

My Project

Generated by Doxygen 1.8.17

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 BasicType Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Constructor & Destructor Documentation	7
4.1.2.1 BasicType()	8
4.1.2.2 ~BasicType()	8
4.1.3 Member Function Documentation	8
4.1.3.1 cmpEQ()	8
4.1.3.2 cmpGE()	8
4.1.3.3 cmpGT()	9
4.1.3.4 cmpLE()	9
4.1.3.5 cmpLT()	9
4.1.3.6 copy()	9
4.1.3.7 formatBin()	10
4.1.3.8 formatTxt()	10
4.1.3.9 getTypeCode()	10
4.1.3.10 getTypeSize()	10
4.1.4 Member Data Documentation	10
4.1.4.1 b_type_code	10
4.1.4.2 b_type_size	11
4.2 bnode Class Reference	11
4.3 Catalog Class Reference	11
4.3.1 Detailed Description	12
4.3.2 Member Function Documentation	12
4.3.2.1 createColumn()	12
4.3.2.2 createDatabase()	12
4.3.2.3 createIndex()	13
4.3.2.4 createTable()	13
4.3.2.5 getObjById()	14
4.3.2.6 getObjByName()	14
4.3.2.7 init()	14
4.3.2.8 initDatabase()	15
4.3.2.9 print()	15
4.3.2.10 shut()	15

4.3.2.11 shutDatabase()	15
4.4 Column Class Reference	16
4.4.1 Detailed Description	16
4.4.2 Constructor & Destructor Documentation	16
4.4.2.1 Column()	16
4.4.2.2 ~Column()	17
4.4.3 Member Function Documentation	17
4.4.3.1 finish()	17
4.4.3.2 getCOffset()	17
4.4.3.3 getCSize()	17
4.4.3.4 getCType()	17
4.4.3.5 getDataType()	18
4.4.3.6 init()	18
4.4.3.7 print()	18
4.4.3.8 setCOffset()	18
4.4.3.9 shut()	18
4.5 Condition Struct Reference	18
4.5.1 Detailed Description	19
4.5.2 Member Data Documentation	19
4.5.2.1 column	19
4.5.2.2 compare	19
4.5.2.3 value	19
4.6 Conditions Struct Reference	19
4.6.1 Detailed Description	20
4.6.2 Member Data Documentation	20
4.6.2.1 condition	20
4.6.2.2 condition_num	20
4.7 Database Class Reference	20
4.7.1 Detailed Description	21
4.7.2 Constructor & Destructor Documentation	21
4.7.2.1 Database()	21
4.7.2.2 ~Database()	21
4.7.3 Member Function Documentation	21
4.7.3.1 addTable()	21
4.7.3.2 finish()	22
4.7.3.3 getTables()	22
4.7.3.4 init()	22
4.7.3.5 insert() [1/2]	22
4.7.3.6 insert() [2/2]	23
4.7.3.7 loadData()	23
4.7.3.8 print()	23
4.7.3.9 shut()	24

4.8 ErrorLog Class Reference	24
4.8.1 Detailed Description	24
4.8.2 Constructor & Destructor Documentation	25
4.8.2.1 ErrorLog()	25
4.8.2.2 ~ErrorLog()	25
4.8.3 Member Function Documentation	25
4.8.3.1 closeLog()	25
4.8.3.2 flushLog()	25
4.8.3.3 getErrorCode()	26
4.8.3.4 getErrorMsg()	26
4.8.3.5 id2Name()	26
4.8.3.6 init()	26
4.8.3.7 log()	27
4.8.3.8 name2Id()	27
4.8.3.9 reset()	27
4.8.3.10 setLevel()	27
4.9 Executor Class Reference	28
4.9.1 Detailed Description	28
4.9.2 Member Function Documentation	28
4.9.2.1 close()	28
4.9.2.2 exec()	29
4.10 Filter Class Reference	29
4.10.1 Detailed Description	29
4.10.2 Member Function Documentation	30
4.10.2.1 getNext()	30
4.10.2.2 init()	30
4.10.2.3 isEnd()	30
4.10.2.4 print()	31
4.10.3 Member Data Documentation	31
4.10.3.1 filter_judge_condition	31
4.10.3.2 filter_judge_num	31
4.11 Groupby Class Reference	31
4.11.1 Detailed Description	32
4.11.2 Member Function Documentation	32
4.11.2.1 getNext()	32
4.11.2.2 init()	32
4.11.2.3 isEnd()	33
4.11.2.4 print()	33
4.12 Groupby_struct Struct Reference	33
4.12.1 Detailed Description	34
4.12.2 Member Data Documentation	34
4.12.2.1 given_condition	34

4.12.2.2 value	34
4.13 HashCell Class Reference	34
4.13.1 Detailed Description	35
4.13.2 Member Data Documentation	35
4.13.2.1 capacity	35
4.13.2.2 ent	35
4.13.2.3 ents	35
4.13.2.4 hc_num	35
4.13.2.5 hc_union	35
4.13.2.6 num_2_or_more	35
4.14 Hashcode_Ptr Class Reference	36
4.14.1 Detailed Description	36
4.14.2 Member Data Documentation	36
4.14.2.1 hash_code	36
4.14.2.2 tuple	36
4.15 HashIndex Class Reference	36
4.15.1 Detailed Description	37
4.15.2 Constructor & Destructor Documentation	37
4.15.2.1 HashIndex()	37
4.15.3 Member Function Documentation	37
4.15.3.1 addIndexDTpye()	38
4.15.3.2 del() [1/2]	38
4.15.3.3 del() [2/2]	38
4.15.3.4 finish()	39
4.15.3.5 init()	39
4.15.3.6 insert() [1/2]	39
4.15.3.7 insert() [2/2]	40
4.15.3.8 lookup() [1/2]	40
4.15.3.9 lookup() [2/2]	41
4.15.3.10 print()	41
4.15.3.11 set_ls() [1/2]	41
4.15.3.12 set_ls() [2/2]	42
4.15.3.13 setCellCap()	42
4.15.3.14 shut()	43
4.16 HashInfo Struct Reference	43
4.16.1 Detailed Description	43
4.16.2 Member Data Documentation	43
4.16.2.1 hash	43
4.16.2.2 last	44
4.16.2.3 ppos	44
4.16.2.4 result	44
4.16.2.5 rnum	44

4.17 HashTable Class Reference	44
4.17.1 Detailed Description	45
4.17.2 Constructor & Destructor Documentation	45
4.17.2.1 HashTable()	45
4.17.2.2 ~HashTable()	45
4.17.3 Member Function Documentation	45
4.17.3.1 add()	46
4.17.3.2 del()	46
4.17.3.3 probe()	46
4.17.3.4 probe_contd()	47
4.17.3.5 show()	47
4.17.3.6 utilization()	47
4.17.4 Member Data Documentation	48
4.17.4.1 avail	48
4.17.4.2 begin	48
4.17.4.3 end	48
4.17.4.4 estimated_duplicates_per_key	48
4.17.4.5 estimated_num_distinct_keys	48
4.17.4.6 free_header	48
4.17.4.7 initial_array_size	48
4.17.4.8 more_allocated	49
4.17.4.9 table	49
4.17.4.10 table_size	49
4.18 Index Class Reference	49
4.18.1 Detailed Description	50
4.18.2 Constructor & Destructor Documentation	50
4.18.2.1 Index()	50
4.18.2.2 ~Index()	51
4.18.3 Member Function Documentation	51
4.18.3.1 del() [1/4]	51
4.18.3.2 del() [2/4]	51
4.18.3.3 del() [3/4]	52
4.18.3.4 del() [4/4]	52
4.18.3.5 finish()	53
4.18.3.6 getlKey()	53
4.18.3.7 getIndexTid()	53
4.18.3.8 getlType()	53
4.18.3.9 init()	53
4.18.3.10 insert() [1/2]	53
4.18.3.11 insert() [2/2]	54
4.18.3.12 lookup() [1/4]	54
4.18.3.13 lookup() [2/4]	55

4.18.3.14 lookup() [3/4]	55
4.18.3.15 lookup() [4/4]	56
4.18.3.16 print()	56
4.18.3.17 scan()	56
4.18.3.18 scan_1() [1/2]	57
4.18.3.19 scan_1() [2/2]	57
4.18.3.20 scan_2() [1/2]	58
4.18.3.21 scan_2() [2/2]	58
4.18.3.22 set_ls() [1/2]	58
4.18.3.23 set_ls() [2/2]	59
4.18.3.24 setIndexTid()	59
4.18.3.25 shut()	60
4.18.3.26 tranToInt64() [1/2]	60
4.18.3.27 tranToInt64() [2/2]	60
4.18.4 Member Data Documentation	61
4.18.4.1 i_key	61
4.18.4.2 i_t_id	61
4.18.4.3 i_type	61
4.19 Join Class Reference	61
4.19.1 Detailed Description	62
4.19.2 Member Function Documentation	62
4.19.2.1 getNext()	62
4.19.2.2 init()	62
4.19.2.3 isEnd()	63
4.19.2.4 print()	63
4.19.3 Member Data Documentation	63
4.19.3.1 Column_id_array_join	63
4.19.3.2 Column_id_array_prepare	64
4.19.3.3 hx	64
4.19.3.4 insert_hash_data	64
4.19.3.5 join_given_condition_num	64
4.19.3.6 join_lchild_rank	64
4.19.3.7 join_rchild_rank	64
4.19.3.8 lookup_hash_data	64
4.20 Key Class Reference	65
4.20.1 Detailed Description	65
4.20.2 Constructor & Destructor Documentation	65
4.20.2.1 Key()	65
4.20.3 Member Function Documentation	65
4.20.3.1 contain()	65
4.20.3.2 getKey()	66
4.20.3.3 operator=()	66

4.20.3.4 print()	66
4.20.3.5 set()	66
4.21 Memory Class Reference	66
4.21.1 Member Function Documentation	67
4.21.1.1 alloc()	67
4.21.1.2 allocTableAddr()	67
4.21.1.3 free()	67
4.21.1.4 init()	68
4.21.1.5 print()	68
4.21.1.6 shut()	68
4.22 MStorage Class Reference	69
4.22.1 Detailed Description	69
4.22.2 Member Function Documentation	69
4.22.2.1 allocRow()	69
4.22.2.2 getRecordNum()	69
4.22.2.3 getRow()	70
4.22.2.4 init()	70
4.22.2.5 shut()	70
4.23 Object Class Reference	71
4.23.1 Detailed Description	71
4.23.2 Constructor & Destructor Documentation	71
4.23.2.1 Object()	71
4.23.3 Member Function Documentation	71
4.23.3.1 changeName()	72
4.23.3.2 getOid()	72
4.23.3.3 getOname()	72
4.23.3.4 getOtype()	72
4.23.3.5 print()	72
4.23.3.6 shut()	72
4.24 Operator Class Reference	73
4.24.1 Detailed Description	73
4.24.2 Member Function Documentation	73
4.24.2.1 getNext()	73
4.24.2.2 init()	74
4.24.2.3 isEnd()	74
4.24.2.4 print()	74
4.24.3 Member Data Documentation	74
4.24.3.1 Column_id_array	74
4.24.3.2 current_buffer	74
4.24.3.3 lchild	75
4.24.3.4 parent	75
4.24.3.5 prev_buffer	75

4.24.3.6 rchild	75
4.24.3.7 row_column_RPattern	75
4.25 Orderby Class Reference	75
4.25.1 Member Function Documentation	76
4.25.1.1 getNext()	76
4.25.1.2 init()	76
4.25.1.3 isEnd()	77
4.25.1.4 print()	77
4.25.2 Member Data Documentation	77
4.25.2.1 count	77
4.25.2.2 orderby	77
4.25.2.3 orderby_data_type	78
4.25.2.4 orderby_number	78
4.25.2.5 orderby_offset	78
4.25.2.6 orderby_vector	78
4.26 Pbtrees Class Reference	78
4.27 pbtrees Class Reference	79
4.28 PbtreesIndex Class Reference	79
4.28.1 Constructor & Destructor Documentation	80
4.28.1.1 PbtreesIndex()	80
4.28.2 Member Function Documentation	80
4.28.2.1 del()	80
4.28.2.2 init()	81
4.28.2.3 insert()	81
4.28.2.4 lookup()	82
4.28.2.5 print()	82
4.28.2.6 scan()	83
4.28.2.7 set_ls()	83
4.28.2.8 setIndexDTpye()	84
4.28.2.9 shut()	84
4.29 PbtreesInfo Struct Reference	85
4.29.1 Detailed Description	85
4.29.2 Member Data Documentation	85
4.29.2.1 area	85
4.29.2.2 cr_area	85
4.29.2.3 cr_resu	86
4.29.2.4 l_ptr	86
4.29.2.5 le_resu	86
4.29.2.6 left	86
4.29.2.7 pos_resu	86
4.29.2.8 result	86
4.29.2.9 right	86

4.29.2.10 s_end	86
4.29.2.11 s_num	87
4.29.2.12 s_pos	87
4.29.2.13 s_ptr	87
4.30 Pointer8B Class Reference	87
4.31 Project Class Reference	88
4.31.1 Detailed Description	88
4.31.2 Member Function Documentation	88
4.31.2.1 getNext()	88
4.31.2.2 init()	88
4.31.2.3 isEnd()	89
4.31.2.4 print()	89
4.31.3 Member Data Documentation	89
4.31.3.1 project_column_id	90
4.32 RequestColumn Struct Reference	90
4.32.1 Detailed Description	90
4.32.2 Member Data Documentation	90
4.32.2.1 name	90
4.33 RequestTable Struct Reference	90
4.33.1 Detailed Description	91
4.34 ResultTable Class Reference	91
4.34.1 Detailed Description	91
4.34.2 Member Function Documentation	91
4.34.2.1 dump()	92
4.34.2.2 getRC()	92
4.34.2.3 init()	92
4.34.2.4 print()	93
4.34.2.5 shut()	93
4.34.2.6 writeRC()	93
4.34.3 Member Data Documentation	93
4.34.3.1 buffer	93
4.34.3.2 buffer_size	94
4.34.3.3 column_number	94
4.34.3.4 column_type	94
4.34.3.5 row_capacity	94
4.34.3.6 row_length	94
4.34.3.7 row_number	94
4.35 RowTable Class Reference	94
4.35.1 Detailed Description	95
4.35.2 Constructor & Destructor Documentation	95
4.35.2.1 RowTable()	95
4.35.3 Member Function Documentation	96

4.35.3.1 del() [1/2]	96
4.35.3.2 del() [2/2]	96
4.35.3.3 finish()	96
4.35.3.4 getMStorage()	97
4.35.3.5 getRecordNum()	97
4.35.3.6 getRecordPtr()	97
4.35.3.7 getRPattern()	97
4.35.3.8 init()	98
4.35.3.9 insert() [1/2]	98
4.35.3.10 insert() [2/2]	98
4.35.3.11 loadData()	99
4.35.3.12 printData()	99
4.35.3.13 select() [1/2]	99
4.35.3.14 select() [2/2]	99
4.35.3.15 selectCol() [1/2]	100
4.35.3.16 selectCol() [2/2]	100
4.35.3.17 selectCols() [1/2]	101
4.35.3.18 selectCols() [2/2]	101
4.35.3.19 shut()	102
4.35.3.20 updateCol() [1/2]	102
4.35.3.21 updateCol() [2/2]	103
4.35.3.22 updateCols() [1/4]	103
4.35.3.23 updateCols() [2/4]	104
4.35.3.24 updateCols() [3/4]	104
4.35.3.25 updateCols() [4/4]	105
4.36 RPattern Class Reference	105
4.36.1 Detailed Description	106
4.36.2 Member Function Documentation	106
4.36.2.1 addColumn()	106
4.36.2.2 getColumnOffset()	106
4.36.2.3 getColumnType()	106
4.36.2.4 getRowSize()	107
4.36.2.5 init()	107
4.36.2.6 print()	107
4.36.2.7 reset()	108
4.36.2.8 shut()	108
4.37 Scan Class Reference	108
4.37.1 Detailed Description	109
4.37.2 Member Function Documentation	109
4.37.2.1 getNext()	109
4.37.2.2 init()	109
4.37.2.3 isEnd()	109

4.37.2.4 print()	110
4.38 SelectQuery Class Reference	110
4.38.1 Detailed Description	111
4.38.2 Member Data Documentation	111
4.38.2.1 database_id	111
4.38.2.2 from_number	111
4.38.2.3 from_table	111
4.38.2.4 groupby	111
4.38.2.5 groupby_number	111
4.38.2.6 having	111
4.38.2.7 orderby	112
4.38.2.8 orderby_number	112
4.38.2.9 select_column	112
4.38.2.10 select_number	112
4.38.2.11 where	112
4.39 Table Class Reference	112
4.39.1 Detailed Description	113
4.39.2 Constructor & Destructor Documentation	113
4.39.2.1 ~Table()	113
4.39.2.2 Table()	114
4.39.3 Member Function Documentation	114
4.39.3.1 addColumn()	114
4.39.3.2 addIndex()	114
4.39.3.3 del() [1/3]	114
4.39.3.4 del() [2/3]	115
4.39.3.5 del() [3/3]	115
4.39.3.6 finish()	116
4.39.3.7 getColumnRank()	116
4.39.3.8 getColumns()	116
4.39.3.9 getIndexRank()	116
4.39.3.10 getIndexs()	117
4.39.3.11 getRank()	117
4.39.3.12 getRecordNum()	117
4.39.3.13 getRecordPtr()	118
4.39.3.14 getTtype()	118
4.39.3.15 init()	118
4.39.3.16 insert() [1/2]	118
4.39.3.17 insert() [2/2]	119
4.39.3.18 loadData()	119
4.39.3.19 print()	119
4.39.3.20 printData()	119
4.39.3.21 select() [1/2]	119

