

# HailDB

2.3.2

Generated by Doxygen 1.7.1

Fri Dec 24 2010 14:35:34



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Data Structures . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	Startup/Shutdown functions . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Function Documentation . . . . .	7
4.1.2.1	ib_init . . . . .	7
4.1.2.2	ib_shutdown . . . . .	7
4.1.2.3	ib_startup . . . . .	8
4.2	Cursor functions . . . . .	8
4.2.1	Function Documentation . . . . .	8
4.2.1.1	ib_cursor_attach_trx . . . . .	8
4.2.1.2	ib_cursor_close . . . . .	9
4.2.1.3	ib_cursor_first . . . . .	9
4.2.1.4	ib_cursor_is_positioned . . . . .	9
4.2.1.5	ib_cursor_last . . . . .	9
4.2.1.6	ib_cursor_moveto . . . . .	10
4.2.1.7	ib_cursor_next . . . . .	10
4.2.1.8	ib_cursor_open_index_using_id . . . . .	10
4.2.1.9	ib_cursor_open_index_using_name . . . . .	10
4.2.1.10	ib_cursor_open_table . . . . .	11
4.2.1.11	ib_cursor_open_table_using_id . . . . .	11

4.2.1.12	<code>ib_cursor_prev</code>	11
4.2.1.13	<code>ib_cursor_reset</code>	12
4.2.1.14	<code>ib_cursor_set_match_mode</code>	12
4.2.1.15	<code>ib_cursor_stmt_begin</code>	12
4.3	Transaction functions	12
4.3.1	Function Documentation	13
4.3.1.1	<code>ib_cursor_lock</code>	13
4.3.1.2	<code>ib_cursor_set_lock_mode</code>	13
4.3.1.3	<code>ib_get_duplicate_key</code>	13
4.3.1.4	<code>ib_savepoint_release</code>	14
4.3.1.5	<code>ib_savepoint_rollback</code>	14
4.3.1.6	<code>ib_savepoint_take</code>	14
4.3.1.7	<code>ib_table_lock</code>	15
4.3.1.8	<code>ib_trx_begin</code>	15
4.3.1.9	<code>ib_trx_commit</code>	15
4.3.1.10	<code>ib_trx_release</code>	15
4.3.1.11	<code>ib_trx_rollback</code>	16
4.3.1.12	<code>ib_trx_set_client_data</code>	16
4.3.1.13	<code>ib_trx_start</code>	16
4.3.1.14	<code>ib_trx_state</code>	16
4.4	Configuration functions	17
4.4.1	Define Documentation	17
4.4.1.1	<code>ib_cfg_set_bool_off</code>	17
4.4.1.2	<code>ib_cfg_set_bool_on</code>	17
4.4.1.3	<code>ib_cfg_set_callback</code>	18
4.4.1.4	<code>ib_cfg_set_int</code>	18
4.4.1.5	<code>ib_cfg_set_text</code>	18
4.4.2	Function Documentation	19
4.4.2.1	<code>ib_cfg_get</code>	19
4.4.2.2	<code>ib_cfg_get_all</code>	19
4.4.2.3	<code>ib_cfg_set</code>	19
4.4.2.4	<code>ib_cfg_var_get_type</code>	19
4.4.2.5	<code>ib_set_panic_handler</code>	20
4.4.2.6	<code>ib_set_trx_is_interrupted_handler</code>	20
4.4.2.7	<code>ib_status_get_all</code>	20
4.5	DDL functions	20

