POINTER8B_SIZE
 - pbtree.h, 301
- POINTER_SIZE
 - pbtree.h, 301
- pos_resu
 - PbtreeInfo, 163
- ppos
 - HashInfo, 91
- prefetchnta
 - gcc_pf_p3.h, 280
- prefetcht0
 - gcc_pf_p3.h, 280
- prefetcht1
 - gcc_pf_p3.h, 280
- print
 - Catalog, 18
 - Column, 22
 - Database, 29
 - HashIndex, 87
 - Index, 110
 - Key, 131
 - Memory, 134
 - Object, 141
 - Pbtree, 151, 155
 - PbtreeIndex, 159
 - Pointer8B, 166

- ResultTable, 177
- RPattern, 195
- Table, 209
- print_flag
 - runaimdb.cc, 314
- printData
 - RowTable, 186
 - Table, 209
- probe
 - HashTable, 99
- probe_contd
 - HashTable, 100
- Project, 166
 - ~Project, 167
 - buf_for_child, 171
 - child, 171
 - close, 168
 - getColnum, 168
 - getNext, 169
 - getSchema, 169
 - in_buf_size, 171
 - in_tuple_size, 172
 - input_cid, 172
 - input_off, 172
 - input_pos, 172
 - input_type, 172
 - open, 169
 - out_to_in, 172
 - output_cid, 172
 - output_type, 172
 - output_type_buf_size, 173
 - output_type_size, 173
 - Project, 167, 168
 - self_buf_size, 173
 - setChild, 169
 - setProjCol, 171
 - top, 171
 - topid, 173
- ptouch
 - gcc_pf_p3.h, 280
- querys
 - runaimdb.cc, 314
- r_pattern
 - RowTable, 192
- r_storage
 - RowTable, 192
- registerObj
 - Catalog, 19
- RequestColumn, 173
 - aggregate_method, 174
 - name, 174
- RequestTable, 174
 - name, 174
- reset
 - ErrorLog, 34
 - RPattern, 195
- result
 - HashInfo, 91
 - PbtreeInfo, 163
- ResultTable, 175
 - append, 175
 - buffer, 178
 - buffer_size, 178
 - column_number, 178
 - column_type, 178
 - dump, 176
 - getRC, 176
 - init, 176
 - offset, 178
 - offset_size, 178
 - print, 177
 - row_capacity, 178
 - row_length, 179
 - row_number, 179
 - shut, 177
 - writeRC, 177
- right
 - Join, 129
 - PbtreeInfo, 164
- right_buf
 - HashJoin, 95
 - IndexJoin, 119
- right_buf_size
 - HashJoin, 96
 - IndexJoin, 119
- right_cid
 - Join, 129
- right_has_next
 - HashJoin, 96
 - IndexJoin, 119
- RIGHT_KEY_NUM
 - pbtrees.cc, 298
- right_key_pos
 - HashJoin, 96
- right_key_type
 - HashJoin, 96
- right_rank
 - Join, 129
- right_tuple_size
 - HashJoin, 96
 - IndexJoin, 119
- rnum
 - HashInfo, 91
- root
 - Executor, 40
- row_capacity
 - ResultTable, 178
- row_length
 - ResultTable, 179
- row_number
 - ResultTable, 179
- ROWTABLE
 - schema.h, 317
- RowTable, 179
 - access, 181

- accessCol, 181
- del, 182
- finish, 182
- getMStorage, 183
- getRecordNum, 183
- getRecordPtr, 183
- getRPattern, 183
- init, 184
- insert, 184
- invalid, 185
- isValid, 185
- loadData, 185
- printData, 186
- r_pattern, 192
- r_storage, 192
- RowTable, 180
- select, 186
- selectCol, 187
- selectCols, 188
- shut, 189
- updateCol, 189, 190
- updateCols, 190–192
- rowtable.h
 - g_memory, 308
- rp_colnum
 - RPattern, 196
- rp_current
 - RPattern, 196
- rp_dtype
 - RPattern, 196
- rp_mem_sz
 - RPattern, 196
- rp_memory
 - RPattern, 196