4.39.3.22 select() [2/2]	120
4.39.3.23 selectCol() [1/2]	120
4.39.3.24 selectCol() [2/2]	121
4.39.3.25 selectCols() [1/2]	121
4.39.3.26 selectCols() [2/2]	122
4.39.3.27 shut()	122
4.39.3.28 updateCol() [1/2]	123
4.39.3.29 updateCol() [2/2]	124
4.39.3.30 updateCols() [1/3]	124
4.39.3.31 updateCols() [2/3]	125
4.39.3.32 updateCols() [3/3]	125
4.40 TypeCharN Class Reference	126
4.40.1 Detailed Description	126
4.40.2 Constructor & Destructor Documentation	127
4.40.2.1 TypeCharN()	127
4.40.3 Member Function Documentation	127
4.40.3.1 cmpEQ()	127
4.40.3.2 cmpGE()	127
4.40.3.3 cmpGT()	127
4.40.3.4 cmpLE()	128
4.40.3.5 cmpLT()	128
4.40.3.6 copy()	128
4.40.3.7 formatBin()	128
4.40.3.8 formatTxt()	129
4.41 TypeDate Class Reference	129
4.41.1 Detailed Description	129
4.41.2 Constructor & Destructor Documentation	129
4.41.2.1 TypeDate()	130
4.41.3 Member Function Documentation	130
4.41.3.1 cmpEQ()	130
4.41.3.2 cmpGE()	130
4.41.3.3 cmpGT()	130
4.41.3.4 cmpLE()	131
4.41.3.5 cmpLT()	131
4.41.3.6 copy()	131
4.41.3.7 formatBin()	131
4.41.3.8 formatTxt()	132
4.42 TypeDateTime Class Reference	132
4.42.1 Detailed Description	132
4.42.2 Constructor & Destructor Documentation	132
4.42.2.1 TypeDateTime()	133
4.42.3 Member Function Documentation	133

4.42.3.1 cmpEQ()	133
4.42.3.2 cmpGE()	133
4.42.3.3 cmpGT()	133
4.42.3.4 cmpLE()	134
4.42.3.5 cmpLT()	134
4.42.3.6 copy()	134
4.42.3.7 formatBin()	134
4.42.3.8 formatTxt()	135
4.43 TypeFloat32 Class Reference	135
4.43.1 Detailed Description	135
4.43.2 Constructor & Destructor Documentation	135
4.43.2.1 TypeFloat32()	136
4.43.3 Member Function Documentation	136
4.43.3.1 cmpEQ()	136
4.43.3.2 cmpGE()	136
4.43.3.3 cmpGT()	136
4.43.3.4 cmpLE()	137
4.43.3.5 cmpLT()	137
4.43.3.6 copy()	137
4.43.3.7 formatBin()	137
4.43.3.8 formatTxt()	138
4.44 TypeFloat64 Class Reference	138
4.44.1 Detailed Description	138
4.44.2 Constructor & Destructor Documentation	138
4.44.2.1 TypeFloat64()	139
4.44.3 Member Function Documentation	139
4.44.3.1 cmpEQ()	139
4.44.3.2 cmpGE()	139
4.44.3.3 cmpGT()	139
4.44.3.4 cmpLE()	140
4.44.3.5 cmpLT()	140
4.44.3.6 copy()	140
4.44.3.7 formatBin()	140
4.44.3.8 formatTxt()	141
4.45 TypeInt16 Class Reference	141
4.45.1 Detailed Description	141
4.45.2 Constructor & Destructor Documentation	141
4.45.2.1 TypeInt16()	142
4.45.3 Member Function Documentation	142
4.45.3.1 cmpEQ()	142
4.45.3.2 cmpGE()	142
4.45.3.3 cmpGT()	142

4.45.3.4 cmpLE()	143
4.45.3.5 cmpLT()	143
4.45.3.6 copy()	143
4.45.3.7 formatBin()	143
4.45.3.8 formatTxt()	144
4.46 TypeInt32 Class Reference	144
4.46.1 Detailed Description	144
4.46.2 Constructor & Destructor Documentation	144
4.46.2.1 TypeInt32()	145
4.46.3 Member Function Documentation	145
4.46.3.1 cmpEQ()	145
4.46.3.2 cmpGE()	145
4.46.3.3 cmpGT()	145
4.46.3.4 cmpLE()	146
4.46.3.5 cmpLT()	146
4.46.3.6 copy()	146
4.46.3.7 formatBin()	146
4.46.3.8 formatTxt()	147
4.47 TypeInt64 Class Reference	147
4.47.1 Detailed Description	147
4.47.2 Constructor & Destructor Documentation	147
4.47.2.1 TypeInt64()	148
4.47.3 Member Function Documentation	148
4.47.3.1 cmpEQ()	148
4.47.3.2 cmpGE()	148
4.47.3.3 cmpGT()	148
4.47.3.4 cmpLE()	149
4.47.3.5 cmpLT()	149
4.47.3.6 copy()	149
4.47.3.7 formatBin()	149
4.47.3.8 formatTxt()	150
4.48 TypeInt8 Class Reference	150
4.48.1 Detailed Description	150
4.48.2 Constructor & Destructor Documentation	150
4.48.2.1 TypeInt8()	151
4.48.3 Member Function Documentation	151
4.48.3.1 cmpEQ()	151
4.48.3.2 cmpGE()	151
4.48.3.3 cmpGT()	151
4.48.3.4 cmpLE()	152
4.48.3.5 cmpLT()	152
4.48.3.6 copy()	152

4.48.3.7 formatBin()	152
4.48.3.8 formatTxt()	153
4.49 TypeTime Class Reference	153
4.49.1 Detailed Description	153
4.49.2 Constructor & Destructor Documentation	153
4.49.2.1 TypeTime()	154
4.49.3 Member Function Documentation	154
4.49.3.1 cmpEQ()	154
4.49.3.2 cmpGE()	154
4.49.3.3 cmpGT()	154
4.49.3.4 cmpLE()	155
4.49.3.5 cmpLT()	155
4.49.3.6 copy()	155
4.49.3.7 formatBin()	155
4.49.3.8 formatTxt()	155
5 File Documentation	157
5.1 catalog.cc File Reference	157
5.2 catalog.h File Reference	157
5.2.1 Detailed Description	157
5.2.2 DESCRIPTION	158
5.3 datatype.h File Reference	158
5.3.1 Detailed Description	158
5.3.2 DESCRIPTION	159
5.3.3 Enumeration Type Documentation	159
5.3.3.1 TypeCode	159
5.4 errorlog.cc File Reference	159
5.4.1 Detailed Description	160
5.4.2 Description	160
5.4.3 Variable Documentation	160
5.4.3.1 EL_src_file_name	160
5.5 errorlog.h File Reference	160
5.5.1 Detailed Description	161
5.5.2 Description	161
5.5.3 Macro Definition Documentation	162
5.5.3.1 EL_LOG_INFO	162
5.5.3.2 EL_LOG_WARN	163
5.6 executor.cc File Reference	163
5.6.1 Detailed Description	163
5.6.2 DESCRIPTION	163
5.6.3 Function Documentation	163
5.6.3.1 compare()	163

5.6.3.2 Divide()	164
5.6.3.3 Sum()	164
5.6.4 Variable Documentation	165
5.6.4.1 orderby_cmp	165
5.7 executor.h File Reference	165
5.7.1 Detailed Description	166
5.7.2 DESCRIPTION	166
5.7.3 Enumeration Type Documentation	166
5.7.3.1 AggrerateMethod	166
5.7.3.2 CompareMethod	166
5.8 hashindex.cc File Reference	167
5.8.1 Detailed Description	167
5.8.2 DESCRIPTION	167
5.9 hashindex.h File Reference	167
5.9.1 Detailed Description	168
5.9.2 DESCRIPTION	168
5.10 mymemory.cc File Reference	168
5.10.1 Detailed Description	168
5.10.2 DESCRIPTION	169
5.11 mymemory.h File Reference	169
5.11.1 Detailed Description	169
5.11.2 DESCRIPTION	169
5.12 pbtrees.cc File Reference	170
5.12.1 Detailed Description	170
5.12.2 LICENSE	170
5.12.3 DESCRIPTION	170
5.13 pbtrees.h File Reference	170
5.13.1 Detailed Description	171
5.13.2 LICENSE	171
5.13.3 DESCRIPTION	171
5.14 pbtreesindex.h File Reference	171
5.14.1 Detailed Description	172
5.14.2 DESCRIPTION	172
5.15 rowtable.cc File Reference	172
5.15.1 Detailed Description	172
5.15.2 DESCRIPTION	172
5.16 rowtable.h File Reference	173
5.16.1 Detailed Description	173
5.16.2 DESCRIPTION	173
5.17 schema.h File Reference	173
5.17.1 Detailed Description	174
5.17.2 DESCRIPTION	174

5.17.3 Enumeration Type Documentation	174
5.17.3.1 ColumnType	174
5.17.3.2 IndexType	175
5.17.3.3 ObjectType	175
5.17.3.4 TableType	176
Index	177

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

BasicType	7
TypeCharN	126
TypeDate	129
TypeDateTime	132
TypeFloat32	135
TypeFloat64	138
TypeInt16	141
TypeInt32	144
TypeInt64	147
TypeInt8	150
TypeTime	153
bnode	11
Catalog	11
Condition	18
Conditions	19
ErrorLog	24
Executor	28
Groupby_struct	33
HashCell	34
HashCode_Ptr	36
HashInfo	43
HashTable	44
Key	65
Memory	66
MStorage	69
Object	71
Column	16
Database	20
Index	49
HashIndex	36
PbtreeIndex	79
Table	112
RowTable	94
Operator	73

Filter	29
Groupby	31
Join	61
Orderby	75
Project	88
Scan	108
Pbtree	78
pmtree	79
PbtreeInfo	85
Pointer8B	87
RequestColumn	90
RequestTable	90
ResultTable	91
RPattern	105
SelectQuery	110

Chapter 2

Class Index

2.1 Class List

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

BasicType	7
bnode	11
Catalog	11
Column	16
Condition	18
Conditions	19
Database	20
ErrorLog	
Array of source file names	24
Executor	28
Filter	29
Groupby	31
Groupby_struct	33
HashCell	34
HashCode_Ptr	36
HashIndex	36
HashInfo	43
HashTable	44
Index	49
Join	61
Key	65
Memory	66
MStorage	69
Object	71
Operator	73
Orderby	75
Pbtree	78
pbtree	79
PbtreeIndex	79
PbtreeInfo	85
Pointer8B	87
Project	88
RequestColumn	90
RequestTable	90
ResultTable	91

RowTable	94
RPattern	105
Scan	108
SelectQuery	110
Table	112
TypeCharN	126
TypeDate	129
TypeDateTime	132
TypeFloat32	135
TypeFloat64	138
TypeInt16	141
TypeInt32	144
TypeInt64	147
TypeInt8	150
TypeTime	153

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

catalog.cc	157
catalog.h	157
datatype.h	158
errorlog.cc	159
errorlog.h	160
executor.cc	163
executor.h	165
gcc_pf_p3.h	??
global.h	??
hashindex.cc	167
hashindex.h	167
hashtable.h	??
mymemory.cc	168
mymemory.h	169
nodepref.h	??
pbtree.cc	170
pbtree.h	170
pbtreeindex.h	171
rowtable.cc	172
rowtable.h	173
schema.h	173

Chapter 4

Class Documentation

4.1 BasicType Class Reference

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

Inheritance diagram for BasicType:

Public Member Functions

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

Protected Attributes

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

4.1.1 Detailed Description

definition of class [BasicType](#).

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BasicType()

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

constructor.

4.1.2.2 ~BasicType()

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

destructor.

4.1.3 Member Function Documentation

4.1.3.1 cmpEQ()

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

equal to.

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

4.1.3.2 cmpGE()

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

greater than or equal to

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

4.1.3.3 cmpGT()

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

greater than.

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

4.1.3.4 cmpLE()

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

less than or equal to.

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

4.1.3.5 cmpLT()

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

less than.

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

4.1.3.6 copy()

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

copy from data to dest.

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

4.1.3.7 formatBin()

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

extract bin format from data(txt) to dest.

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

4.1.3.8 formatTxt()

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

extract txt format from data(bin) to dest.

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

4.1.3.9 getTypeCode()

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

get type code of this data type.

4.1.3.10 getTypeSize()

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

get data size when stored in bin format.

4.1.4 Member Data Documentation

4.1.4.1 b_type_code

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

data type code

4.1.4.2 b_type_size

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

data type size

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

- [datatype.h](#)

4.2 bnode Class Reference

Collaboration diagram for bnode:

Public Member Functions

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

Public Attributes

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

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

- [pbtree.h](#)

4.3 Catalog Class Reference

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

Public Member Functions

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

4.3.1 Detailed Description

definition of class [Catalog](#).

4.3.2 Member Function Documentation

4.3.2.1 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

4.3.2.2 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
-------------	---------

Return values

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

4.3.2.3 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

4.3.2.4 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
<i>false</i>	failure

4.3.2.5 getObjById()

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

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

Parameters

$o \leftrightarrow$ _id	identifier of object
----------------------------	----------------------

Return values

!=	NULL available
==	NULL unavaliable, deleted or not exist

4.3.2.6 getObjByName()

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

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

Parameters

name	of an object
------	--------------

Return values

!=	NULL available
==	NULL unavaliable, deleted or not exist

4.3.2.7 init()

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

init operation.

4.3.2.8 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

4.3.2.9 print()

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

print the catalog

4.3.2.10 shut()

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

shut down, free all memory of Objects.

4.3.2.11 shutDatabase()

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

shutdown database

Parameters

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

Return values

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

Return values

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

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

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

4.4 Column Class Reference

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

Inheritance diagram for Column:

Collaboration diagram for Column:

Public Member Functions

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

4.4.1 Detailed Description

definition of class [Column](#).

4.4.2 Constructor & Destructor Documentation

4.4.2.1 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 identiier
<i>c_name</i>	column name
<i>c_type</i>	column type
<i>c_size</i>	data type size

4.4.2.2 ~Column()

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

destructor.

4.4.3 Member Function Documentation

4.4.3.1 finish()

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

finish column setting.

4.4.3.2 getCoffset()

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

get column offset.

4.4.3.3 getCSize()

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

get data dize.

4.4.3.4 getCType()

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

get column type.

4.4.3.5 getDataType()

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

get data type of the column

4.4.3.6 init()

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

init column.

4.4.3.7 print()

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

print column information

Reimplemented from [Object](#).

4.4.3.8 setCoffset()

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

get column offset.

4.4.3.9 shut()

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

shut down column.

Reimplemented from [Object](#).

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

- [schema.h](#)

4.5 Condition Struct Reference

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

Collaboration diagram for Condition:

Public Attributes

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

4.5.1 Detailed Description

definition of compare condition.

4.5.2 Member Data Documentation

4.5.2.1 column

[RequestColumn](#) Condition::column

which column

4.5.2.2 compare

[CompareMethod](#) Condition::compare

which method

4.5.2.3 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:

- [executor.h](#)

4.6 Conditions Struct Reference

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

Collaboration diagram for Conditions:

Public Attributes

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

4.6.1 Detailed Description

definition of conditions.

4.6.2 Member Data Documentation

4.6.2.1 condition

`Condition` Conditions::condition[4]

support maximum 4 & conditions

4.6.2.2 condition_num

`int` Conditions::condition_num

number of condition in use

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

- [executor.h](#)

4.7 Database Class Reference

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

Inheritance diagram for Database:

Collaboration diagram for Database:

Public Member Functions

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

4.7.1 Detailed Description

definition of class [Database](#).

4.7.2 Constructor & Destructor Documentation

4.7.2.1 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

4.7.2.2 ~Database()

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

destructor.

4.7.3 Member Function Documentation

4.7.3.1 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>flase</i>	failure

4.7.3.2 finish()

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

finish, important interface for son class

4.7.3.3 getTables()

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

get table identifier container.

4.7.3.4 init()

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

init, important interface for son class

4.7.3.5 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

4.7.3.6 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

4.7.3.7 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

4.7.3.8 print()

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

print database information

Reimplemented from [Object](#).

4.7.3.9 shut()

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

shut down this database, free all memory.

Reimplemented from [Object](#).

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

- [schema.h](#)

4.8 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](#) ()
- void [reset](#) ()
- void [log](#) (int level, const char *src_name, const int lineno,...)
- int [getErrorCode](#) (void)
- const char * [getErrorMsg](#) (void)

Static Public Member Functions

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

Static Public Attributes

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

4.8.1 Detailed Description

an array of source file names

4.8.2 Constructor & Destructor Documentation

4.8.2.1 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

4.8.2.2 ~ErrorLog()

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

destructor

4.8.3 Member Function Documentation

4.8.3.1 closeLog()

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

close the log file

4.8.3.2 flushLog()

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

flush the log file

4.8.3.3 getErrorCode()

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

return the last error code

4.8.3.4 getErrorMsg()

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

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

4.8.3.5 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

4.8.3.6 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

4.8.3.7 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

4.8.3.8 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

4.8.3.9 reset()

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

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

4.8.3.10 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
--------------	---------------------------

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

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

4.9 Executor Class Reference

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

Public Member Functions

- virtual int [exec](#) ([SelectQuery](#) *query, [ResultTable](#) *result)
- virtual int [close](#) ()

4.9.1 Detailed Description

definition of class executor.