---

4.5.1	Function Documentation	21
4.5.1.1	ib_cursor_truncate	21
4.5.1.2	ib_database_create	22
4.5.1.3	ib_database_drop	22
4.5.1.4	ib_index_create	22
4.5.1.5	ib_index_drop	22
4.5.1.6	ib_index_get_id	23
4.5.1.7	ib_index_schema_add_col	23
4.5.1.8	ib_index_schema_create	23
4.5.1.9	ib_index_schema_delete	23
4.5.1.10	ib_index_schema_set_clustered	24
4.5.1.11	ib_index_schema_set_unique	24
4.5.1.12	ib_schema_lock_exclusive	24
4.5.1.13	ib_schema_lock_is_exclusive	24
4.5.1.14	ib_schema_lock_shared	25
4.5.1.15	ib_schema_tables_iterate	25
4.5.1.16	ib_schema_unlock	25
4.5.1.17	ib_table_create	25
4.5.1.18	ib_table_drop	26
4.5.1.19	ib_table_get_id	26
4.5.1.20	ib_table_rename	26
4.5.1.21	ib_table_schema_add_col	27
4.5.1.22	ib_table_schema_add_index	27
4.5.1.23	ib_table_schema_create	27
4.5.1.24	ib_table_schema_delete	28
4.5.1.25	ib_table_schema_visit	28
4.5.1.26	ib_table_truncate	28
4.6	Miscellaneous functions	28
4.6.1	Function Documentation	29
4.6.1.1	ib_api_version	29
4.6.1.2	ib_get_index_stat_n_diff_key_vals	29
4.6.1.3	ib_get_table_statistics	30
4.6.1.4	ib_logger_set	30
4.6.1.5	ib_set_client_compare	30
4.6.1.6	ib_status_get_i64	30
4.6.1.7	ib_strerror	31

4.6.1.8	<code>ib_update_table_statistics</code>	31
4.7	Tuple functions	31
4.7.1	Function Documentation	31
4.7.1.1	<code>ib_clust_read_tuple_create</code>	31
4.7.1.2	<code>ib_clust_search_tuple_create</code>	32
4.7.1.3	<code>ib_sec_read_tuple_create</code>	32
4.7.1.4	<code>ib_sec_search_tuple_create</code>	32
4.7.1.5	<code>ib_tuple_clear</code>	32
4.7.1.6	<code>ib_tuple_copy</code>	33
4.7.1.7	<code>ib_tuple_delete</code>	33
4.7.1.8	<code>ib_tuple_get_cluster_key</code>	33
4.7.1.9	<code>ib_tuple_get_n_cols</code>	33
4.7.1.10	<code>ib_tuple_get_n_user_cols</code>	34
4.8	SQL functions	34
4.8.1	Function Documentation	34
4.8.1.1	<code>ib_cursor_set_simple_select</code>	34
4.9	DML functions	34
4.9.1	Function Documentation	35
4.9.1.1	<code>ib_col_copy_value</code>	35
4.9.1.2	<code>ib_col_get_len</code>	36
4.9.1.3	<code>ib_col_get_meta</code>	36
4.9.1.4	<code>ib_col_get_value</code>	36
4.9.1.5	<code>ib_col_set_value</code>	36
4.9.1.6	<code>ib_cursor_delete_row</code>	37
4.9.1.7	<code>ib_cursor_insert_row</code>	37
4.9.1.8	<code>ib_cursor_read_row</code>	37
4.9.1.9	<code>ib_cursor_set_cluster_access</code>	37
4.9.1.10	<code>ib_cursor_update_row</code>	38
4.9.1.11	<code>ib_tuple_read_double</code>	38
4.9.1.12	<code>ib_tuple_read_float</code>	38
4.9.1.13	<code>ib_tuple_read_i16</code>	38
4.9.1.14	<code>ib_tuple_read_i32</code>	39
4.9.1.15	<code>ib_tuple_read_i64</code>	39
4.9.1.16	<code>ib_tuple_read_i8</code>	39
4.9.1.17	<code>ib_tuple_read_u16</code>	40
4.9.1.18	<code>ib_tuple_read_u32</code>	40