- rp_offset
 - RPattern, 196
- rp_row_sz
 - RPattern, 196
- RPattern, 193
 - addColumn, 193
 - getColumnOffset, 193
 - getColumnType, 194
 - getRowSize, 194
 - init, 194
 - par, 196
 - print, 195
 - reset, 195
 - rp_colnum, 196
 - rp_current, 196
 - rp_dtype, 196
 - rp_mem_sz, 196
 - rp_memory, 196
 - rp_offset, 196
 - rp_row_sz, 196
 - shut, 195
- runaimdb.cc
 - load_data, 312
 - load_schema, 313
 - main, 313
 - print_flag, 314
 - querys, 314
 - table_name, 314
 - test, 313
 - testOne, 313
- s_end
 - PbtreeInfo, 164
- s_num
 - PbtreeInfo, 164
- s_pos
 - PbtreeInfo, 164
- s_ptr
 - PbtreeInfo, 164
- Scan, 197
 - ~Scan, 198
 - close, 198
 - getNext, 198
 - next_record, 199
 - open, 198
 - Scan, 197
 - scan_table, 199
 - setTable, 199
 - total_record, 199
- scan
 - Index, 110
 - Pbtree, 151, 155
 - PbtreeIndex, 159
- scan_1
 - Index, 110, 112
- scan_2
 - Index, 112, 113
- scan_table
 - Scan, 199
- schema.h
 - ARTTREEINDEX, 317
 - BPTREEINDEX, 317
 - CHARN, 316
 - COLTABLE, 317
 - COLUMN, 317
 - ColumnType, 316
 - DATABASE, 317
 - DATE, 316
 - DATETIME, 316
 - FLOAT32, 316
 - FLOAT64, 316
 - HASHINDEX, 317
 - INDEX, 317
 - IndexType, 316
 - INT16, 316
 - INT32, 316
 - INT64, 316
 - INT8, 316
 - INVID_C, 316
 - INVID_I, 316
 - INVID_O, 317
 - INVID_T, 317
 - MAXTYPE_C, 316

- MAXTYPE_I, [317](#)
- MAXTYPE_O, [317](#)
- MAXTYPE_T, [317](#)
- OBJ_NAME_MAX, [316](#)
- ObjectType, [317](#)
- ROWTABLE, [317](#)
- TABLE, [317](#)
- TableType, [317](#)
- TIME, [316](#)
- select
 - RowTable, [186](#)
 - Table, [209](#), [210](#)
- select_column
 - SelectQuery, [201](#)
- select_number
 - SelectQuery, [201](#)
- selectCol
 - RowTable, [187](#)
 - Table, [210](#)
- selectCols
 - RowTable, [188](#)
 - Table, [211](#)
- SelectQuery, [200](#)
 - database_id, [200](#)
 - from_number, [200](#)
 - from_table, [201](#)
 - groupby, [201](#)
 - groupby_number, [201](#)
 - having, [201](#)
 - orderby, [201](#)
 - orderby_number, [201](#)
 - select_column, [201](#)
 - select_number, [201](#)
 - where, [202](#)
- self_buf_size
 - Orderby, [149](#)
 - Project, [173](#)
- set
 - GroupbyAggre, [71](#)
 - Key, [131](#)
 - Orderby, [147](#)
- set_Is
 - HashIndex, [87](#)
 - Index, [113](#), [114](#)
 - PbtreeIndex, [160](#)
- setBuffer
 - Operator, [144](#)
- setCellCap
 - HashIndex, [88](#)
- setChild
 - Filter, [46](#)
 - GroupbyAggre, [72](#)
 - Orderby, [147](#)
 - Project, [169](#)
- setCoffset
 - Column, [22](#)
- setColumn
 - Filter, [47](#)
- setFiltCond
 - Filter, [48](#)
- setIndexDTpye
 - PbtreeIndex, [161](#)
- setIndexTid
 - Index, [114](#)
- setJoinCol
 - Join, [128](#)
- setLeftOp
 - Join, [128](#)
- setLevel
 - ErrorLog, [34](#)
- setProjCol
 - Project, [171](#)
- setRightOp
 - Join, [128](#)
- setTabIdx
 - IndexScan, [122](#)
- setTable
 - Scan, [199](#)
- show
 - HashTable, [100](#)