4.9.2 Member Function Documentation

4.9.2.1 close()

```
int Executor::close ( ) [virtual]
```

close function.

Parameters

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

Return values

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

4.9.2.2 exec()

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

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

<i>>0</i>	number of result rows stored in result
<i><=0</i>	no more result

we need the Valiadation 'Y'--> result may print it as 89.

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

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

4.10 Filter Class Reference

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

Inheritance diagram for Filter:

Collaboration diagram for Filter:

Public Member Functions

- bool [init](#) ([Conditions](#) filter_given_conditions)
- bool [getNext](#) ()
- bool [isEnd](#) ()
- void [print](#) (int n)

Public Attributes

- int [filter_judge_num](#)
- [Condition](#) [filter_judge_condition](#) [4]

4.10.1 Detailed Description

definition of class [Filter](#).

4.10.2 Member Function Documentation

4.10.2.1 getNext()

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

[Filter](#)'s `getNext()`, everytime returns one result tuple.

Return values

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

Reimplemented from [Operator](#).

4.10.2.2 init()

```
bool Filter::init (
    Conditions filter_given_conditions )
```

init function for [Filter](#).

Parameters

<i>filter_given_conditions</i>	record 4 buckets to store conditions, every bucket stores 4 filter condition table[i]'s filter condition is stored in filter_given_condition[i]
--------------------------------	---

Return values

<i>true</i>	success.
<i>false</i>	fail.

4.10.2.3 isEnd()

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

free space used in [Filter Operator](#).

Return values

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

Reimplemented from [Operator](#).

4.10.2.4 print()

```
void Filter::print (
    int n ) [virtual]
```

print [Filter Operator](#) lies in [Operator](#) tree's layer.

Parameters

<i>n</i>	the <i>n</i> th layer Filter Operator lies at.
----------	--

Reimplemented from [Operator](#).

4.10.3 Member Data Documentation

4.10.3.1 filter_judge_condition

```
Condition Filter::filter_judge_condition[4]
```

record the corrsponding conditions given to corrsponding table.

4.10.3.2 filter_judge_num

```
int Filter::filter_judge_num
```

the number of filter conditions.

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

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

4.11 Groupby Class Reference

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

Inheritance diagram for Groupby:

Collaboration diagram for Groupby:

Public Member Functions

- bool [init](#) (int groupby_number, [RequestColumn](#) groupby_conditions[4], int groupby_compute_num, [RequestColumn](#) groupby_column[4])
- bool [getNext](#) ()
- bool [isEnd](#) ()
- void [print](#) (int n)

Additional Inherited Members

4.11.1 Detailed Description

definition of [Groupby Operator](#).

4.11.2 Member Function Documentation

4.11.2.1 getNext()

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

get one tuple of init result.

Return values

<i>true</i>	groupby suceess.
<i>false</i>	groupby failure.

Reimplemented from [Operator](#).

4.11.2.2 init()

```
bool Groupby::init (
    int groupby_number,
    RequestColumn groupby_conditions[4],
    int groupby_compute_num,
    RequestColumn groupby_column[4] )
```

init function of groupby, need to finish the construction of groupby_vector.

Parameters

<i>groupby_number</i>	the number of columns to groupby.
<i>groupby_conditions[4]</i>	the conditions of having column.
<i>groupby_compute_num</i>	the number of columns of having.
<i>groupby_column</i>	the name of groupby column.

Return values

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

4.11.2.3 isEnd()

```
bool Groupby::isEnd ( ) [virtual]
```

free the space allocated to groupby [Operator](#).

Return values

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

Reimplemented from [Operator](#).

4.11.2.4 print()

```
void Groupby::print (
    int n ) [virtual]
```

print the groupby [Operator](#) in [Operator](#) tree.

Parameters

<i>n</i>	the <i>n</i> th layer of Operator tree, where groupby lies at.
----------	--

Reimplemented from [Operator](#).

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

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

4.12 Groupby_struct Struct Reference

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

Public Attributes

- char [given_condition](#) [4][1024]
- char [value](#) [4][1024]

4.12.1 Detailed Description

definition of [Groupby_struct](#).

4.12.2 Member Data Documentation

4.12.2.1 given_condition

```
char Groupby_struct::given_condition[4][1024]
```

the array deals with groupby column and having column, if the groupby column has the same value, then we do not need to allocate a new space, we simply conduct SUM,AVG,MIN,MAX compute, if the groupby column is different from all the column's of given_condition, then we add it.

4.12.2.2 value

```
char Groupby_struct::value[4][1024]
```

store the having column's corresponding data.

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

- [executor.h](#)

4.13 HashCell Class Reference

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

Collaboration diagram for HashCell:

Public Attributes

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

4.13.1 Detailed Description

definition of class [HashCell](#).

4.13.2 Member Data Documentation

4.13.2.1 capacity

```
int HashCell::capacity
```

maximun number of array in this structure

4.13.2.2 ent

```
Hashcode_Ptr HashCell::ent
```

[Hashcode_Ptr](#), hit,decrease cache miss

4.13.2.3 ents

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

pointer of [Hashcode_Ptr](#) array

4.13.2.4 hc_num

```
int HashCell::hc_num
```

[Hashcode_Ptr](#) number in this [HashCell](#)

4.13.2.5 hc_union

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

if `hc_num == 1`,store one [Hashcode_Ptr](#); if `hc_num > 1`,store `num_2_or_more`

4.13.2.6 num_2_or_more

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

struct of a [Hashcode_Ptr](#) array,not hit,seach a array,not a link-list,decrease cache miss

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

- hashtable.h

4.14 Hashcode_Ptr Class Reference

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

Public Attributes

- `int64_t` [hash_code](#)
- `char *` [tuple](#)

4.14.1 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 conflict, a variable sized array of hash code and pointer is allocated.

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

4.14.2 Member Data Documentation

4.14.2.1 hash_code

```
int64_t Hashcode_Ptr::hash_code
```

hash_code of specific data

4.14.2.2 tuple

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

pointer of a record tuple

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

- hashtable.h

4.15 HashIndex Class Reference

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

Inheritance diagram for HashIndex:

Collaboration diagram for HashIndex:

Public Member Functions

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

Additional Inherited Members

4.15.1 Detailed Description

definition of [HashIndex](#).

4.15.2 Constructor & Destructor Documentation

4.15.2.1 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

4.15.3 Member Function Documentation

4.15.3.1 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

4.15.3.2 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](#).

4.15.3.3 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
-------------	---------

Return values

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

Reimplemented from [Index](#).

4.15.3.4 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](#).

4.15.3.5 init()

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

init hashindex, to calculate initial value.

Return values

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

Reimplemented from [Index](#).

4.15.3.6 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](#).

4.15.3.7 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](#).

4.15.3.8 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 set_ls
<i>result</i>	reference of record pointer

Return values

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

Reimplemented from [Index](#).

4.15.3.9 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 set_ls
<i>result</i>	reference of record pointer

Return values

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

Reimplemented from [Index](#).

4.15.3.10 print()

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

print hash index information.

Reimplemented from [Index](#).

4.15.3.11 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](#).

4.15.3.12 `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](#).

4.15.3.13 `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^cell_capbits
---------------------	--

4.15.3.14 shut()

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

free memory of hash table and other strucure.

Return values

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

Reimplemented from [Index](#).

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

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

4.16 HashInfo Struct Reference

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

Public Attributes

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

4.16.1 Detailed Description

definition of [HashInfo](#).

4.16.2 Member Data Documentation

4.16.2.1 hash

```
int64_t HashInfo::hash
```

hashcell position in [HashTable](#)

4.16.2.2 last

```
int HashInfo::last
```

retval of [HashTable](#) lookup

4.16.2.3 ppos

```
int HashInfo::ppos
```

pair with last, |last|

4.16.2.4 result

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

buffer for value of void *pointer

4.16.2.5 rnum

```
int HashInfo::rnum
```

current result number

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

- [hashindex.h](#)

4.17 HashTable Class Reference

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

Collaboration diagram for HashTable:

Public Member Functions

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

Public Attributes

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

4.17.1 Detailed Description

definition of class [HashTable](#).

4.17.2 Constructor & Destructor Documentation

4.17.2.1 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

4.17.2.2 ~HashTable()

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

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

4.17.3 Member Function Documentation

4.17.3.1 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

4.17.3.2 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

4.17.3.3 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

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

4.17.3.4 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

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

4.17.3.5 show()

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

display data in this hash table, for debug use.

4.17.3.6 utilization()

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

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

4.17.4 Member Data Documentation

4.17.4.1 avail

`Hashcode_Ptr* HashTable::avail`

pointer of next available [Hashcode_Ptr](#)

4.17.4.2 begin

`char* HashTable::begin`

start pointer of HashCells

4.17.4.3 end

`Hashcode_Ptr* HashTable::end`

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

4.17.4.4 estimated_duplicates_per_key

`double HashTable::estimated_duplicates_per_key`

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

4.17.4.5 estimated_num_distinct_keys

`int HashTable::estimated_num_distinct_keys`

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

4.17.4.6 free_header

`Hashcode_Ptr* HashTable::free_header[16]`

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

4.17.4.7 initial_array_size

`int HashTable::initial_array_size`

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

4.17.4.8 more_allocated

```
int HashTable::more_allocated
```

analysis of more memory allocated from g_memory

4.17.4.9 table

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

pointer of an array of [HashCell](#)

4.17.4.10 table_size

```
int HashTable::table_size
```

the number of HashCells in this table

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

- hashtable.h
- hashtable.cc

4.18 Index Class Reference

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

Inheritance diagram for Index:

Collaboration diagram for Index:

Public Member Functions

- [Index](#) (int64_t i_id, const char *i_name, [IndexType](#) i_type, [Key](#) &i_key)
- virtual [~Index](#) (void)
- virtual bool [init](#) (void)
- virtual bool [finish](#) (void)
- virtual bool [shut](#) (void)
- virtual bool [insert](#) (void *i_data, void *p_in)
- virtual bool [insert](#) (void *i_data[], void *p_in)
- virtual bool [del](#) (void *i_data)
- virtual bool [del](#) (void *i_data[])
- virtual bool [del](#) (void *i_data, void *p_del)
- virtual bool [del](#) (void *i_data[], void *p_del)
- virtual bool [update](#) (void *i_data, void *p_in)
- virtual bool [update](#) (void *i_data[], void *p_in)
- virtual bool [set_ls](#) (void *i_data1, void *i_data2, void *info)
- virtual bool [set_ls](#) (void *i_data1[], void *i_data2[], void *info)
- virtual bool [lookup](#) (void *i_data, void *&result)

- virtual bool [lookup](#) (void *i_data[], void *&result)
- virtual bool [lookup](#) (void *i_data, void *info, void *&result)
- virtual bool [lookup](#) (void *i_data[], void *info, void *&result)
- virtual 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 int64_t [tranToInt64](#) (void *i_data)
- virtual int64_t [tranToInt64](#) (void *i_data[])
- virtual void [print](#) (void)
- [IndexType](#) [getIType](#) (void)
- [Key](#) & [getKey](#) (void)
- virtual void [setIndexTid](#) (int64_t tid)
- virtual int64_t [getIndexTid](#) (void)

Protected Attributes

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

4.18.1 Detailed Description

definition of class [Index](#)

4.18.2 Constructor & Destructor Documentation

4.18.2.1 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

4.18.2.2 ~Index()

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

destructor.

4.18.3 Member Function Documentation

4.18.3.1 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](#).

4.18.3.2 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](#).

4.18.3.3 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](#).

4.18.3.4 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

4.18.3.5 finish()

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

finish index, important interface for son class

Reimplemented in [HashIndex](#).

4.18.3.6 getIKey()

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

get index key

4.18.3.7 getIndexTid()

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

get index in which table

4.18.3.8 getIType()

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

get index type

4.18.3.9 init()

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

init index, important interface for son class

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

4.18.3.10 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](#).

4.18.3.11 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](#).

4.18.3.12 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

4.18.3.13 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](#).

4.18.3.14 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

4.18.3.15 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](#).

4.18.3.16 print()

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

print index information

Reimplemented from [Object](#).

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

4.18.3.17 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](#).

4.18.3.18 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

4.18.3.19 scan_1() [2/2]

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

pepare 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

4.18.3.20 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

4.18.3.21 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

4.18.3.22 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](#).

4.18.3.23 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>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](#).

4.18.3.24 setIndexTid()

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

set index in which table

4.18.3.25 shut()

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

shut down the index, free memory.

Reimplemented from [Object](#).

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

4.18.3.26 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_data
----------------	----------------------

4.18.3.27 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_data
----------------	----------------------

4.18.4 Member Data Documentation

4.18.4.1 i_key

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

index keys

4.18.4.2 i_t_id

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

in which table

4.18.4.3 i_type

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

index type

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

- [schema.h](#)

4.19 Join Class Reference

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

Inheritance diagram for Join:

Collaboration diagram for Join:

Public Member Functions

- `bool` `init` (`Conditions` join_given_conditions)
- `bool` `getNext` ()
- `bool` `isEnd` ()
- `void` `print` (int n)

Public Attributes

- int [join_given_condition_num](#)
- std::vector< int64_t > [Column_id_array_prepare](#)
- std::vector< int64_t > [Column_id_array_join](#)
- int [join_lchild_rank](#) [4]
- int [join_rchild_rank](#) [4]
- [HashIndex](#) * [hx](#)
- std::vector< void * > [insert_hash_data](#)
- char * [lookup_hash_data](#)

4.19.1 Detailed Description

definition of [Join](#) node.

4.19.2 Member Function Documentation

4.19.2.1 getNext()

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

[getNext\(\)](#) function. this function call [hashindex.h](#) 's [lookup](#) to get one tuple of successful join.

Return values

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

Reimplemented from [Operator](#).

4.19.2.2 init()

```
bool Join::init (
    Conditions join\_given\_conditions )
```

init function. in init phase, we need to finish the construction of hash table, call [hashindex.h](#) 's [insert](#), [setCellCap](#), [addIndexDType](#),finsh.

Parameters

<i>join_given_conditions</i>	the condition of join Operator ,we only deal with two table's join.
------------------------------	---

Return values

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

4.19.2.3 isEnd()

```
bool Join::isEnd ( ) [virtual]
```

free the space allocated to [Join Operator](#).

Return values

<i>true</i>	free success
<i>false</i>	free fail

Reimplemented from [Operator](#).

4.19.2.4 print()

```
void Join::print (
    int n ) [virtual]
```

print the [Join Operator](#) in [Operator](#) tree.

Parameters

<i>n</i>	the nth layer Join Operator lies in.
----------	--

Reimplemented from [Operator](#).

4.19.3 Member Data Documentation**4.19.3.1 Column_id_array_join**

```
std::vector<int64_t> Join::Column_id_array_join
```

record the column_id need to join on right table(table B).

4.19.3.2 Column_id_array_prepare

```
std::vector<int64_t> Join::Column_id_array_prepare
```

record the column_id need to join on left table(table A).

4.19.3.3 hx

```
HashIndex* Join::hx
```

the index needed to build hash table.

4.19.3.4 insert_hash_data

```
std::vector<void *> Join::insert_hash_data
```

the hash_data to insert in hash table.

4.19.3.5 join_given_condition_num

```
int Join::join_given_condition_num
```

the number of join condition.

4.19.3.6 join_lchild_rank

```
int Join::join_lchild_rank[4]
```

the offset of left table's join column's column_id in lchild's column_id_array

4.19.3.7 join_rchild_rank

```
int Join::join_rchild_rank[4]
```

the offset of right table's join column's column_id in lchild's column_id_array

4.19.3.8 lookup_hash_data

```
char* Join::lookup_hash_data
```

right table's data, used to lookup in hash table.

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

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

4.20 Key Class Reference

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

Public Member Functions

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

4.20.1 Detailed Description

definition of class [Key](#).

4.20.2 Constructor & Destructor Documentation

4.20.2.1 Key()

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

constructor.

4.20.3 Member Function Documentation

4.20.3.1 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

4.20.3.2 getKey()

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

get key data.

4.20.3.3 operator=()

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

over write operator(=).

4.20.3.4 print()

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

print key information.

4.20.3.5 set()

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

set key.

Parameters

<i>in_key</i>	keys(column identifiers) in vector
---------------	------------------------------------

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

- [schema.h](#)

4.21 Memory Class Reference

Public Member Functions

- int [init](#) (int64_t total, int64_t mins)
- int64_t [alloc](#) (char *&p, int64_t size)
- int64_t [free](#) (char *p, int64_t size)
- int [shut](#) (void)
- int [print](#) (void)
- int [allocTableAddr](#) (char *&p)

4.21.1 Member Function Documentation

4.21.1.1 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

<i>==size</i>	successfully allocated from db memory
<i><=0</i>	failure

4.21.1.2 allocTableAddr()

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

alloc table address

Parameters

<i>p</i>	the pointer of memory to free
----------	-------------------------------

Return values

<i>0</i>	successfully free to db memory
----------	--------------------------------

4.21.1.3 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

<code>==size</code>	successfully free to db memory
<code><=0</code>	failure

4.21.1.4 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

<code>==0</code>	success
<code><0</code>	failure

4.21.1.5 print()

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

print memory usage.

4.21.1.6 shut()

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

free db memory to operate system.