4.9.1.19	<code>ib_tuple_read_u64</code>	40
4.9.1.20	<code>ib_tuple_read_u8</code>	40
4.9.1.21	<code>ib_tuple_write_double</code>	41
4.9.1.22	<code>ib_tuple_write_float</code>	41
4.9.1.23	<code>ib_tuple_write_i16</code>	41
4.9.1.24	<code>ib_tuple_write_i32</code>	42
4.9.1.25	<code>ib_tuple_write_i64</code>	42
4.9.1.26	<code>ib_tuple_write_i8</code>	42
4.9.1.27	<code>ib_tuple_write_u16</code>	42
4.9.1.28	<code>ib_tuple_write_u32</code>	43
4.9.1.29	<code>ib_tuple_write_u64</code>	43
4.9.1.30	<code>ib_tuple_write_u8</code>	43
4.10	Debug and Testing functions	44
4.10.1	Function Documentation	44
4.10.1.1	<code>ib_error_inject</code>	44
<b>5</b>	<b>Data Structure Documentation</b>	<b>45</b>
5.1	<code>ib_col_meta_t</code> Struct Reference	45
5.1.1	Detailed Description	45
5.1.2	Field Documentation	45
5.1.2.1	<code>attr</code>	45
5.1.2.2	<code>charset</code>	45
5.1.2.3	<code>client_type</code>	45
5.1.2.4	<code>type</code>	45
5.1.2.5	<code>type_len</code>	46
5.2	<code>ib_schema_visitor_t</code> Struct Reference	46
5.2.1	Detailed Description	46
5.2.2	Field Documentation	46
5.2.2.1	<code>index</code>	46
5.2.2.2	<code>index_col</code>	46
5.2.2.3	<code>table</code>	46
5.2.2.4	<code>table_col</code>	46
5.2.2.5	<code>version</code>	46
5.3	<code>ib_table_stats_t</code> Struct Reference	47
5.3.1	Detailed Description	47
5.3.2	Field Documentation	47
5.3.2.1	<code>stat_clustered_index_size</code>	47

5.3.2.2	stat_modified_counter . . . . .	47
5.3.2.3	stat_n_rows . . . . .	47
5.3.2.4	stat_sum_of_other_index_sizes . . . . .	47
<b>6</b>	<b>File Documentation</b>	<b>49</b>
6.1	haildb.h File Reference . . . . .	49
6.1.1	Detailed Description . . . . .	55
6.1.2	Define Documentation . . . . .	55
6.1.2.1	IB_FALSE . . . . .	55
6.1.2.2	IB_MAX_COL_NAME_LEN . . . . .	55
6.1.2.3	IB_MAX_TABLE_NAME_LEN . . . . .	56
6.1.2.4	IB_N_SYS_COLS . . . . .	56
6.1.2.5	IB_SQL_NULL . . . . .	56
6.1.2.6	ib_tbl_sch_add_blob_col . . . . .	56
6.1.2.7	ib_tbl_sch_add_text_col . . . . .	56
6.1.2.8	ib_tbl_sch_add_u32_col . . . . .	56
6.1.2.9	ib_tbl_sch_add_u64_col . . . . .	57
6.1.2.10	ib_tbl_sch_add_u64_notnull_col . . . . .	57
6.1.2.11	ib_tbl_sch_add_varchar_col . . . . .	57
6.1.2.12	IB_TRUE . . . . .	57
6.1.2.13	MAX_TEXT_LEN . . . . .	58
6.1.2.14	UNIV_NO_IGNORE . . . . .	58
6.1.3	Typedef Documentation . . . . .	58
6.1.3.1	ib_bool_t . . . . .	58
6.1.3.2	ib_byte_t . . . . .	58
6.1.3.3	ib_cb_t . . . . .	58
6.1.3.4	ib_charset_t . . . . .	58
6.1.3.5	ib_client_cmp_t . . . . .	58
6.1.3.6	ib_csr_t . . . . .	58
6.1.3.7	ib_err_t . . . . .	58
6.1.3.8	ib_i16_t . . . . .	58
6.1.3.9	ib_i32_t . . . . .	59
6.1.3.10	ib_i64_t . . . . .	59
6.1.3.11	ib_i8_t . . . . .	59
6.1.3.12	ib_id_t . . . . .	59
6.1.3.13	ib_idx_sch_t . . . . .	59
6.1.3.14	ib_msg_log_t . . . . .	59