- shut
 - Catalog, [19](#)
 - Column, [23](#)
 - Database, [29](#)
 - HashIndex, [88](#)
 - Index, [114](#)
 - Memory, [134](#)
 - MStorage, [138](#)
 - Object, [141](#)
 - Pbtree, [152](#), [155](#)
 - PbtreeIndex, [161](#)
 - ResultTable, [177](#)
 - RowTable, [189](#)
 - RPattern, [195](#)
 - Table, [212](#)
- shutDatabase
 - Catalog, [19](#)
- size2leve
 - Pbtree, [152](#), [155](#)
- size_to_slot
 - HashTable, [101](#)
- slot
 - Memory, [135](#)
- SUM
 - executor.h, [269](#)
- sum
 - GrAggRecord, [51](#)
- sum_table
 - GroupbyAggre, [78](#)
- sumFloat32
 - GroupbyAggre, [72](#)
- sumFloat64
 - GroupbyAggre, [72](#)
- sumInt16
 - GroupbyAggre, [72](#)
- sumInt32

- GroupbyAggre, 73
- sumInt64
 - GroupbyAggre, 73
- sumInt8
 - GroupbyAggre, 74
- system/catalog.cc, 247
- system/catalog.d, 248
- system/catalog.h, 248, 249
- system/datatype.h, 250, 251
- system/errorlog.cc, 256
- system/errorlog.d, 258
- system/errorlog.h, 258, 263
- system/executor.cc, 265
- system/executor.d, 267
- system/executor.h, 267, 270
- system/gcc_pf_p3.h, 279, 281
- system/global.cc, 281
- system/global.d, 282
- system/global.h, 282, 284
- system/hashindex.cc, 284
- system/hashindex.d, 284
- system/hashindex.h, 284, 286
- system/hashtable.cc, 286
- system/hashtable.d, 287
- system/hashtable.h, 287, 288
- system/mymemory.cc, 289
- system/mymemory.d, 290
- system/mymemory.h, 290, 292
- system/nodepref.h, 292, 294
- system/pbtree.cc, 297
- system/pbtree.d, 298
- system/pbtree.h, 298, 301
- system/pbtreeindex.cc, 305
- system/pbtreeindex.d, 305
- system/pbtreeindex.h, 305, 306
- system/rowtable.cc, 307
- system/rowtable.d, 307
- system/rowtable.h, 307, 309
- system/runaimdb.cc, 311
- system/runaimdb.d, 314
- system/schema.h, 314, 318
- t_columns
 - Table, 215
- t_index
 - Table, 215
- t_type
 - Table, 215
- TABLE
 - schema.h, 317
- Table, 202
 - ~Table, 203
 - addColumn, 204
 - addIndex, 204
 - del, 204, 205
 - finish, 205
 - getColumnRank, 205
 - getColumns, 206
 - getIndexRank, 206
 - getIndexes, 206
 - getRank, 207
 - getRecordNum, 207
 - getRecordPtr, 207
 - getTtype, 207
 - init, 207
 - insert, 208
 - loadData, 208
 - print, 209
 - printData, 209
 - select, 209, 210
 - selectCol, 210
 - selectCols, 211
 - shut, 212
 - t_columns, 215
 - t_index, 215
 - t_type, 215
 - Table, 203
 - updateCol, 212, 213
 - updateCols, 213, 214
- table
 - HashTable, 102
- TABLE_MEMORY_ALLOC_INC
 - mymemory.h, 291
- TABLE_MEMORY_ALLOC_MAX
 - mymemory.h, 291
- TABLE_MEMORY_INIT_ADDR
 - mymemory.h, 291
- TABLE_MEMORY_MAX_ADDR
 - mymemory.h, 291
- table_name
 - runaimdb.cc, 314
- table_size
 - HashTable, 102
- TableType
 - schema.h, 317
- test
 - runaimdb.cc, 313
- testOne
 - runaimdb.cc, 313
- thread_el
 - errorlog.cc, 257
 - errorlog.h, 263
- TIME
 - schema.h, 316
- TIME_TC
 - datatype.h, 251
- top