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

- [mymemory.h](#)
- [mymemory.cc](#)

4.22 MStorage Class Reference

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

Public Member Functions

- bool [init](#) (int64_t record_size)
- int64_t [allocRow](#) (char *&pointer)
- char * [getRow](#) (int64_t record_rank)
- void [shut](#) (void)
- int64_t [getRecordNum](#) (void)

4.22.1 Detailed Description

definition of [MStorage](#), table storage manager.

4.22.2 Member Function Documentation

4.22.2.1 [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

4.22.2.2 [getRecordNum\(\)](#)

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

get the last record rank till now.

4.22.2.3 `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 <i>n</i> th row in the table
--------------------	----------------------------------

Return values

<i>!=NULL</i>	valid
<i>==NULL</i>	param error

4.22.2.4 `init()`

```
bool MStorage::init (
    int64_t record_size ) [inline]
```

make `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

4.22.2.5 `shut()`

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

shut down, free memory to system.

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

- [rowtable.h](#)

4.23 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)
- virtual bool [shut](#) (void)
- virtual void [print](#) (void)
- int64_t [getOid](#) (void)
- [ObjectType](#) [getOtype](#) (void)
- char * [getOname](#) (void)
- bool [changeName](#) (char *o_name)

4.23.1 Detailed Description

definition of [Object](#), basic element in database.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 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

4.23.3 Member Function Documentation

4.23.3.1 changeName()

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

change object name(not in use).

4.23.3.2 getOid()

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

get identifier of object.

4.23.3.3 getOname()

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

get object name.

4.23.3.4 getOtype()

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

get object type.

4.23.3.5 print()

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

print the object infomation.

Reimplemented in [Index](#), [Database](#), [Table](#), [Column](#), [HashIndex](#), and [PbtreeIndex](#).

4.23.3.6 shut()

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

shut down the object.

Reimplemented in [Index](#), [Database](#), [Table](#), [RowTable](#), [Column](#), [HashIndex](#), and [PbtreeIndex](#).

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

- [schema.h](#)

4.24 Operator Class Reference

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

Inheritance diagram for Operator:

Collaboration diagram for Operator:

Public Member Functions

- virtual bool [init](#) ()
- virtual bool [getNext](#) ()
- virtual bool [isEnd](#) ()
- virtual void [print](#) (int n)

Public Attributes

- void * [lchild](#)
- void * [rchild](#)
- void * [parent](#)
- [RPattern](#) [row_column_RPattern](#)
- [std::vector< int64_t >](#) [Column_id_array](#)
- char * [prev_buffer](#)
- char * [current_buffer](#)

4.24.1 Detailed Description

definition of Class [Operator](#).

4.24.2 Member Function Documentation

4.24.2.1 [getNext\(\)](#)

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

[getNext\(\)](#) function usually compute the result, call [getNext\(\)](#) once, it returns a result([current_buffer](#)) if it has finished, return false;

Reimplemented in [Orderby](#), [Groupby](#), [Join](#), [Project](#), [Filter](#), and [Scan](#).

4.24.2.2 init()

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

init function define the result format --> init row_column_RPattern define the result (tuple)'s column_id --> init Column_id_array allocate space for current_buffer prepare parameters for [getNext\(\)](#) function

4.24.2.3 isEnd()

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

[isEnd\(\)](#) function free space for current [Operator](#) usually parent's isEnd will call child's isEnd

Reimplemented in [OrderBy](#), [Groupby](#), [Join](#), [Project](#), [Filter](#), and [Scan](#).

4.24.2.4 print()

```
virtual void Operator::print (
    int n ) [inline], [virtual]
```

[print\(\)](#) function this function is not related to function, it's used to debug print the [Operator](#) tree to check the construction of [Operator](#) tree's correctness.

Parameters

<i>n</i>	means the Operator lines at the nth layer of Operator tree.
----------	---

Reimplemented in [OrderBy](#), [Groupby](#), [Join](#), [Project](#), [Filter](#), and [Scan](#).

4.24.3 Member Data Documentation

4.24.3.1 Column_id_array

```
std::vector<int64_t> Operator::Column_id_array
```

same function as [catalog.h Key](#), record column_id, corresponding to row_column_RPattern

4.24.3.2 current_buffer

```
char* Operator::current_buffer
```

the most important buffer, record result tuple

4.24.3.3 lchild

```
void* Operator::lchild
```

lchild, points out the lchild of current [Operator](#), lchild is also an [Operator](#)

4.24.3.4 parent

```
void* Operator::parent
```

parent, points out the parent node of current [Operator](#), not use here, become it's top-down

4.24.3.5 prev_buffer

```
char* Operator::prev_buffer
```

buffer to store child node's current_buffer, receive child's output result, actually rarely use

4.24.3.6 rchild

```
void* Operator::rchild
```

rchild, points out the lchild of current [Operator](#), lchild is also an [Operator](#)

4.24.3.7 row_column_RPattern

```
RPattern Operator::row_column_RPattern
```

define the output format, only record data_type and it's arrangement in output result

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

- [executor.h](#)

4.25 Orderby Class Reference

Inheritance diagram for Orderby:

Collaboration diagram for Orderby:

Public Member Functions

- bool [init](#) (int orderby_given_number, [RequestColumn](#) *orderby_given)
- bool [getNext](#) ()
- bool [isEnd](#) ()
- void [print](#) (int n)

Public Attributes

- int [orderby_offset](#) [4]
- [BasicType](#) * [orderby_data_type](#) [4]
- std::vector< char * > [orderby_vector](#)
- int [orderby_number](#)
- [RequestColumn](#) [orderby](#) [4]
- int [count](#)

4.25.1 Member Function Documentation

4.25.1.1 getNext()

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

get one tuple at a time from orderby init function.

Return values

<i>true</i>	get tuple success.
<i>false</i>	get tuple failure or the end.

Reimplemented from [Operator](#).

4.25.1.2 init()

```
bool Orderby::init (
    int orderby_given_number,
    RequestColumn * orderby_given )
```

init function, also need to fill in the qsort's parameter.

Parameters

<i>orderby_given_number</i>	number of columns to orderby.
<i>orderby_given</i>	name of columns to orderby.

Return values

<i>true</i>	orderby init success.
<i>false</i>	orderby init failure.

4.25.1.3 isEnd()

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

free the space allocated to orderby.

Return values

<i>true</i>	free success.
<i>false</i>	free fail.

Reimplemented from [Operator](#).

4.25.1.4 print()

```
void Orderby::print (
    int n ) [virtual]
```

print the layer of [Orderby Operator](#) in [Operator](#) tree.

Parameters

<i>n</i>	the n th layer of Operator tree(which Orderby Operator lies at).
----------	--

Reimplemented from [Operator](#).

4.25.2 Member Data Documentation

4.25.2.1 count

```
int Orderby::count
```

tag the current number of orderby's result tuple.

4.25.2.2 orderby

```
RequestColumn Orderby::orderby[4]
```

name of columns to orderby.

4.25.2.3 orderby_data_type

`BasicType* Orderby::orderby_data_type[4]`

the data_type of orderby column.

4.25.2.4 orderby_number

`int Orderby::orderby_number`

number of columns to orderby.

4.25.2.5 orderby_offset

`int Orderby::orderby_offset[4]`

the offset of orderby column in it's lchild's Column_id_array.

4.25.2.6 orderby_vector

`std::vector<char *> Orderby::orderby_vector`

vector used for qsort.

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

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

4.26 Pbtreet Class Reference

Public Member Functions

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

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

- [pbtreet.h](#)
- [pbtreet.cc](#)

4.27 pbtree Class Reference

Collaboration diagram for pbtree:

Public Member Functions

- `int` **init** (void)
- `int` **shut** (void)
- `void *` **lookup** (key_type key, int *pos)
- `void *` **get_recptr** (void *p, int pos)
- `key_type` **get_key** (void *p, int pos)
- `void **` **get_recptr_sp** (void *p, int pos)
- `void` **insert** (key_type key, void *ptr)
- `void` **del** (key_type key)
- `int` **scan** (void **p, int *spos, key_type startkey, key_type endkey, void *area[], int *num)
- `void` **print** ()
- `void` **check** (key_type *start, key_type *end)
- `int` **level** ()
- `void` **save** (char *filename)
- `void` **load** (char *filename)

Public Attributes

- `bnode *` **tree_root**
- `int` **root_level**

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

- [pbtree.h](#)
- [pbtree.cc](#)

4.28 PbtreetIndex Class Reference

Inheritance diagram for PbtreetIndex:

Collaboration diagram for PbtreetIndex:

Public Member Functions

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

Additional Inherited Members

4.28.1 Constructor & Destructor Documentation

4.28.1.1 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

4.28.2 Member Function Documentation

4.28.2.1 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](#).

4.28.2.2 init()

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

init [PbtreeIndex](#)

Return values

<i>true</i>	successfully initied
-------------	----------------------

Reimplemented from [Index](#).

4.28.2.3 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](#).

4.28.2.4 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 <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](#).

4.28.2.5 print()

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

print [Pbtree](#) index information.

print pbtree index information.

Reimplemented from [Index](#).

4.28.2.6 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](#).

4.28.2.7 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](#).

4.28.2.8 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

4.28.2.9 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](#).

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

- [pbtreetreeindex.h](#)
- [pbtreetreeindex.cc](#)

4.29 PbtreeInfo Struct Reference

```
#include <pbtreetreeindex.h>
```

Public Attributes

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

4.29.1 Detailed Description

definition of [PbtreeInfo](#).

4.29.2 Member Data Documentation

4.29.2.1 `area`

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

buffer for [Pbtree](#)

4.29.2.2 `cr_area`

```
int PbtreeInfo::cr_area
```

current area array pos in use

4.29.2.3 cr_resu

```
int PbtreeInfo::cr_resu
```

current result array pos in use

4.29.2.4 l_ptr

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

lookup elements ptr

4.29.2.5 le_resu

```
int PbtreeInfo::le_resu
```

len of current result

4.29.2.6 left

```
key_type PbtreeInfo::left
```

lookup key or scan left edge

4.29.2.7 pos_resu

```
int PbtreeInfo::pos_resu
```

pos in match array, init 0

4.29.2.8 result

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

buffer for element

4.29.2.9 right

```
key_type PbtreeInfo::right
```

scan right edge

4.29.2.10 s_end

```
int PbtreeInfo::s_end
```

scan tag, scan has more? 0 means more

4.29.2.11 s_num

```
int PbtreeInfo::s_num
```

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

4.29.2.12 s_pos

```
int PbtreeInfo::s_pos
```

scan pos in bnode, acquired by lookup_s

4.29.2.13 s_ptr

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

scan bnode ptr, acquired by lookup_s

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

- [pbtreeindex.h](#)

4.30 Pointer8B Class Reference

Public Member Functions

- [Pointer8B](#) & **operator=** (const void *ptr)
- [Pointer8B](#) & **operator=** (const [Pointer8B](#) &p)
- **operator void *** ()
- **operator char *** ()
- **operator struct bnode *** ()
- **operator struct bleaf *** ()
- **operator unsigned long long** ()
- void **print** (void)

Public Attributes

- unsigned long long **value**

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

- [pbtree.h](#)

4.31 Project Class Reference

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

Inheritance diagram for Project:

Collaboration diagram for Project:

Public Member Functions

- bool [init](#) (int project_given_number, [RequestColumn](#) *project_given_request)
- bool [getNext](#) ()
- bool [isEnd](#) ()
- void [print](#) (int n)

Public Attributes

- [Column](#) * [project_column_id](#) [4]

4.31.1 Detailed Description

definition of Class [Project](#)

4.31.2 Member Function Documentation

4.31.2.1 getNext()

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

[getNext\(\)](#) is used to get (lchild)'s result, and choose the requested column to project.

Return values

<i>true</i>	Project getNext() success
<i>false</i>	Project getNext() failure

Reimplemented from [Operator](#).

4.31.2.2 init()

```
bool Project::init (
    int project_given_number,
    RequestColumn * project_given_request )
```

init function of [Project](#).

Parameters

<i>project_given_number</i>	the number of Columns to project.
<i>project_given_request</i>	the name of Columns to project.

Return values

<i>true</i>	Project init success.
<i>false</i>	Project init fail.

4.31.2.3 isEnd()

```
bool Project::isEnd ( ) [virtual]
```

free the space allocated to [Project](#).

Return values

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

Reimplemented from [Operator](#).

4.31.2.4 print()

```
void Project::print (
    int n ) [virtual]
```

print the layer of [Project](#) in [Operator](#) tree.

Parameters

<i>n</i>	the n th layer Project Operator lies in Operator tree.
----------	--

Reimplemented from [Operator](#).

4.31.3 Member Data Documentation

4.31.3.1 project_column_id

`Column* Project::project_column_id[4]`

the column_id of project [Column](#).

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

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

4.32 RequestColumn Struct Reference

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

Public Attributes

- char [name](#) [128]
- [AggrerateMethod](#) [aggrerate_method](#)

4.32.1 Detailed Description

definition of request column.

4.32.2 Member Data Documentation

4.32.2.1 name

```
char RequestColumn::name[128]
```

name of column

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

- [executor.h](#)

4.33 RequestTable Struct Reference

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

Public Attributes

- char **name** [128]

4.33.1 Detailed Description

definition of request table.

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

- [executor.h](#)

4.34 ResultTable Class Reference

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

Collaboration diagram for ResultTable:

Public Member Functions

- int **init** ([BasicType](#) *col_types[], int col_num, int64_t capacity=1024 *1024)
- char * **getRC** (int row, int column)
- int **writeRC** (int row, int column, void *data)
- int **print** (void)
- int **dump** (FILE *fp)
- int **shut** (void)

Public Attributes

- int [column_number](#)
- [BasicType](#) ** [column_type](#)
- char * [buffer](#)
- int64_t [buffer_size](#)
- int [row_length](#)
- int [row_number](#)
- int [row_capacity](#)
- int * **offset**
- int **offset_size**

4.34.1 Detailed Description

definition of result table.

4.34.2 Member Function Documentation

4.34.2.1 dump()

```
int ResultTable::dump (
    FILE * fp )
```

write to file with FILE *fp

4.34.2.2 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

4.34.2.3 init()

```
int ResultTable::init (
    BaseType * col_types[],
    int col_num,
    int64_t capacity = 1024*1024 )
```

init alloc memory and set initial value @col_types array of column type pointers @col_num number of columns in this [ResultTable](#)

Parameters

<i>capacity</i>	buffer_size, power of 2
-----------------	-------------------------

Return values

<i>>0</i>	success
<i><=0</i>	failure

4.34.2.4 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
------------	------------------------

4.34.2.5 shut()

```
int ResultTable::shut (
    void )
```

free memory of this result table to g_memory

4.34.2.6 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 @data data pointer of a column

Return values

<i>!=NULL</i>	pointer of a column
<i>==NULL</i>	error

4.34.3 Member Data Documentation

4.34.3.1 buffer

```
char* ResultTable::buffer
```

pointer of buffer alloced from g_memory

4.34.3.2 buffer_size

```
int64_t ResultTable::buffer_size
```

size of buffer, power of 2

4.34.3.3 column_number

```
int ResultTable::column_number
```

columns number that a result row consist of

4.34.3.4 column_type

```
BasicType** ResultTable::column_type
```

each column data type

4.34.3.5 row_capacity

```
int ResultTable::row_capacity
```

maximum capacity of rows according to buffer size and length of row

4.34.3.6 row_length

```
int ResultTable::row_length
```

length per result row

4.34.3.7 row_number

```
int ResultTable::row_number
```

current usage of rows

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

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

4.35 RowTable Class Reference

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

Inheritance diagram for RowTable:

Collaboration diagram for RowTable:

Public Member Functions

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

4.35.1 Detailed Description

definition of class [RowTable](#).

4.35.2 Constructor & Destructor Documentation

4.35.2.1 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

4.35.3 Member Function Documentation

4.35.3.1 `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](#).

4.35.3.2 `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](#).

4.35.3.3 `finish()`

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

finish, leave it empty.

Reimplemented from [Table](#).

4.35.3.4 getMStorage()

```
MStorage& RowTable::getMStorage (
    void ) [inline]
```

get storage of table.

4.35.3.5 getRecordNum()

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

get the last record rank.

Reimplemented from [Table](#).

4.35.3.6 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](#).

4.35.3.7 getRPattern()

```
RPattern& RowTable::getRPattern (
    void ) [inline]
```

get pattern of table.

4.35.3.8 init()

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

init, leave it empty.

Reimplemented from [Table](#).

4.35.3.9 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](#).

4.35.3.10 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](#).

4.35.3.11 loadData()

```
bool RowTable::loadData (
    const char * filename ) [virtual]
```

load data of the table(not in use).

Reimplemented from [Table](#).

4.35.3.12 printData()

```
bool RowTable::printData (
    void ) [virtual]
```

print table data, for debug.

Reimplemented from [Table](#).

4.35.3.13 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](#).

4.35.3.14 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](#).

4.35.3.15 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](#).

4.35.3.16 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](#).

4.35.3.17 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](#).

4.35.3.18 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](#).

4.35.3.19 shut()

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

shut down r_pattern and r_storage, free their memory.

Reimplemented from [Table](#).

4.35.3.20 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](#).

4.35.3.21 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](#).

4.35.3.22 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](#).

4.35.3.23 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](#).

4.35.3.24 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](#).

4.35.3.25 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](#).