---

6.1.3.15	<code>ib_msg_stream_t</code>	59
6.1.3.16	<code>ib_opaque_t</code>	59
6.1.3.17	<code>ib_panic_handler_t</code>	59
6.1.3.18	<code>ib_schema_visitor_index_col_t</code>	59
6.1.3.19	<code>ib_schema_visitor_index_t</code>	60
6.1.3.20	<code>ib_schema_visitor_table_all_t</code>	60
6.1.3.21	<code>ib_schema_visitor_table_col_t</code>	60
6.1.3.22	<code>ib_schema_visitor_table_t</code>	60
6.1.3.23	<code>ib_tbl_sch_t</code>	60
6.1.3.24	<code>ib_tpl_t</code>	60
6.1.3.25	<code>ib_trx_is_interrupted_handler_t</code>	60
6.1.3.26	<code>ib_trx_t</code>	60
6.1.3.27	<code>ib_u16_t</code>	60
6.1.3.28	<code>ib_u32_t</code>	61
6.1.3.29	<code>ib_u64_t</code>	61
6.1.3.30	<code>ib_u8_t</code>	61
6.1.3.31	<code>ib_ulint_t</code>	61
6.1.4	Enumeration Type Documentation	61
6.1.4.1	<code>db_err</code>	61
6.1.4.2	<code>ib_cfg_type_t</code>	63
6.1.4.3	<code>ib_col_attr_t</code>	63
6.1.4.4	<code>ib_col_type_t</code>	63
6.1.4.5	<code>ib_lck_mode_t</code>	64
6.1.4.6	<code>ib_match_mode_t</code>	64
6.1.4.7	<code>ib_schema_visitor_version_t</code>	64
6.1.4.8	<code>ib_shutdown_t</code>	64
6.1.4.9	<code>ib_srch_mode_t</code>	64
6.1.4.10	<code>ib_tbl_fmt_t</code>	65
6.1.4.11	<code>ib_trx_level_t</code>	65
6.1.4.12	<code>ib_trx_state_t</code>	65
6.1.5	Function Documentation	66
6.1.5.1	<code>ib_schema_lock_is_shared</code>	66
6.1.6	Variable Documentation	66
6.1.6.1	<code>ib_client_compare</code>	66



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

Startup/Shutdown functions . . . . .	7
Cursor functions . . . . .	8
Transaction functions . . . . .	12
Configuration functions . . . . .	17
DDL functions . . . . .	20
Miscellaneous functions . . . . .	28
Tuple functions . . . . .	31
SQL functions . . . . .	34
DML functions . . . . .	34
Debug and Testing functions . . . . .	44



# Chapter 2

## Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">ib_col_meta_t</a> . . . . .	45
<a href="#">ib_schema_visitor_t</a> . . . . .	46
<a href="#">ib_table_stats_t</a> . . . . .	47



# Chapter 3

## File Index

### 3.1 File List

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

<a href="#">haildb.h</a> . . . . .	49
------------------------------------	----





# Chapter 4

## Module Documentation

### 4.1 Startup/Shutdown functions

#### Functions

- HAILDB\_API [ib\\_err\\_t ib\\_init](#) (void) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_err\\_t ib\\_startup](#) (const char \*format) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_err\\_t ib\\_shutdown](#) ([ib\\_shutdown\\_t](#) flag) UNIV\_NO\_IGNORE

#### 4.1.1 Detailed Description

Define the Doxygen groups:

#### 4.1.2 Function Documentation

##### 4.1.2.1 HAILDB\_API [ib\\_err\\_t ib\\_init](#) ( void )

Initialize the InnoDB engine. This must be called prior to calling any other InnoDB API function. You can call only the `ib_cfg_*` functions between calls to [ib\\_init\(\)](#) and [ib\\_startup\(\)](#). No other HailDB functions should be called.

#### Returns

DB\_SUCCESS or error code

##### 4.1.2.2 HAILDB\_API [ib\\_err\\_t ib\\_shutdown](#) ( [ib\\_shutdown\\_t](#) *flag* )

Shutdown the InnoDB engine. Call this function when there are no active transactions. It will close all files and release all memory on successful completion. All internal variables will be reset to their default values.

#### Parameters

*flag* is the shutdown flag

#### Returns

DB\_SUCCESS or error code

### 4.1.2.3 HAILDB\_API `ib_err_t ib_startup ( const char * format )`

Startup the InnoDB engine. If this function is called on a non-existent database then based on the default or user specified configuration settings it will create all the necessary files. If the database was shutdown cleanly but the user deleted the REDO log files then it will recreate the REDO log files.

#### Parameters

*format* is the max file format name that the engine supports. Currently this is either Antelope or Barracuda although more may be added in the future without API changes.