 - Project, 171
- topid
 - Project, 173
- total_record
 - Scan, 199
- tranToInt64
 - HashIndex, 88, 89
 - Index, 114, 115
- tuple
 - Hashcode_Ptr, 80

- tuple_size
 - Orderby, [149](#)
- txt_buf
 - HashJoin, [96](#)
- type_array
 - GroupbyAggre::group_by_key, [52](#)
- TypeCharN, [216](#)
 - cmpEQ, [217](#)
 - cmpGE, [217](#)
 - cmpGT, [217](#)
 - cmpLE, [217](#)
 - cmpLT, [217](#)
 - copy, [218](#)
 - formatBin, [218](#)
 - formatTxt, [218](#)
 - TypeCharN, [216](#)
- TypeCode
 - datatype.h, [250](#)
- TypeDate, [219](#)
 - cmpEQ, [219](#)
 - cmpGE, [220](#)
 - cmpGT, [220](#)
 - cmpLE, [220](#)
 - cmpLT, [220](#)
 - copy, [221](#)
 - formatBin, [221](#)
 - formatTxt, [221](#)
 - TypeDate, [219](#)
- TypeDateTime, [222](#)
 - cmpEQ, [222](#)
 - cmpGE, [223](#)
 - cmpGT, [223](#)
 - cmpLE, [223](#)
 - cmpLT, [223](#)
 - copy, [224](#)
 - formatBin, [224](#)
 - formatTxt, [224](#)
 - TypeDateTime, [222](#)
- TypeFloat32, [225](#)
 - cmpEQ, [225](#)
 - cmpGE, [226](#)
 - cmpGT, [226](#)
 - cmpLE, [226](#)
 - cmpLT, [226](#)
 - copy, [227](#)
 - formatBin, [227](#)
 - formatTxt, [227](#)
 - TypeFloat32, [225](#)
- TypeFloat64, [228](#)
 - cmpEQ, [228](#)
 - cmpGE, [229](#)
 - cmpGT, [229](#)
 - cmpLE, [229](#)
 - cmpLT, [229](#)
 - copy, [230](#)
 - formatBin, [230](#)
 - formatTxt, [230](#)
 - TypeFloat64, [228](#)
- TypeInt16, [231](#)
 - cmpEQ, [231](#)
 - cmpGE, [232](#)
 - cmpGT, [232](#)
 - cmpLE, [232](#)
 - cmpLT, [232](#)
 - copy, [233](#)
 - formatBin, [233](#)
 - formatTxt, [233](#)
 - TypeInt16, [231](#)
- TypeInt32, [234](#)
 - cmpEQ, [234](#)
 - cmpGE, [235](#)
 - cmpGT, [235](#)
 - cmpLE, [235](#)
 - cmpLT, [235](#)
 - copy, [236](#)
 - formatBin, [236](#)
 - formatTxt, [236](#)
 - TypeInt32, [234](#)
- TypeInt64, [237](#)
 - cmpEQ, [237](#)
 - cmpGE, [238](#)
 - cmpGT, [238](#)
 - cmpLE, [238](#)
 - cmpLT, [238](#)
 - copy, [239](#)
 - formatBin, [239](#)
 - formatTxt, [239](#)
 - TypeInt64, [237](#)
- TypeInt8, [240](#)
 - cmpEQ, [240](#)
 - cmpGE, [241](#)
 - cmpGT, [241](#)
 - cmpLE, [241](#)
 - cmpLT, [241](#)
 - copy, [242](#)
 - formatBin, [242](#)
 - formatTxt, [242](#)
 - TypeInt8, [240](#)
- Time, [243](#)
 - cmpEQ, [243](#)
 - cmpGE, [244](#)
 - cmpGT, [244](#)
 - cmpLE, [244](#)
 - cmpLT, [244](#)
 - copy, [245](#)
 - formatBin, [245](#)
 - formatTxt, [245](#)
 - Time, [243](#)
- update
 - Index, [115](#)
- updateCol
 - RowTable, [189](#), [190](#)
 - Table, [212](#), [213](#)
- updateCols
 - RowTable, [190–192](#)
 - Table, [213](#), [214](#)

- updateKey
 - IndexScan, [122](#)
- upper_iter
 - HashJoin, [96](#)
- utilization
 - HashTable, [101](#)
- value
 - Condition, [24](#)
 - Filter, [49](#)
 - Pointer8B, [166](#)
- value_array
 - GroupbyAggre::group_by_key, [52](#)
- where
 - SelectQuery, [202](#)
- writeRC
 - ResultTable, [177](#)