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

- [rowtable.h](#)
- [rowtable.cc](#)

4.36 RPattern Class Reference

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

Public Member Functions

- bool [init](#) (int64_t col_num)
- bool [addColumn](#) (BasicType *col_type)
- int64_t [getColumnOffset](#) (int64_t col_rank)
- BasicType * [getColumnType](#) (int64_t col_rank)
- void [reset](#) (void)
- void [shut](#) (void)
- int64_t [getRowSize](#) (void)
- int64_t [print](#) (char *r_ptr)

4.36.1 Detailed Description

definition of class [RPattern](#), describe row struture.

4.36.2 Member Function Documentation

4.36.2.1 addColumn()

```
bool RPattern::addColumn (
    BaseType * col_type ) [inline]
```

add column information.

Parameters

<i>col_type</i>	data type of the column
-----------------	-------------------------

4.36.2.2 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

≥ 0	valid offset
$= -1$	input error

4.36.2.3 getColumnType()

```
BaseType* RPattern::getColumnType (
    int64_t col_rank ) [inline]
```

get data type of column.

Parameters

<i>col_rank</i>	the <i>n</i> th column in the table
-----------------	-------------------------------------

Return values

<i>!=</i>	NULL valid pointer
<i>==</i>	NULL input error

4.36.2.4 getRowSize()

```
int64_t RPattern::getRowSize (
    void ) [inline]
```

get size of a row record.

Return values

<i>the</i>	size of a row record
------------	----------------------

4.36.2.5 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

4.36.2.6 print()

```
int64_t RPattern::print (
    char * r_ptr ) [inline]
```

print a row following this pattern.

Parameters

<code>r_ptr</code>	pointer of a row
--------------------	------------------

Return values

<code>==rp_row_size</code>	success
<code>!=rp_row_size</code>	error

4.36.2.7 reset()

```
void RPattern::reset (
    void ) [inline]
```

reset if addcolumn error happen.

4.36.2.8 shut()

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

shut down, free memory allocated from g_memory.

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

- [rowtable.h](#)

4.37 Scan Class Reference

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

Inheritance diagram for Scan:

Collaboration diagram for Scan:

Public Member Functions

- bool [init](#) (char *table_name)
- bool [getNext](#) ()
- bool [isEnd](#) ()
- void [print](#) (int n)

Additional Inherited Members

4.37.1 Detailed Description

definition of class [Scan](#).

4.37.2 Member Function Documentation

4.37.2.1 getNext()

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

[getNext\(\)](#) helps to record the current tuple ptr, and everytime copies one from origin table to current_buffer. current↔_buffer stores the [Scan Operator](#) result.

Return values

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

Reimplemented from [Operator](#).

4.37.2.2 init()

```
bool Scan::init (
    char * table_name )
```

[init\(\)](#) function.

Parameters

<i>table_name</i>	the name of the table to scan from
-------------------	------------------------------------

Return values

<i>true</i>	successfully init.
<i>false</i>	init failure.

4.37.2.3 isEnd()

```
bool Scan::isEnd ( ) [virtual]
```

free sapce of [Scan Operator](#).

Return values

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

Reimplemented from [Operator](#).

4.37.2.4 print()

```
void Scan::print (
    int n ) [virtual]
```

print [Operator](#) tree, whether [Operator](#) tree uses [Scan Operator](#), which layer [Scan Operator](#) lies at.

Parameters

<i>n</i>	the n th layer it lies at.
----------	----------------------------

print function.

Reimplemented from [Operator](#).

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

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

4.38 SelectQuery Class Reference

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

Collaboration diagram for SelectQuery:

Public Attributes

- [int64_t database_id](#)
- [int select_number](#)
- [RequestColumn select_column](#) [4]
- [int from_number](#)
- [RequestTable from_table](#) [4]
- [Conditions where](#)
- [int groupby_number](#)
- [RequestColumn groupby](#) [4]
- [Conditions having](#)
- [int orderby_number](#)
- [RequestColumn orderby](#) [4]

4.38.1 Detailed Description

definition of selectquery.

4.38.2 Member Data Documentation

4.38.2.1 database_id

```
int64_t SelectQuery::database_id
```

database to execute

4.38.2.2 from_number

```
int SelectQuery::from_number
```

number of tables to select from

4.38.2.3 from_table

```
RequestTable SelectQuery::from_table[4]
```

tables to select from, maximum 4

4.38.2.4 groupby

```
RequestColumn SelectQuery::groupby[4]
```

columns to groupby

4.38.2.5 groupby_number

```
int SelectQuery::groupby_number
```

number of columns to groupby

4.38.2.6 having

```
Conditions SelectQuery::having
```

groupby conditions

4.38.2.7 orderby

[RequestColumn](#) `SelectQuery::orderby[4]`

columns to orderby

4.38.2.8 orderby_number

`int SelectQuery::orderby_number`

number of columns to orderby

4.38.2.9 select_column

[RequestColumn](#) `SelectQuery::select_column[4]`

columns to select, maximum 4

4.38.2.10 select_number

`int SelectQuery::select_number`

number of column to select

4.38.2.11 where

[Conditions](#) `SelectQuery::where`

where meets conditions, maximum 4 & conditions

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

- [executor.h](#)

4.39 Table Class Reference

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

Inheritance diagram for Table:

Collaboration diagram for Table:

Public Member Functions

- [TableType](#) [getType](#) (void)
- virtual [~Table](#) (void)
- std::vector< int64_t > & [getColumns](#) (void)
- std::vector< int64_t > & [getIndexes](#) (void)
- int64_t [getColumnRank](#) (int64_t c_id)
- int64_t [getIndexRank](#) (int64_t i_id)
- int64_t [getRank](#) (std::vector< int64_t > &vec, int64_t id)
- [Table](#) (int64_t t_id, const char *t_name, [TableType](#) t_type)
- virtual void [print](#) (void)
- virtual bool [init](#) (void)
- virtual bool [addColumn](#) (int64_t column_id)
- virtual bool [addIndex](#) (int64_t index_id)
- virtual bool [finish](#) (void)
- virtual bool [shut](#) (void)
- virtual bool [selectCol](#) (int64_t record_rank, int64_t column_rank, char *dest)
- virtual bool [selectCols](#) (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *dest)
- virtual bool [select](#) (int64_t record_rank, char *dest)
- virtual bool [selectCol](#) (char *row_pointer, int64_t column_rank, char *dest)
- virtual bool [selectCols](#) (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *dest)
- virtual bool [select](#) (char *row_pointer, char *dest)
- virtual bool [updateCol](#) (int64_t record_rank, int64_t column_rank, char *source)
- virtual bool [updateCol](#) (char *row_pointer, int64_t column_rank, char *source)
- virtual bool [updateCols](#) (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *source)
- virtual bool [updateCols](#) (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *source)
- virtual bool [updateCols](#) (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *source[])
- virtual bool [updateCols](#) (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *source[])
- virtual bool [del](#) (int64_t record_rank)
- virtual bool [del](#) (char *row_pointer)
- virtual bool [del](#) (char *columns[])
- virtual bool [insert](#) (char *source)
- virtual bool [insert](#) (char *columns[])
- virtual int64_t [getRecordNum](#) (void)
- virtual void * [getRecordPtr](#) (int64_t row_rank)
- virtual bool [loadData](#) (const char *filename)
- virtual bool [printData](#) (void)

4.39.1 Detailed Description

definition of class [Table](#).

4.39.2 Constructor & Destructor Documentation

4.39.2.1 ~Table()

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

destructor.

4.39.2.2 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

4.39.3 Member Function Documentation

4.39.3.1 addColumn()

```
virtual bool Table::addColumn (
    int64_t column_id ) [inline], [virtual]
```

add column identifier to this table.

4.39.3.2 addIndex()

```
virtual bool Table::addIndex (
    int64_t index_id ) [inline], [virtual]
```

add index identifier to this table.

4.39.3.3 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
-------------	---------

Return values

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

4.39.3.4 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](#).

4.39.3.5 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](#).

4.39.3.6 finish()

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

finish, important interface for son class

Reimplemented in [RowTable](#).

4.39.3.7 getColumnRank()

```
int64_t Table::getColumnRank (
    int64_t c_id ) [inline]
```

get column rank in this table

Parameters

$c \leftrightarrow$ _id	column identifier
----------------------------	-------------------

Return values

≥ 0	0 valid rank
< 0	invalid, not exist

4.39.3.8 getColumns()

```
std::vector< int64_t >& Table::getColumns (
    void ) [inline]
```

get column identifier vector.

Return values

<i>vector</i>	of column identifiers
---------------	-----------------------

4.39.3.9 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

4.39.3.10 getIndexes()

```
std::vector< int64_t >& Table::getIndexes (
    void ) [inline]
```

get index identifier vector.

4.39.3.11 getRank()

```
int64_t Table::getRank (
    std::vector< int64_t > & vec,
    int64_t id ) [inline]
```

get rank in a vector

Parameters

vec	vector to search in
id	object identifier

Return values

\geq	0 valid rank
< 0	invalid, not exist

4.39.3.12 getRecordNum()

```
virtual int64_t Table::getRecordNum (
    void ) [inline], [virtual]
```

get record number.

Reimplemented in [RowTable](#).

4.39.3.13 getRecordPtr()

```
virtual void* Table::getRecordPtr (
    int64_t row_rank ) [inline], [virtual]
```

get record pointer.

Reimplemented in [RowTable](#).

4.39.3.14 getTtype()

```
TableType Table::getTtype (
    void ) [inline]
```

get table type.

4.39.3.15 init()

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

init, important interface for son class

Reimplemented in [RowTable](#).

4.39.3.16 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](#).

4.39.3.17 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](#).

4.39.3.18 loadData()

```
virtual bool Table::loadData (
    const char * filename ) [inline], [virtual]
```

load data(not in use).

Reimplemented in [RowTable](#).

4.39.3.19 print()

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

print table information.

Reimplemented from [Object](#).

4.39.3.20 printData()

```
virtual bool Table::printData (
    void ) [inline], [virtual]
```

print data in table.

Reimplemented in [RowTable](#).

4.39.3.21 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](#).

4.39.3.22 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](#).

4.39.3.23 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](#).

4.39.3.24 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](#).

4.39.3.25 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](#).

4.39.3.26 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](#).

4.39.3.27 shut()

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

shut, important interface for son class

Reimplemented from [Object](#).

Reimplemented in [RowTable](#).

4.39.3.28 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](#).

4.39.3.29 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](#).

4.39.3.30 updateCols() [1/3]

```
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](#).

4.39.3.31 updateCols() [2/3]

```
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](#).

4.39.3.32 updateCols() [3/3]

```
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](#).

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

- [schema.h](#)

4.40 TypeCharN Class Reference

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

Inheritance diagram for TypeCharN:

Collaboration diagram for TypeCharN:

Public Member Functions

- [TypeCharN](#) (int64_t typesize)
- **TypeCharN** (TypeCode typecode=[CHARN_TC](#), int64_t typesize=32)
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.40.1 Detailed Description

definition of class [TypeCharN](#), please refer to [BasicType](#), it's same.

4.40.2 Constructor & Destructor Documentation

4.40.2.1 TypeCharN()

```
TypeCharN::TypeCharN (
    int64_t typesize ) [inline]
```

constructor.

4.40.3 Member Function Documentation

4.40.3.1 cmpEQ()

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

equal to.

Reimplemented from [BasicType](#).

4.40.3.2 cmpGE()

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

greater than or equal to

Reimplemented from [BasicType](#).

4.40.3.3 cmpGT()

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

greater than.

Reimplemented from [BasicType](#).

4.40.3.4 cmpLE()

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

less than or equal to.

Reimplemented from [BasicType](#).

4.40.3.5 cmpLT()

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

less than.

Reimplemented from [BasicType](#).

4.40.3.6 copy()

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

copy from data to dest.

Reimplemented from [BasicType](#).

4.40.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.40.3.8 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:

- [datatype.h](#)

4.41 TypeDate Class Reference

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

Inheritance diagram for TypeDate:

Collaboration diagram for TypeDate:

Public Member Functions

- [TypeDate](#) ([TypeCode](#) typecode=[DATE_TC](#), int64_t typesize=sizeof(time_t))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.41.1 Detailed Description

definition of class [TypeDate](#), please refer to [BasicType](#), it's same.

4.41.2 Constructor & Destructor Documentation

4.41.2.1 TypeDate()

```
TypeDate::TypeDate (
    TypeCode typecode = DATE_TC,
    int64_t typesize = sizeof(time_t) ) [inline]
```

constructor.

4.41.3 Member Function Documentation

4.41.3.1 cmpEQ()

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

equal to.

Reimplemented from [BasicType](#).

4.41.3.2 cmpGE()

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

greater than or equal to

Reimplemented from [BasicType](#).

4.41.3.3 cmpGT()

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

greater than.

Reimplemented from [BasicType](#).

4.41.3.4 cmpLE()

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

less than or equal to.

Reimplemented from [BasicType](#).

4.41.3.5 cmpLT()

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

less than.

Reimplemented from [BasicType](#).

4.41.3.6 copy()

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

copy from data to dest.

Reimplemented from [BasicType](#).

4.41.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.41.3.8 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:

- [datatype.h](#)

4.42 TypeDateTime Class Reference

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

Inheritance diagram for TypeDateTime:

Collaboration diagram for TypeDateTime:

Public Member Functions

- [TypeDateTime](#) ([TypeCode](#) typecode=[DATETIME_TC](#), int64_t typesize=sizeof(time_t))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.42.1 Detailed Description

definition of class [TypeDateTime](#), please refer to [BasicType](#), it's same.

4.42.2 Constructor & Destructor Documentation

4.42.2.1 TypeDateTime()

```
TypeDateTime::TypeDateTime (
    TypeCode typecode = DATETIME_TC,
    int64_t typesize = sizeof(time_t) ) [inline]
```

constructor.

4.42.3 Member Function Documentation

4.42.3.1 cmpEQ()

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

equal to.

Reimplemented from [BasicType](#).

4.42.3.2 cmpGE()

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

greater than or equal to

Reimplemented from [BasicType](#).

4.42.3.3 cmpGT()

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

greater than.

Reimplemented from [BasicType](#).

4.42.3.4 cmpLE()

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

less than or equal to.

Reimplemented from [BasicType](#).

4.42.3.5 cmpLT()

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

less than.

Reimplemented from [BasicType](#).

4.42.3.6 copy()

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

copy from data to dest.

Reimplemented from [BasicType](#).

4.42.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.42.3.8 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:

- [datatype.h](#)

4.43 TypeFloat32 Class Reference

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

Inheritance diagram for TypeFloat32:

Collaboration diagram for TypeFloat32:

Public Member Functions

- [TypeFloat32](#) ([TypeCode](#) typecode=[FLOAT32_TC](#), int64_t typesize=sizeof(float))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.43.1 Detailed Description

definition of class [TypeFloat32](#), please refer to [BasicType](#), it's same.

4.43.2 Constructor & Destructor Documentation

4.43.2.1 TypeFloat32()

```
TypeFloat32::TypeFloat32 (
    TypeCode typecode = Float32_TC,
    int64_t typesize = sizeof(float) ) [inline]
```

constructor.

4.43.3 Member Function Documentation

4.43.3.1 cmpEQ()

```
bool TypeFloat32::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.43.3.2 cmpGE()

```
bool TypeFloat32::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.43.3.3 cmpGT()

```
bool TypeFloat32::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.43.3.4 cmpLE()

```
bool TypeFloat32::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.43.3.5 cmpLT()

```
bool TypeFloat32::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.43.3.6 copy()

```
int TypeFloat32::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.43.3.7 formatBin()

```
int TypeFloat32::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.43.3.8 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:

- [datatype.h](#)

4.44 TypeFloat64 Class Reference

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

Inheritance diagram for TypeFloat64:

Collaboration diagram for TypeFloat64:

Public Member Functions

- [TypeFloat64](#) ([TypeCode](#) typecode=[FLOAT64_TC](#), int64_t typesize=sizeof(double))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.44.1 Detailed Description

definition of class [TypeFloat64](#), please refer to [BasicType](#), it's same.

4.44.2 Constructor & Destructor Documentation

4.44.2.1 TypeFloat64()

```
TypeFloat64::TypeFloat64 (
    TypeCode typecode = FLOAT64_TC,
    int64_t typesize = sizeof(double) ) [inline]
```

constructor.

4.44.3 Member Function Documentation

4.44.3.1 cmpEQ()

```
bool TypeFloat64::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.44.3.2 cmpGE()

```
bool TypeFloat64::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.44.3.3 cmpGT()

```
bool TypeFloat64::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.44.3.4 cmpLE()

```
bool TypeFloat64::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.44.3.5 cmpLT()

```
bool TypeFloat64::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.44.3.6 copy()

```
int TypeFloat64::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.44.3.7 formatBin()

```
int TypeFloat64::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.44.3.8 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:

- [datatype.h](#)

4.45 TypeInt16 Class Reference

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

Inheritance diagram for TypeInt16:

Collaboration diagram for TypeInt16:

Public Member Functions

- [TypeInt16](#) ([TypeCode](#) typecode=[INT16_TC](#), int64_t typesize=sizeof(int16_t))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.45.1 Detailed Description

definition of class [TypeInt16](#), please refer to [BasicType](#), it's same.

4.45.2 Constructor & Destructor Documentation

4.45.2.1 TypeInt16()

```
TypeInt16::TypeInt16 (
    TypeCode typecode = INT16_TC,
    int64_t typesize = sizeof(int16_t) ) [inline]
```

constructor.

4.45.3 Member Function Documentation

4.45.3.1 cmpEQ()

```
bool TypeInt16::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.45.3.2 cmpGE()

```
bool TypeInt16::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.45.3.3 cmpGT()

```
bool TypeInt16::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.45.3.4 cmpLE()

```
bool TypeInt16::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.45.3.5 cmpLT()

```
bool TypeInt16::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.45.3.6 copy()

```
int TypeInt16::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.45.3.7 formatBin()

```
int TypeInt16::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.45.3.8 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:

- [datatype.h](#)

4.46 TypeInt32 Class Reference

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

Inheritance diagram for TypeInt32:

Collaboration diagram for TypeInt32:

Public Member Functions

- [TypeInt32](#) ([TypeCode](#) typecode=[INT32_TC](#), int64_t typesize=sizeof(int32_t))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.46.1 Detailed Description

definition of class [TypeInt32](#), please refer to [BasicType](#), it's same.