#### Returns

DB\_SUCCESS or error code

#### See also

[DB\\_SUCCESS](#)

## 4.2 Cursor functions

### Functions

- HAILDB\_API `ib_err_t ib_cursor_open_table_using_id (ib_id_t table_id, ib_trx_t ib_trx, ib_csr_t *ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_open_index_using_id (ib_id_t index_id, ib_trx_t ib_trx, ib_csr_t *ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_open_index_using_name (ib_csr_t ib_open_csr, const char *index_name, ib_csr_t *ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_open_table (const char *name, ib_trx_t ib_trx, ib_csr_t *ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_reset (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_close (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_prev (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_next (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_first (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_last (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_moveto (ib_csr_t ib_csr, ib_tpl_t ib_tpl, ib_srch_mode_t ib_srch_mode, int *result) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_cursor_attach_trx (ib_csr_t ib_csr, ib_trx_t ib_trx)`
- HAILDB\_API `void ib_cursor_set_match_mode (ib_csr_t ib_csr, ib_match_mode_t match_mode)`
- HAILDB\_API `ib_bool_t ib_cursor_is_positioned (const ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_cursor_stmt_begin (ib_csr_t ib_csr)`

### 4.2.1 Function Documentation

#### 4.2.1.1 HAILDB\_API `void ib_cursor_attach_trx ( ib_csr_t ib_csr, ib_trx_t ib_trx )`

Attach the cursor to the transaction. The cursor must not already be attached to another transaction.

#### Parameters

*ib\_csr* is the cursor instance

*ib\_trx* is the transaction to attach to the cursor

**4.2.1.2 HAILDB\_API `ib_err_t ib_cursor_close ( ib_crsr_t ib_crsr )`**

Close an InnoDB table and free the cursor.

**Parameters**

*ib\_crsr* is an open cursor

**Returns**

DB\_SUCCESS or err code

**4.2.1.3 HAILDB\_API `ib_err_t ib_cursor_first ( ib_crsr_t ib_crsr )`**

Move cursor to the first record in the table.

**Parameters**

*ib\_crsr* is the cursor instance

**Returns**

DB\_SUCCESS or err code

**4.2.1.4 HAILDB\_API `ib_bool_t ib_cursor_is_positioned ( const ib_crsr_t ib_crsr )`**

Check if cursor is positioned.

**Parameters**

*ib\_crsr* is the cursor instance to check

**Returns**

IB\_TRUE if positioned

**4.2.1.5 HAILDB\_API `ib_err_t ib_cursor_last ( ib_crsr_t ib_crsr )`**

Move cursor to the last record in the table.

**Parameters**

*ib\_crsr* is the cursor instance

**Returns**

DB\_SUCCESS or err code

#### 4.2.1.6 HAILDB\_API *ib\_err\_t* *ib\_cursor\_moveto* ( *ib\_crsr\_t* *ib\_crsr*, *ib\_tpl\_t* *ib\_tpl*, *ib\_srch\_mode\_t* *ib\_srch\_mode*, *int* \* *result* )

Search for key.

##### Parameters

*ib\_crsr* is an open cursor instance

*ib\_tpl* is a key to search for

*ib\_srch\_mode* is the search mode

[out] *result* is -1, 0 or 1 depending on tuple eq or gt than the current row

##### Returns

DB\_SUCCESS or err code

#### 4.2.1.7 HAILDB\_API *ib\_err\_t* *ib\_cursor\_next* ( *ib\_crsr\_t* *ib\_crsr* )

Move cursor to the next user record in the table.

##### Parameters

*ib\_crsr* is the cursor instance

##### Returns

DB\_SUCCESS or err code

#### 4.2.1.8 HAILDB\_API *ib\_err\_t* *ib\_cursor\_open\_index\_using\_id* ( *ib\_id\_t* *index\_id*, *ib\_trx\_t* *ib\_trx*, *ib\_crsr\_t* \* *ib\_crsr* )

Open an InnoDB index and return a cursor handle to it.

##### Parameters

*index\_id* is the id of the index to open

*ib\_trx* is the current transaction handle can be NULL

[out] *ib\_crsr* is the new cursor

##### Returns

DB\_SUCCESS or err code

#### 4.2.1.9 HAILDB\_API *ib\_err\_t* *ib\_cursor\_open\_index\_using\_name* ( *ib\_crsr\_t* *ib\_open\_crsr*, *const char* \* *index\_name*, *ib\_crsr\_t* \* *ib\_crsr* )

Open an InnoDB secondary index cursor and return a cursor handle to it.

##### Parameters

*ib\_open\_crsr* is an open cursor

*index\_name* is the name of the index

[out] *ib\_crsr* is the new cursor

### Returns

DB\_SUCCESS or err code

#### 4.2.1.10 HAILDB\_API *ib\_err\_t* *ib\_cursor\_open\_table* ( *const char \* name*, *ib\_trx\_t ib\_trx*, *ib\_crsr\_t \* ib\_crsr* )

Open an InnoDB table by name and return a cursor handle to it.

### Parameters

*name* is the table name to open

*ib\_trx* is the current transaction, can be NULL

*ib\_crsr* is the new cursor

### Returns

DB\_SUCCESS or err code

#### 4.2.1.11 HAILDB\_API *ib\_err\_t* *ib\_cursor\_open\_table\_using\_id* ( *ib\_id\_t table\_id*, *ib\_trx\_t ib\_trx*, *ib\_crsr\_t \* ib\_crsr* )

Open an InnoDB table and return a cursor handle to it.

### Parameters

*table\_id* is the id of the table to open

*ib\_trx* is the current transaction handle, can be NULL

[out] *ib\_crsr* is the new cursor

### Returns

DB\_SUCCESS or err code

#### 4.2.1.12 HAILDB\_API *ib\_err\_t* *ib\_cursor\_prev* ( *ib\_crsr\_t ib\_crsr* )

Move cursor to the prev user record in the table.

### Parameters

*ib\_crsr* is the cursor instance

### Returns

DB\_SUCCESS or err code

**4.2.1.13 HAILDB\_API `ib_err_t ib_cursor_reset ( ib_crsr_t ib_crsr )`**

Reset the cursor.

**Parameters**

*ib\_crsr* is an open cursor

**Returns**

DB\_SUCCESS or err code

**4.2.1.14 HAILDB\_API void `ib_cursor_set_match_mode ( ib_crsr_t ib_crsr, ib_match_mode_t match_mode )`**

Set the match mode for `ib_cursor_move()`.

**Parameters**

*ib\_crsr* is the cursor instance

*match\_mode* is the match mode to set

**4.2.1.15 HAILDB\_API void `ib_cursor_stmt_begin ( ib_crsr_t ib_crsr )`**

Inform the cursor that it's the start of an SQL statement.

**Parameters**

*ib\_crsr* is the cursor instance

## 4.3 Transaction functions

**Functions**

- HAILDB\_API `ib_err_t ib_trx_start (ib_trx_t ib_trx, ib_trx_level_t ib_trx_level) UNIV_NO_IGNORE`
- HAILDB\_API `ib_trx_t ib_trx_begin (ib_trx_level_t ib_trx_level) UNIV_NO_IGNORE`
- HAILDB\_API void `ib_trx_set_client_data (ib_trx_t ib_trx, void *client_data)`
- HAILDB\_API `ib_trx_state_t ib_trx_state (ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_trx_release (ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_trx_commit (ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_trx_rollback (ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_lock (ib_crsr_t ib_crsr, ib_lck_mode_t ib_lck_mode) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_table_lock (ib_trx_t ib_trx, ib_id_t table_id, ib_lck_mode_t ib_lck_mode) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_set_lock_mode (ib_crsr_t ib_crsr, ib_lck_mode_t ib_lck_mode) UNIV_NO_IGNORE`
- HAILDB\_API void `ib_savepoint_take (ib_trx_t ib_trx, const void *name, ib_ulint_t name_len)`

- HAILDB\_API `ib_err_t ib_savepoint_release (ib_trx_t ib_trx, const void *name, ib_ulint_t name_len) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_savepoint_rollback (ib_trx_t ib_trx, const void *name, ib_ulint_t name_len) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_get_duplicate_key (ib_trx_t ib_trx, const char **table_name, const char **index_name)`

### 4.3.1 Function Documentation

#### 4.3.1.1 HAILDB\_API `ib_err_t ib_cursor_lock ( ib_crsr_t ib_crsr, ib_lck_mode_t ib_lck_mode )`

Lock an InnoDB cursor/table.