4.46.2 Constructor & Destructor Documentation

4.46.2.1 TypeInt32()

```
TypeInt32::TypeInt32 (
    TypeCode typecode = INT32_TC,
    int64_t typesize = sizeof(int32_t) ) [inline]
```

constructor.

4.46.3 Member Function Documentation

4.46.3.1 cmpEQ()

```
bool TypeInt32::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.46.3.2 cmpGE()

```
bool TypeInt32::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.46.3.3 cmpGT()

```
bool TypeInt32::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.46.3.4 cmpLE()

```
bool TypeInt32::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.46.3.5 cmpLT()

```
bool TypeInt32::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.46.3.6 copy()

```
int TypeInt32::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.46.3.7 formatBin()

```
int TypeInt32::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.46.3.8 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:

- [datatype.h](#)

4.47 TypeInt64 Class Reference

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

Inheritance diagram for TypeInt64:

Collaboration diagram for TypeInt64:

Public Member Functions

- [TypeInt64](#) ([TypeCode](#) typecode=[INT64_TC](#), int64_t typesize=sizeof(int64_t))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.47.1 Detailed Description

definition of class [TypeInt64](#), please refer to [BasicType](#), it's same.

4.47.2 Constructor & Destructor Documentation

4.47.2.1 TypeInt64()

```
TypeInt64::TypeInt64 (
    TypeCode typecode = INT64_TC,
    int64_t typesize = sizeof(int64_t) ) [inline]
```

constructor.

4.47.3 Member Function Documentation

4.47.3.1 cmpEQ()

```
bool TypeInt64::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.47.3.2 cmpGE()

```
bool TypeInt64::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.47.3.3 cmpGT()

```
bool TypeInt64::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.47.3.4 cmpLE()

```
bool TypeInt64::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.47.3.5 cmpLT()

```
bool TypeInt64::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.47.3.6 copy()

```
int TypeInt64::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.47.3.7 formatBin()

```
int TypeInt64::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.47.3.8 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:

- [datatype.h](#)

4.48 TypeInt8 Class Reference

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

Inheritance diagram for TypeInt8:

Collaboration diagram for TypeInt8:

Public Member Functions

- [TypeInt8](#) ([TypeCode](#) typecode=[INT8_TC](#), int64_t typesize=sizeof(int8_t))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.48.1 Detailed Description

definition of class [TypeInt8](#), please refer to [BasicType](#), it's same.

4.48.2 Constructor & Destructor Documentation

4.48.2.1 TypeInt8()

```
TypeInt8::TypeInt8 (
    TypeCode typecode = INT8_TC,
    int64_t typesize = sizeof(int8_t) ) [inline]
```

constructor.

4.48.3 Member Function Documentation

4.48.3.1 cmpEQ()

```
bool TypeInt8::cmpEQ (
    void * data1,
    void * data2 ) [inline], [virtual]
```

equal to.

Reimplemented from [BasicType](#).

4.48.3.2 cmpGE()

```
bool TypeInt8::cmpGE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than or equal to

Reimplemented from [BasicType](#).

4.48.3.3 cmpGT()

```
bool TypeInt8::cmpGT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

greater than.

Reimplemented from [BasicType](#).

4.48.3.4 cmpLE()

```
bool TypeInt8::cmpLE (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than or equal to.

Reimplemented from [BasicType](#).

4.48.3.5 cmpLT()

```
bool TypeInt8::cmpLT (
    void * data1,
    void * data2 ) [inline], [virtual]
```

less than.

Reimplemented from [BasicType](#).

4.48.3.6 copy()

```
int TypeInt8::copy (
    void * dest,
    void * data ) [inline], [virtual]
```

copy from data to dest.

Reimplemented from [BasicType](#).

4.48.3.7 formatBin()

```
int TypeInt8::formatBin (
    void * dest,
    void * data ) [inline], [virtual]
```

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.48.3.8 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:

- [datatype.h](#)

4.49 TypeTime Class Reference

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

Inheritance diagram for TypeTime:

Collaboration diagram for TypeTime:

Public Member Functions

- [TypeTime](#) ([TypeCode](#) typecode=[TIME_TC](#), int64_t typesize=sizeof(time_t))
- int [copy](#) (void *dest, void *data)
- int [formatTxt](#) (void *dest, void *data)
- int [formatBin](#) (void *dest, void *data)
- bool [cmpLT](#) (void *data1, void *data2)
- bool [cmpLE](#) (void *data1, void *data2)
- bool [cmpEQ](#) (void *data1, void *data2)
- bool [cmpGT](#) (void *data1, void *data2)
- bool [cmpGE](#) (void *data1, void *data2)

Additional Inherited Members

4.49.1 Detailed Description

definition of class [TypeTime](#), please refer to [BasicType](#), it's same.

4.49.2 Constructor & Destructor Documentation

4.49.2.1 TypeTime()

```
TypeTime::TypeTime (
    TypeCode typecode = TIME\_TC,
    int64_t typesize = sizeof(time_t) ) [inline]
```

constructor.

4.49.3 Member Function Documentation

4.49.3.1 cmpEQ()

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

equal to.

Reimplemented from [BasicType](#).

4.49.3.2 cmpGE()

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

greater than or equal to

Reimplemented from [BasicType](#).

4.49.3.3 cmpGT()

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

greater than.

Reimplemented from [BasicType](#).

4.49.3.4 cmpLE()

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

less than or equal to.

Reimplemented from [BasicType](#).

4.49.3.5 cmpLT()

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

less than.

Reimplemented from [BasicType](#).

4.49.3.6 copy()

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

copy from data to dest.

Reimplemented from [BasicType](#).

4.49.3.7 formatBin()

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

extract bin format from data(txt) to dest.

Reimplemented from [BasicType](#).

4.49.3.8 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:

- [datatype.h](#)

Chapter 5

File Documentation

5.1 catalog.cc File Reference

```
#include "catalog.h"
```

Include dependency graph for catalog.cc:

5.2 catalog.h File Reference

```
#include <vector>
#include <unordered_map>
#include "schema.h"
#include "rowtable.h"
#include "hashindex.h"
#include "pbtreetreeindex.h"
```

Include dependency graph for catalog.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Catalog](#)

Variables

- [Catalog](#) `g_catalog`

5.2.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.2.2 DESCRIPTION

this file provides an element container of database, which can be described as the stem of a tree with this [Catalog](#), the only one in global system, you can access all elements of system

basic usage:

(1) you use `g_catalog.createXXX[Database,Table,Column,Index]` to create objects. (2) you can get the pointer of `[Database,Table,Column,Index]` by call `g_catalog.getObjById/Name` (3) in `[Database,Table,Column,Index]`, you can define the relation of different objects by adding operation in those object (4) when all relation defined in database, you must call `initDatabase` to actually getting the database prepared to be used. (5) `shut` will get everything shutup can free `shutDatabase` will only shutup the selected database (6) for more tips, you may learn from `debug_catalog.cc`

5.3 datatype.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <time.h>
```

Include dependency graph for `datatype.h`: This graph shows which files directly or indirectly include this file:

Classes

- class [BasicType](#)
- class [TypeInt8](#)
- class [TypeInt16](#)
- class [TypeInt32](#)
- class [TypeInt64](#)
- class [TypeFloat32](#)
- class [TypeFloat64](#)
- class [TypeCharN](#)
- class [TypeDate](#)
- class [TypeTime](#)
- class [TypeDateTime](#)

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` }

5.3.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.3.2 DESCRIPTION

all datatype supported by this system

5.3.3 Enumeration Type Documentation

5.3.3.1 TypeCode

enum [TypeCode](#)

data type code.

Enumerator

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

5.4 errorlog.cc File Reference

```
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <stdarg.h>
#include <execinfo.h>
#include <cxxabi.h>
#include "errorlog.h"
Include dependency graph for errorlog.cc:
```

Macros

- `#define EL_TOTAL_FILES (sizeof(EL_src_file_name)/sizeof(char*))`

Variables

- `const char * EL_src_file_name []`
- `ErrorLog_Thread_local * thread_el = NULL`

5.4.1 Detailed Description

Author

Shimin Chen chensm@ict.ac.cn

Version

0.1

5.4.2 Description

This file provides the error handling and logging utility.

5.4.3 Variable Documentation

5.4.3.1 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  
}
```

5.5 errorlog.h File Reference

```
#include <pthread.h>  
#include <stdio.h>  
#include <string>  
#include <unordered_map>  
#include <time.h>
```

Include dependency graph for errorlog.h: This graph shows which files directly or indirectly include this file:

Classes

- class [ErrorLog](#)
an array of source file names

Macros

- `#define EL_DEBUG 1`
- `#define EL_INFO 2`
- `#define EL_WARN 3`
- `#define EL_ERROR 4`
- `#define EL_SERIOUS 5`
- `#define EL_LEVEL_COMPILE EL_INFO`
- `#define EL_LOG_DEBUG(...)`
- `#define EL_LOG_INFO(...)`
- `#define EL_LOG_WARN(...)`
- `#define EL_LOG_ERROR(...) thread_el->log(EL_ERROR, __FILE__, __LINE__, __VA_ARGS__)`
- `#define EL_LOG_SERIOUS(...) thread_el->log(EL_SERIOUS, __FILE__, __LINE__, __VA_ARGS__)`
- `#define EL_RESET() (thread_el->reset())`
- `#define EL_ERRCODE() (thread_el->getErrorCode())`
- `#define EL_ERRMSG() (thread_el->getErrMsg())`
- `#define EL_ASSERT(t)`
- `#define EL_ERROR_CODE(fileid, lineno) ((fileid)*100000 + (lineno))`
- `#define EL_GET_FILEID(errcode) ((errcode)/100000)`
- `#define EL_GET_LINENO(errcode) ((errcode)%100000)`
- `#define EL_OK (0)`
- `#define EL_BAD_FILEID (99999999)`
- `#define EL_GET_FILENAME(errcode) (ErrorLog::id2Name(EL_GET_FILEID(errcode)))`
- `#define _Thread_local __thread`

Variables

- `const char * EL_src_file_name []`
- `_Thread_local ErrorLog * thread_el`

5.5.1 Detailed Description

Author

Shimin Chen chensm@ict.ac.cn

Version

0.1

5.5.2 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();
```

5.5.3 Macro Definition Documentation

5.5.3.1 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)
```

5.5.3.2 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)
```

5.6 executor.cc File Reference

```
#include "executor.h"
```

Include dependency graph for executor.cc:

Functions

- bool [compare](#) ([BasicType](#) *type, void *data1, char *value, [CompareMethod](#) method)
- bool [Sum](#) ([BasicType](#) *data_type, void *data1, void *data2)
- bool [Divide](#) ([BasicType](#) *data_type, void *data1, void *data2)

Variables

- [Orderby](#) * [orderby_cmp](#)

5.6.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.6.2 DESCRIPTION

definition of executor

5.6.3 Function Documentation

5.6.3.1 compare()

```
bool compare (
    BasicType * type,
    void * data1,
    char * value,
    CompareMethod method )
```

the function is used in filter.

Parameters

<i>type</i>	point out the BasicType .
<i>data1</i>	operand1
<i>value</i>	we first need to transfer char* to correst BasicType , operand2;
<i>method</i>	the compare type LT/LE/EQ/NE/GT/GE

Return values

<i>true</i>	compare result
<i>false</i>	compare result --> the opposite

5.6.3.2 Divide()

```
bool Divide (
    BasicType * data_type,
    void * data1,
    void * data2 )
```

compute different types data's Divide. refer to [datatype.h](#)

Parameters

<i>data_type</i>	use this to get datatype.h 's function getTypeCode()
<i>data1</i>	operand1
<i>data2</i>	operand2

Return values

<i>true</i>	we have successfully write the result to data1 ,function successfully compute operand1 / operand2
<i>false</i>	write fail,can cause segment fault.

5.6.3.3 Sum()

```
bool Sum (
    BasicType * data_type,
    void * data1,
    void * data2 )
```

compute different types data's sum. refer to [datatype.h](#)

Parameters

<i>data_type</i>	use this to get datatype.h 's function getTypeCode()
<i>data1</i>	operand1
<i>data2</i>	operand2

Return values

<i>true</i>	we have successfully write the result to data1
<i>false</i>	write fail,can cause segment fault.

5.6.4 Variable Documentation

5.6.4.1 orderby_cmp

`Orderby*` `orderby_cmp`

use gloabl pointer.pointer to the compare function of qsort first want to place it in [Orderby Operator](#), but it stays at dump and cause segment fault,so choose to place it as global variable.

5.7 executor.h File Reference

```
#include "catalog.h"
#include "mymemory.h"
```

Include dependency graph for executor.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [RequestColumn](#)
- struct [RequestTable](#)
- struct [Condition](#)
- struct [Conditions](#)
- class [SelectQuery](#)
- class [ResultTable](#)
- class [Executor](#)
- class [Operator](#)
- class [Scan](#)
- class [Filter](#)
- class [Project](#)
- class [Join](#)
- struct [Groupby_struct](#)
- class [Groupby](#)
- class [Orderby](#)

Enumerations

- enum [AggrerateMethod](#) {
[NONE_AM](#) = 0, [COUNT](#), [SUM](#), [AVG](#),
[MAX](#), [MIN](#), [MAX_AM](#) }
- enum [CompareMethod](#) {
[NONE_CM](#) = 0, [LT](#), [LE](#), [EQ](#),
[NE](#), [GT](#), [GE](#), [LINK](#),
[MAX_CM](#) }

5.7.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.7.2 DESCRIPTION

definition of executor

5.7.3 Enumeration Type Documentation

5.7.3.1 AggrerateMethod

enum [AggrerateMethod](#)

aggrerate method.

Enumerator

NONE_AM	none
COUNT	count of rows
SUM	sum of data
AVG	average of data
MAX	maximum of data
MIN	minimum of data

5.7.3.2 CompareMethod

enum [CompareMethod](#)

compare method.

Enumerator

LT	less than
LE	less than or equal to
EQ	equal to
NE	not equal than

Enumerator

GT	greater than
GE	greater than or equal to
LINK	join

5.8 hashindex.cc File Reference

```
#include "hashindex.h"
```

Include dependency graph for hashindex.cc:

5.8.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.8.2 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.

5.9 hashindex.h File Reference

```
#include "schema.h"
```

```
#include "hashtable.h"
```

Include dependency graph for hashindex.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [HashInfo](#)
- class [HashIndex](#)

Macros

- `#define HASHINFO_CAPACITY (8)`

5.9.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.9.2 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 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 [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.

5.10 mymemory.cc File Reference

```
#include "mymemory.h"
```

Include dependency graph for mymemory.cc:

Variables

- [Memory](#) `g_memory`

5.10.1 Detailed Description

@author liugang(liugang@ict.ac.cn)

Version

0.1

5.10.2 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

5.11 mymemory.h File Reference

```
#include <stdint.h>
#include <stdio.h>
#include <vector>
#include <sys/mman.h>
```

Include dependency graph for mymemory.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Memory](#)

Macros

- #define **MEMORY_OK** 0
- #define **NON_TABLE_MEMORY_ADDR** ((void*)0x600000000000)
- #define **TABLE_MEMORY_INIT_ADDR** ((void*)0x700000000000)
- #define **TABLE_MEMORY_MAX_ADDR** ((void*)0x7f0000000000)
- #define **TABLE_MEMORY_ALLOC_MAX** (1L<<34)
- #define **TABLE_MEMORY_ALLOC_INC** (1L<<28)

Variables

- [Memory](#) g_memory

5.11.1 Detailed Description

@author liugang(liugang@ict.ac.cn)

Version

0.1

5.11.2 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

5.12 pbtree.cc File Reference

```
#include "pbtree.h"
```

Include dependency graph for pbtree.cc:

Macros

- `#define LEFT_KEY_NUM ((LEAF_KEY_NUM+1)/2)`
- `#define RIGHT_KEY_NUM ((LEAF_KEY_NUM+1) - LEFT_KEY_NUM)`
- `#define LEFT_KEY_NUM ((NON_LEAF_KEY_NUM)/2)`
- `#define RIGHT_KEY_NUM ((NON_LEAF_KEY_NUM) - LEFT_KEY_NUM)`

5.12.1 Detailed Description

Author

Shimin Chen shimin.chen@gmail.com

Version

1.0

5.12.2 LICENSE

TBD

5.12.3 DESCRIPTION

The pbtree class implements prefetching B+-tree (without jump pointer arrays). with NO_PREFETCH, this becomes an B+-tree implementation.

5.13 pbtree.h File Reference

```
#include <assert.h>
#include <stdlib.h>
#include "mymemory.h"
#include "nodepref.h"
#include <stdio.h>
```

Include dependency graph for pbtree.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Pointer8B](#)
- class [bnode](#)
- class [pbtree](#)
- class [Pbtree](#)

Macros

- `#define KEY_SIZE 8`
- `#define POINTER_SIZE 8`
- `#define POINTER8B_SIZE 8`
- `#define MAX_KEY ((key_type)(0x7fffffffffffffff))`
- `#define MIN_KEY ((key_type)(0x8000000000000000))`
- `#define BKEY_NUM (BNODE_SIZE/(KEY_SIZE+POINTER8B_SIZE) - 1)`
- `#define NON_LEAF_KEY_NUM BKEY_NUM`
- `#define LEAF_KEY_NUM BKEY_NUM`
- `#define bleaf bnode`
- `#define bnum(ptr) (((bleaf*)(ptr))->k(0))`
- `#define bnext(ptr) (((bleaf*)(ptr))->ch(0))`

Typedefs

- `typedef long long key_type`

5.13.1 Detailed Description

Author

Shimin Chen shimin.chen@gmail.com

Version

1.0

5.13.2 LICENSE

TBD

5.13.3 DESCRIPTION

The pbtree class implements prefetching B+-tree (without jump pointer arrays). with NO_PREFETCH, this becomes an B+-tree implementation.

5.14 pbtreeindex.h File Reference

```
#include "schema.h"
#include "pbtree.h"
```

Include dependency graph for pbtreeindex.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [PbtreeInfo](#)
- class [PbtreeIndex](#)

Macros

- `#define PBTREEINFO_CAPACITY (16)`

5.14.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.14.2 DESCRIPTION

pmtree 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

5.15 rowtable.cc File Reference

```
#include "rowtable.h"
```

Include dependency graph for rowtable.cc:

5.15.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.15.2 DESCRIPTION

rowtable implementation, this file implement all interface required by class table data space is managed by g_↔ memory, which decreases malloc overhead when create rowtable, I will make one more column to represent its validation. if delete a record, put down the label to set it "invalid" for index in this table, delete the entry inside index

5.16 rowtable.h File Reference

```
#include "mymemory.h"
#include "schema.h"
```

Include dependency graph for rowtable.h: This graph shows which files directly or indirectly include this file:

Classes

- class [RPattern](#)
- class [MStorage](#)
- class [RowTable](#)

Variables

- [Memory](#) `g_memory`

5.16.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.16.2 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

5.17 schema.h File Reference

```
#include <string.h>
#include <vector>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include "datatype.h"
```

Include dependency graph for schema.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Object](#)
- class [Column](#)
- class [Table](#)
- class [Database](#)
- class [Key](#)
- class [Index](#)

Macros

- `#define OBJ_NAME_MAX (128)`

Enumerations

- enum [ObjectType](#) {
`INVID_O = 0, DATABASE, TABLE, COLUMN,`
`INDEX, MAXTYPE_O }`
- enum [IndexType](#) {
`INVID_I = 0, HASHINDEX, BPTREEINDEX, ARTTREEINDEX,`
`MAXTYPE_I }`
- enum [TableType](#) { `INVID_T = 0, ROWTABLE, COLTABLE, MAXTYPE_T }`
- enum [ColumnType](#) {
`INVID_C = 0, INT8, INT16, INT32,`
`INT64, FLOAT32, FLOAT64, CHARN,`
`DATE, TIME, DATETIME, MAXTYPE_C }`

5.17.1 Detailed Description

Author

liugang(liugang@ict.ac.cn)

Version

0.1

5.17.2 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 ()

5.17.3 Enumeration Type Documentation

5.17.3.1 ColumnType

enum [ColumnType](#)

an enum for column.

Enumerator

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

5.17.3.2 IndexType

enum [IndexType](#)

an enum for [Index](#).

Enumerator

HASHINDEX	hash index
BPTREEINDEX	bptree index
ARTTREEINDEX	art tree index

5.17.3.3 ObjectType

enum [ObjectType](#)

an enum for ObjectType label.

Enumerator

DATABASE	database
TABLE	table
COLUMN	column
INDEX	index

5.17.3.4 TableType

enum `TableType`

an enum for `Table`.

Enumerator

ROWTABLE	row table
COLTABLE	column table

Index

- ~BasicType
 - BasicType, 8
- ~Column
 - Column, 17
- ~Database
 - Database, 21
- ~ErrorLog
 - ErrorLog, 25
- ~HashTable
 - HashTable, 45
- ~Index
 - Index, 51
- ~Table
 - Table, 113
- add
 - HashTable, 45
- addColumn
 - RPattern, 106
 - Table, 114
- addIndex
 - Table, 114
- addIndexDTpye
 - HashIndex, 37
- addTable
 - Database, 21
- AggrerateMethod
 - executor.h, 166
- alloc
 - Memory, 67
- allocRow
 - MStorage, 69
- allocTableAddr
 - Memory, 67
- area
 - PbtreenInfo, 85
- ARTTREEINDEX
 - schema.h, 175
- avail
 - HashTable, 48
- AVG
 - executor.h, 166
- b_type_code
 - BasicType, 10
- b_type_size
 - BasicType, 10
- BasicType, 7
 - ~BasicType, 8
 - b_type_code, 10
 - b_type_size, 10
 - BasicType, 7
 - cmpEQ, 8
 - cmpGE, 8
 - cmpGT, 8
 - cmpLE, 9
 - cmpLT, 9
 - copy, 9
 - formatBin, 9
 - formatTxt, 10
 - getTypeCode, 10
 - getTypeSize, 10
- begin
 - HashTable, 48
- bnode, 11
- BPTREEINDEX
 - schema.h, 175
- buffer
 - ResultTable, 93
- buffer_size
 - ResultTable, 93
- capacity
 - HashCell, 35
- Catalog, 11
 - createColumn, 12
 - createDatabase, 12
 - createIndex, 13
 - createTable, 13
 - getObjById, 14
 - getObjByName, 14
 - init, 14
 - initDatabase, 14
 - print, 15
 - shut, 15
 - shutDatabase, 15
- catalog.cc, 157
- catalog.h, 157
- changeName
 - Object, 71
- CHARN
 - schema.h, 175
- CHARN_TC
 - datatype.h, 159
- close
 - Executor, 28
- closeLog
 - ErrorLog, 25
- cmpEQ
 - BasicType, 8

- TypeCharN, [127](#)
- TypeDate, [130](#)
- TypeDateTime, [133](#)
- TypeFloat32, [136](#)
- TypeFloat64, [139](#)
- TypeInt16, [142](#)
- TypeInt32, [145](#)
- TypeInt64, [148](#)
- TypeInt8, [151](#)
- TypeTime, [154](#)
- cmpGE
 - BasicType, [8](#)
 - TypeCharN, [127](#)
 - TypeDate, [130](#)
 - TypeDateTime, [133](#)
 - TypeFloat32, [136](#)
 - TypeFloat64, [139](#)
 - TypeInt16, [142](#)
 - TypeInt32, [145](#)
 - TypeInt64, [148](#)
 - TypeInt8, [151](#)
 - TypeTime, [154](#)
- cmpGT
 - BasicType, [8](#)
 - TypeCharN, [127](#)
 - TypeDate, [130](#)
 - TypeDateTime, [133](#)
 - TypeFloat32, [136](#)
 - TypeFloat64, [139](#)
 - TypeInt16, [142](#)
 - TypeInt32, [145](#)
 - TypeInt64, [148](#)
 - TypeInt8, [151](#)
 - TypeTime, [154](#)
- cmpLE
 - BasicType, [9](#)
 - TypeCharN, [127](#)
 - TypeDate, [130](#)
 - TypeDateTime, [133](#)
 - TypeFloat32, [136](#)
 - TypeFloat64, [139](#)
 - TypeInt16, [142](#)
 - TypeInt32, [145](#)
 - TypeInt64, [148](#)
 - TypeInt8, [151](#)
 - TypeTime, [154](#)
- cmpLT
 - BasicType, [9](#)
 - TypeCharN, [128](#)
 - TypeDate, [131](#)
 - TypeDateTime, [134](#)
 - TypeFloat32, [137](#)
 - TypeFloat64, [140](#)
 - TypeInt16, [143](#)
 - TypeInt32, [146](#)
 - TypeInt64, [149](#)
 - TypeInt8, [152](#)
 - TypeTime, [155](#)
- COLTABLE
 - schema.h, [176](#)
- COLUMN
 - schema.h, [175](#)
- Column, [16](#)
 - ~Column, [17](#)
 - Column, [16](#)
 - finish, [17](#)
 - getCoffset, [17](#)
 - getCSize, [17](#)
 - getCType, [17](#)
 - getDataType, [17](#)
 - init, [18](#)
 - print, [18](#)
 - setCoffset, [18](#)
 - shut, [18](#)
- column
 - Condition, [19](#)
- Column_id_array
 - Operator, [74](#)
- Column_id_array_join
 - Join, [63](#)
- Column_id_array_prepare
 - Join, [63](#)
- column_number
 - ResultTable, [94](#)
- column_type
 - ResultTable, [94](#)
- ColumnType
 - schema.h, [174](#)
- compare
 - Condition, [19](#)
 - executor.cc, [163](#)
- CompareMethod
 - executor.h, [166](#)
- Condition, [18](#)
 - column, [19](#)
 - compare, [19](#)
 - value, [19](#)
- condition
 - Conditions, [20](#)
- condition_num
 - Conditions, [20](#)
- Conditions, [19](#)
 - condition, [20](#)
 - condition_num, [20](#)
- contain
 - Key, [65](#)
- copy
 - BasicType, [9](#)
 - TypeCharN, [128](#)
 - TypeDate, [131](#)
 - TypeDateTime, [134](#)
 - TypeFloat32, [137](#)
 - TypeFloat64, [140](#)
 - TypeInt16, [143](#)
 - TypeInt32, [146](#)
 - TypeInt64, [149](#)

- TypeInt8, [152](#)
- TypeTime, [155](#)
- COUNT
 - executor.h, [166](#)
- count
 - Orderby, [77](#)
- cr_area
 - PbtreeInfo, [85](#)
- cr_resu
 - PbtreeInfo, [85](#)
- createColumn
 - Catalog, [12](#)
- createDatabase
 - Catalog, [12](#)
- createIndex
 - Catalog, [13](#)
- createTable
 - Catalog, [13](#)
- current_buffer
 - Operator, [74](#)
- DATABASE
 - schema.h, [175](#)
- Database, [20](#)
 - ~Database, [21](#)
 - addTable, [21](#)
 - Database, [21](#)
 - finish, [22](#)
 - getTables, [22](#)
 - init, [22](#)
 - insert, [22](#)
 - loadData, [23](#)
 - print, [23](#)
 - shut, [23](#)
- database_id
 - SelectQuery, [111](#)
- datatype.h, [158](#)
 - CHARN_TC, [159](#)
 - DATE_TC, [159](#)
 - DATETIME_TC, [159](#)
 - FLOAT32_TC, [159](#)
 - FLOAT64_TC, [159](#)
 - INT16_TC, [159](#)
 - INT32_TC, [159](#)
 - INT64_TC, [159](#)
 - INT8_TC, [159](#)
 - TIME_TC, [159](#)
 - TypeCode, [159](#)
- DATE
 - schema.h, [175](#)
- DATE_TC
 - datatype.h, [159](#)
- DATETIME
 - schema.h, [175](#)
- DATETIME_TC
 - datatype.h, [159](#)
- del
 - HashIndex, [38](#)
 - HashTable, [46](#)
- Index, [51](#), [52](#)
- PbtreeIndex, [80](#)
- RowTable, [96](#)
- Table, [114](#), [115](#)
- Divide
 - executor.cc, [164](#)
- dump
 - ResultTable, [91](#)
- EL_LOG_INFO
 - errorlog.h, [162](#)
- EL_LOG_WARN
 - errorlog.h, [162](#)
- EL_src_file_name
 - errorlog.cc, [160](#)
- end
 - HashTable, [48](#)
- ent
 - HashCell, [35](#)
- ents
 - HashCell, [35](#)
- EQ
 - executor.h, [166](#)
- ErrorLog, [24](#)
 - ~ErrorLog, [25](#)
 - closeLog, [25](#)
 - ErrorLog, [25](#)
 - flushLog, [25](#)
 - getErrorCode, [25](#)
 - getErrorMsg, [26](#)
 - id2Name, [26](#)
 - init, [26](#)
 - log, [26](#)
 - name2Id, [27](#)
 - reset, [27](#)
 - setLevel, [27](#)
- errorlog.cc, [159](#)
 - EL_src_file_name, [160](#)
- errorlog.h, [160](#)
 - EL_LOG_INFO, [162](#)
 - EL_LOG_WARN, [162](#)
- estimated_duplicates_per_key
 - HashTable, [48](#)
- estimated_num_distinct_keys
 - HashTable, [48](#)
- exec
 - Executor, [28](#)
- Executor, [28](#)
 - close, [28](#)
 - exec, [28](#)
- executor.cc, [163](#)
 - compare, [163](#)
 - Divide, [164](#)
 - orderby_cmp, [165](#)
 - Sum, [164](#)
- executor.h, [165](#)
 - AggrerateMethod, [166](#)
 - AVG, [166](#)
 - CompareMethod, [166](#)

- COUNT, 166
- EQ, 166
- GE, 167
- GT, 167
- LE, 166
- LINK, 167
- LT, 166
- MAX, 166
- MIN, 166
- NE, 166
- NONE_AM, 166
- SUM, 166
- Filter, 29
 - filter_judge_condition, 31
 - filter_judge_num, 31
 - getNext, 30
 - init, 30
 - isEnd, 30
 - print, 31
- filter_judge_condition
 - Filter, 31
- filter_judge_num
 - Filter, 31
- finish
 - Column, 17
 - Database, 22
 - HashIndex, 39
 - Index, 53
 - RowTable, 96
 - Table, 115
- FLOAT32
 - schema.h, 175
- FLOAT32_TC
 - datatype.h, 159
- FLOAT64
 - schema.h, 175
- FLOAT64_TC
 - datatype.h, 159
- flushLog
 - ErrorLog, 25
- formatBin
 - BasicType, 9
 - TypeCharN, 128
 - TypeDate, 131
 - TypeDateTime, 134
 - TypeFloat32, 137
 - TypeFloat64, 140
 - TypeInt16, 143
 - TypeInt32, 146
 - TypeInt64, 149
 - TypeInt8, 152
 - TypeTime, 155
- formatTxt
 - BasicType, 10
 - TypeCharN, 128
 - TypeDate, 131
 - TypeDateTime, 134
 - TypeFloat32, 137
 - TypeFloat64, 140
 - TypeInt16, 143
 - TypeInt32, 146
 - TypeInt64, 149
 - TypeInt8, 152
 - TypeTime, 155
- free
 - Memory, 67
- free_header
 - HashTable, 48
- from_number
 - SelectQuery, 111
- from_table
 - SelectQuery, 111
- GE
 - executor.h, 167
- getCoffset
 - Column, 17
- getColumnOffset
 - RPattern, 106
- getColumnRank
 - Table, 116
- getColumns
 - Table, 116
- getColumnType
 - RPattern, 106
- getCSize
 - Column, 17
- getCType
 - Column, 17
- getDataType
 - Column, 17
- getErrorCode
 - ErrorLog, 25
- getErrMsg
 - ErrorLog, 26
- getIKey
 - Index, 53
- getIndexRank
 - Table, 116
- getIndexs
 - Table, 117
- getIndexTid
 - Index, 53
- getIType
 - Index, 53
- getKey
 - Key, 66
- getMStorage
 - RowTable, 97
- getNext
 - Filter, 30
 - Groupby, 32
 - Join, 62
 - Operator, 73
 - Orderby, 76
 - Project, 88
 - Scan, 109

- getObjById
 - Catalog, 14
- getObjByName
 - Catalog, 14
- getOid
 - Object, 72
- getOname
 - Object, 72
- getOtype
 - Object, 72
- getRank
 - Table, 117
- getRC
 - ResultTable, 92
- getRecordNum
 - MStorage, 69
 - RowTable, 97
 - Table, 117
- getRecordPtr
 - RowTable, 97
 - Table, 117
- getRow
 - MStorage, 69
- getRowSize
 - RPattern, 107
- getRPattern
 - RowTable, 97
- getTables
 - Database, 22
- getType
 - Table, 118
- getTypeCode
 - BasicType, 10
- getTypeSize
 - BasicType, 10
- given_condition
 - Groupby_struct, 34
- Groupby, 31
 - getNext, 32
 - init, 32
 - isEnd, 33
 - print, 33
- groupby
 - SelectQuery, 111
- groupby_number
 - SelectQuery, 111
- Groupby_struct, 33
 - given_condition, 34
 - value, 34
- GT
 - executor.h, 167
- hash
 - HashInfo, 43
- hash_code
 - Hashcode_Ptr, 36
- HashCell, 34
 - capacity, 35
 - ent, 35
 - ents, 35
 - hc_num, 35
 - hc_union, 35
 - num_2_or_more, 35
- Hashcode_Ptr, 36
 - hash_code, 36
 - tuple, 36
- HASHINDEX
 - schema.h, 175
- HashIndex, 36
 - addIndexDType, 37
 - del, 38
 - finish, 39
 - HashIndex, 37
 - init, 39
 - insert, 39, 40
 - lookup, 40, 41
 - print, 41
 - set_Is, 41, 42
 - setCellCap, 42
 - shut, 43
- hashindex.cc, 167
- hashindex.h, 167
- HashInfo, 43
 - hash, 43
 - last, 43
 - ppos, 44
 - result, 44
 - rnum, 44
- HashTable, 44
 - ~HashTable, 45
 - add, 45
 - avail, 48
 - begin, 48
 - del, 46
 - end, 48
 - estimated_duplicates_per_key, 48
 - estimated_num_distinct_keys, 48
 - free_header, 48
 - HashTable, 45
 - initial_array_size, 48
 - more_allocated, 48
 - probe, 46
 - probe_contd, 47
 - show, 47
 - table, 49
 - table_size, 49
 - utilization, 47
- having
 - SelectQuery, 111
- hc_num
 - HashCell, 35
- hc_union
 - HashCell, 35
- hx
 - Join, 64
- i_key
 - Index, 61

- i_t_id
 - Index, [61](#)
- i_type
 - Index, [61](#)
- id2Name
 - ErrorLog, [26](#)
- INDEX
 - schema.h, [175](#)
- Index, [49](#)
 - ~Index, [51](#)
 - del, [51](#), [52](#)
 - finish, [53](#)
 - getlKey, [53](#)
 - getIndexTid, [53](#)
 - getType, [53](#)
 - i_key, [61](#)
 - i_t_id, [61](#)
 - i_type, [61](#)
 - Index, [50](#)
 - init, [53](#)
 - insert, [53](#), [54](#)
 - lookup, [54](#)–[56](#)
 - print, [56](#)
 - scan, [56](#)
 - scan_1, [57](#)
 - scan_2, [57](#), [58](#)
 - set_ls, [58](#), [59](#)
 - setIndexTid, [59](#)
 - shut, [59](#)
 - tranToInt64, [60](#)
- IndexType
 - schema.h, [175](#)
- init
 - Catalog, [14](#)
 - Column, [18](#)
 - Database, [22](#)
 - ErrorLog, [26](#)
 - Filter, [30](#)
 - Groupby, [32](#)
 - HashIndex, [39](#)
 - Index, [53](#)
 - Join, [62](#)
 - Memory, [68](#)
 - MStorage, [70](#)
 - Operator, [73](#)
 - Orderby, [76](#)
 - PbtreeIndex, [80](#)
 - Project, [88](#)
 - ResultTable, [92](#)
 - RowTable, [97](#)
 - RPattern, [107](#)
 - Scan, [109](#)
 - Table, [118](#)
- initDatabase
 - Catalog, [14](#)
- initial_array_size
 - HashTable, [48](#)
- insert
 - Database, [22](#)
 - HashIndex, [39](#), [40](#)
 - Index, [53](#), [54](#)
 - PbtreeIndex, [81](#)
 - RowTable, [98](#)
 - Table, [118](#)
- insert_hash_data
 - Join, [64](#)
- INT16
 - schema.h, [175](#)
- INT16_TC
 - datatype.h, [159](#)
- INT32
 - schema.h, [175](#)
- INT32_TC
 - datatype.h, [159](#)
- INT64
 - schema.h, [175](#)
- INT64_TC
 - datatype.h, [159](#)
- INT8
 - schema.h, [175](#)
- INT8_TC
 - datatype.h, [159](#)
- isEnd
 - Filter, [30](#)
 - Groupby, [33](#)
 - Join, [63](#)
 - Operator, [74](#)
 - Orderby, [76](#)
 - Project, [89](#)
 - Scan, [109](#)
- Join, [61](#)
 - Column_id_array_join, [63](#)
 - Column_id_array_prepare, [63](#)
 - getNext, [62](#)
 - hx, [64](#)
 - init, [62](#)
 - insert_hash_data, [64](#)
 - isEnd, [63](#)
 - join_given_condition_num, [64](#)
 - join_lchild_rank, [64](#)
 - join_rchild_rank, [64](#)
 - lookup_hash_data, [64](#)
 - print, [63](#)
- join_given_condition_num
 - Join, [64](#)
- join_lchild_rank
 - Join, [64](#)
- join_rchild_rank
 - Join, [64](#)
- Key, [65](#)
 - contain, [65](#)
 - getKey, [66](#)
 - Key, [65](#)
 - operator=, [66](#)
 - print, [66](#)

- set, 66
- l_ptr
 - PbtreeInfo, 86
- last
 - HashInfo, 43
- lchild
 - Operator, 74
- LE
 - executor.h, 166
- le_resu
 - PbtreeInfo, 86
- left
 - PbtreeInfo, 86
- LINK
 - executor.h, 167
- loadData
 - Database, 23
 - RowTable, 98
 - Table, 119
- log
 - ErrorLog, 26
- lookup
 - HashIndex, 40, 41
 - Index, 54–56
 - PbtreeIndex, 81
- lookup_hash_data
 - Join, 64
- LT
 - executor.h, 166
- MAX
 - executor.h, 166
- Memory, 66
 - alloc, 67
 - allocTableAddr, 67
 - free, 67
 - init, 68
 - print, 68
 - shut, 68
- MIN
 - executor.h, 166
- more_allocated
 - HashTable, 48
- MStorage, 69
 - allocRow, 69
 - getRecordNum, 69
 - getRow, 69
 - init, 70
 - shut, 70
- mymemory.cc, 168
- mymemory.h, 169
- name
 - RequestColumn, 90
- name2Id
 - ErrorLog, 27
- NE
 - executor.h, 166
- NONE_AM
 - executor.h, 166
- num_2_or_more
 - HashCell, 35
- Object, 71
 - changeName, 71
 - getOid, 72
 - getOname, 72
 - getOtype, 72
 - Object, 71
 - print, 72
 - shut, 72
- ObjectType
 - schema.h, 175
- Operator, 73
 - Column_id_array, 74
 - current_buffer, 74
 - getNext, 73
 - init, 73
 - isEnd, 74
 - lchild, 74
 - parent, 75
 - prev_buffer, 75
 - print, 74
 - rchild, 75
 - row_column_RPattern, 75
- operator=
 - Key, 66
- Orderby, 75
 - count, 77
 - getNext, 76
 - init, 76
 - isEnd, 76
 - orderby, 77
 - orderby_data_type, 77
 - orderby_number, 78
 - orderby_offset, 78
 - orderby_vector, 78
 - print, 77
- orderby
 - Orderby, 77
 - SelectQuery, 111
- orderby_cmp
 - executor.cc, 165
- orderby_data_type
 - Orderby, 77
- orderby_number
 - Orderby, 78
 - SelectQuery, 112
- orderby_offset
 - Orderby, 78
- orderby_vector
 - Orderby, 78
- parent
 - Operator, 75
- Pbtree, 78
- pbtrees, 79

- pbtree.cc, [170](#)
- pbtree.h, [170](#)
- PbtreeIndex, [79](#)
 - del, [80](#)
 - init, [80](#)
 - insert, [81](#)
 - lookup, [81](#)
 - PbtreeIndex, [80](#)
 - print, [82](#)
 - scan, [82](#)
 - set_ls, [83](#)
 - setIndexDTpye, [84](#)
 - shut, [84](#)
- pbtreeindex.h, [171](#)
- PbtreeInfo, [85](#)
 - area, [85](#)
 - cr_area, [85](#)
 - cr_resu, [85](#)
 - l_ptr, [86](#)
 - le_resu, [86](#)
 - left, [86](#)
 - pos_resu, [86](#)
 - result, [86](#)
 - right, [86](#)
 - s_end, [86](#)
 - s_num, [86](#)
 - s_pos, [87](#)
 - s_ptr, [87](#)
- Pointer8B, [87](#)
- pos_resu
 - PbtreeInfo, [86](#)
- ppos
 - HashInfo, [44](#)
- prev_buffer
 - Operator, [75](#)
- print
 - Catalog, [15](#)
 - Column, [18](#)
 - Database, [23](#)
 - Filter, [31](#)
 - Groupby, [33](#)
 - HashIndex, [41](#)
 - Index, [56](#)
 - Join, [63](#)
 - Key, [66](#)
 - Memory, [68](#)
 - Object, [72](#)
 - Operator, [74](#)
 - Orderby, [77](#)
 - PbtreeIndex, [82](#)
 - Project, [89](#)
 - ResultTable, [92](#)
 - RPattern, [107](#)
 - Scan, [110](#)
 - Table, [119](#)
- printData
 - RowTable, [99](#)
 - Table, [119](#)
- probe
 - HashTable, [46](#)
- probe_contd
 - HashTable, [47](#)
- Project, [88](#)
 - getNext, [88](#)
 - init, [88](#)
 - isEnd, [89](#)
 - print, [89](#)
 - project_column_id, [89](#)
- project_column_id
 - Project, [89](#)
- rchild
 - Operator, [75](#)
- RequestColumn, [90](#)
 - name, [90](#)
- RequestTable, [90](#)
- reset
 - ErrorLog, [27](#)
 - RPattern, [108](#)
- result
 - HashInfo, [44](#)
 - PbtreeInfo, [86](#)
- ResultTable, [91](#)
 - buffer, [93](#)
 - buffer_size, [93](#)
 - column_number, [94](#)
 - column_type, [94](#)
 - dump, [91](#)
 - getRC, [92](#)
 - init, [92](#)
 - print, [92](#)
 - row_capacity, [94](#)
 - row_length, [94](#)
 - row_number, [94](#)
 - shut, [93](#)
 - writeRC, [93](#)
- right
 - PbtreeInfo, [86](#)
- rnum
 - HashInfo, [44](#)
- row_capacity
 - ResultTable, [94](#)
- row_column_RPattern
 - Operator, [75](#)
- row_length
 - ResultTable, [94](#)
- row_number
 - ResultTable, [94](#)
- ROWTABLE
 - schema.h, [176](#)
- RowTable, [94](#)
 - del, [96](#)
 - finish, [96](#)
 - getMStorage, [97](#)
 - getRecordNum, [97](#)
 - getRecordPtr, [97](#)
 - getRPattern, [97](#)

- init, 97
- insert, 98
- loadData, 98
- printData, 99
- RowTable, 95
- select, 99
- selectCol, 100
- selectCols, 101
- shut, 102
- updateCol, 102, 103
- updateCols, 103–105
- rowtable.cc, 172
- rowtable.h, 173
- RPattern, 105
 - addColumn, 106
 - getColumnOffset, 106
 - getColumnType, 106
 - getRowSize, 107
 - init, 107
 - print, 107
 - reset, 108
 - shut, 108
- s_end
 - PbtreeNode, 86
- s_num
 - PbtreeNode, 86
- s_pos
 - PbtreeNode, 87
- s_ptr
 - PbtreeNode, 87
- Scan, 108
 - getNext, 109
 - init, 109
 - isEnd, 109
 - print, 110
- scan
 - Index, 56
 - PbtreeNode, 82
- scan_1
 - Index, 57
- scan_2
 - Index, 57, 58
- schema.h, 173
 - ARTTREEINDEX, 175
 - BPTREEINDEX, 175
 - CHARN, 175
 - COLTABLE, 176
 - COLUMN, 175
 - ColumnType, 174
 - DATABASE, 175
 - DATE, 175
 - DATETIME, 175
 - FLOAT32, 175
 - FLOAT64, 175
 - HASHINDEX, 175
 - INDEX, 175
 - IndexType, 175
 - INT16, 175
 - INT32, 175
 - INT64, 175
 - INT8, 175
 - ObjectType, 175
 - ROWTABLE, 176
 - TABLE, 175
 - TableType, 175
 - TIME, 175
- select
 - RowTable, 99
 - Table, 119, 120
- select_column
 - SelectQuery, 112
- select_number
 - SelectQuery, 112
- selectCol
 - RowTable, 100
 - Table, 120, 121
- selectCols
 - RowTable, 101
 - Table, 121, 122
- SelectQuery, 110
 - database_id, 111
 - from_number, 111
 - from_table, 111
 - groupby, 111
 - groupby_number, 111
 - having, 111
 - orderby, 111
 - orderby_number, 112
 - select_column, 112
 - select_number, 112
 - where, 112
- set
 - Key, 66
- set_ls
 - HashIndex, 41, 42
 - Index, 58, 59
 - PbtreeNode, 83
- setCellCap
 - HashIndex, 42
- setCoffset
 - Column, 18
- setIndexDTpye
 - PbtreeNode, 84
- setIndexTid
 - Index, 59
- setLevel
 - ErrorLog, 27
- show
 - HashTable, 47
- shut
 - Catalog, 15
 - Column, 18
 - Database, 23
 - HashIndex, 43
 - Index, 59
 - Memory, 68

- MStorage, [70](#)
- Object, [72](#)
- PbtreeIndex, [84](#)
- ResultTable, [93](#)
- RowTable, [102](#)
- RPattern, [108](#)
- Table, [122](#)
- shutDatabase
 - Catalog, [15](#)
- SUM
 - executor.h, [166](#)
- Sum
 - executor.cc, [164](#)
- TABLE
 - schema.h, [175](#)
- Table, [112](#)
 - ~Table, [113](#)
 - addColumn, [114](#)
 - addIndex, [114](#)
 - del, [114](#), [115](#)
 - finish, [115](#)
 - getColumnRank, [116](#)
 - getColumns, [116](#)
 - getIndexRank, [116](#)
 - getIndexes, [117](#)
 - getRank, [117](#)
 - getRecordNum, [117](#)
 - getRecordPtr, [117](#)
 - getType, [118](#)
 - init, [118](#)
 - insert, [118](#)
 - loadData, [119](#)
 - print, [119](#)
 - printData, [119](#)
 - select, [119](#), [120](#)
 - selectCol, [120](#), [121](#)
 - selectCols, [121](#), [122](#)
 - shut, [122](#)
 - Table, [113](#)
 - updateCol, [122](#), [124](#)
 - updateCols, [124](#), [125](#)
- table
 - HashTable, [49](#)
- table_size
 - HashTable, [49](#)
- TableType
 - schema.h, [175](#)
- TIME
 - schema.h, [175](#)
- TIME_TC
 - datatype.h, [159](#)
- tranToInt64
 - Index, [60](#)
- tuple
 - Hashcode_Ptr, [36](#)
- TypeCharN, [126](#)
 - cmpEQ, [127](#)
 - cmpGE, [127](#)
 - cmpGT, [127](#)
 - cmpLE, [127](#)
 - cmpLT, [128](#)
 - copy, [128](#)
 - formatBin, [128](#)
 - formatTxt, [128](#)
 - TypeCharN, [127](#)
- TypeCode
 - datatype.h, [159](#)
- TypeDate, [129](#)
 - cmpEQ, [130](#)
 - cmpGE, [130](#)
 - cmpGT, [130](#)
 - cmpLE, [130](#)
 - cmpLT, [131](#)
 - copy, [131](#)
 - formatBin, [131](#)
 - formatTxt, [131](#)
 - TypeDate, [129](#)
- TypeDateTime, [132](#)
 - cmpEQ, [133](#)
 - cmpGE, [133](#)
 - cmpGT, [133](#)
 - cmpLE, [133](#)
 - cmpLT, [134](#)
 - copy, [134](#)
 - formatBin, [134](#)
 - formatTxt, [134](#)
 - TypeDateTime, [132](#)
- TypeFloat32, [135](#)
 - cmpEQ, [136](#)
 - cmpGE, [136](#)
 - cmpGT, [136](#)
 - cmpLE, [136](#)
 - cmpLT, [137](#)
 - copy, [137](#)
 - formatBin, [137](#)
 - formatTxt, [137](#)
 - TypeFloat32, [135](#)
- TypeFloat64, [138](#)
 - cmpEQ, [139](#)
 - cmpGE, [139](#)
 - cmpGT, [139](#)
 - cmpLE, [139](#)
 - cmpLT, [140](#)
 - copy, [140](#)
 - formatBin, [140](#)
 - formatTxt, [140](#)
 - TypeFloat64, [138](#)
- TypeInt16, [141](#)
 - cmpEQ, [142](#)
 - cmpGE, [142](#)
 - cmpGT, [142](#)
 - cmpLE, [142](#)
 - cmpLT, [143](#)
 - copy, [143](#)
 - formatBin, [143](#)
 - formatTxt, [143](#)

- TypeInt16, [141](#)
- TypeInt32, [144](#)
 - cmpEQ, [145](#)
 - cmpGE, [145](#)
 - cmpGT, [145](#)
 - cmpLE, [145](#)
 - cmpLT, [146](#)
 - copy, [146](#)
 - formatBin, [146](#)
 - formatTxt, [146](#)
 - TypeInt32, [144](#)
- TypeInt64, [147](#)
 - cmpEQ, [148](#)
 - cmpGE, [148](#)
 - cmpGT, [148](#)
 - cmpLE, [148](#)
 - cmpLT, [149](#)
 - copy, [149](#)
 - formatBin, [149](#)
 - formatTxt, [149](#)
 - TypeInt64, [147](#)
- TypeInt8, [150](#)
 - cmpEQ, [151](#)
 - cmpGE, [151](#)
 - cmpGT, [151](#)
 - cmpLE, [151](#)
 - cmpLT, [152](#)
 - copy, [152](#)
 - formatBin, [152](#)
 - formatTxt, [152](#)
 - TypeInt8, [150](#)
- TypeTime, [153](#)
 - cmpEQ, [154](#)
 - cmpGE, [154](#)
 - cmpGT, [154](#)
 - cmpLE, [154](#)
 - cmpLT, [155](#)
 - copy, [155](#)
 - formatBin, [155](#)
 - formatTxt, [155](#)
 - TypeTime, [153](#)
- updateCol
 - RowTable, [102](#), [103](#)
 - Table, [122](#), [124](#)
- updateCols
 - RowTable, [103–105](#)
 - Table, [124](#), [125](#)
- utilization
 - HashTable, [47](#)
- value
 - Condition, [19](#)
 - Groupby_struct, [34](#)
- where
 - SelectQuery, [112](#)
- writeRC
 - ResultTable, [93](#)