##### Parameters

- `ib_crsr` is the cursor instance
- `ib_lck_mode` is the lock mode

##### Returns

DB\_SUCCESS or error code

#### 4.3.1.2 HAILDB\_API `ib_err_t ib_cursor_set_lock_mode ( ib_crsr_t ib_crsr, ib_lck_mode_t ib_lck_mode )`

Set the Lock mode of the cursor.

##### Parameters

- `ib_crsr` is the cursor instance for which we want to set the lock mode
- `ib_lck_mode` is the lock mode

##### Returns

DB\_SUCCESS or error code

#### 4.3.1.3 HAILDB\_API `ib_err_t ib_get_duplicate_key ( ib_trx_t ib_trx, const char ** table_name, const char ** index_name )`

Get which key caused a duplicate key error.

In the event of `DB_DUPLICATE_KEY` error, you can call this function immediately after to get the name of the table and index that caused the error.

##### Parameters

- `ib_trx` Transaction where error occurred
- `table_name` pointer to be set to table\_name. Valid until next `ib_` function call. If you would like to keep it, make a copy.
- `index_name` pointer to be set to the index name. Valid until next `ib_` function call. If you would like to keep it, make a copy.

#### 4.3.1.4 HAILDB\_API *ib\_err\_t* *ib\_savepoint\_release* ( *ib\_trx\_t* *ib\_trx*, *const void \** *name*, *ib\_ulint\_t* *name\_len* )

Releases only the named savepoint. Savepoints which were set after this savepoint are left as is.

##### Parameters

*ib\_trx* is the active transaction  
*name* is the name of the savepoint  
*name\_len* is the length of name in bytes

##### Returns

if no savepoint of the name found then DB\_NO\_SAVEPOINT, otherwise DB\_SUCCESS

#### 4.3.1.5 HAILDB\_API *ib\_err\_t* *ib\_savepoint\_rollback* ( *ib\_trx\_t* *ib\_trx*, *const void \** *name*, *ib\_ulint\_t* *name\_len* )

Rolls back a transaction back to a named savepoint. Modifications after the savepoint are undone but InnoDB does NOT release the corresponding locks which are stored in memory. If a lock is 'implicit', that is, a new inserted row holds a lock where the lock information is carried by the trx id stored in the row, these locks are naturally released in the rollback. Savepoints which were set after this savepoint are deleted. If name equals NULL then all the savepoints are rolled back.

##### Parameters

*ib\_trx* is the active transaction  
*name* is the savepoint name can be NULL  
*name\_len* is the length of name in bytes

##### Returns

if no savepoint of the name found then DB\_NO\_SAVEPOINT, otherwise DB\_SUCCESS

#### 4.3.1.6 HAILDB\_API *void* *ib\_savepoint\_take* ( *ib\_trx\_t* *ib\_trx*, *const void \** *name*, *ib\_ulint\_t* *name\_len* )

Creates a named savepoint. The transaction must be started. If there is already a savepoint of the same name, this call erases that old savepoint and replaces it with a new. Savepoints are deleted in a transaction commit or rollback.

##### Parameters

*ib\_trx* is the transaction instance  
*name* is the name of the savepoint  
*name\_len* is the length of name in bytes



**4.3.1.7 HAILDB\_API `ib_err_t ib_table_lock ( ib_trx_t ib_trx, ib_id_t table_id, ib_lck_mode_t ib_lck_mode )`**

Set the Lock an InnoDB table using the table id.

**Parameters**

*ib\_trx* is a transaction instance

*table\_id* is the table to lock

*ib\_lck\_mode* is the lock mode

**Returns**

DB\_SUCCESS or error code

**4.3.1.8 HAILDB\_API `ib_trx_t ib_trx_begin ( ib_trx_level_t ib_trx_level )`**

Begin a transaction. This will allocate a new transaction handle and put the transaction in the active state.

**Parameters**

*ib\_trx\_level* is the transaction isolation level

**Returns**

innobase txn handle

**4.3.1.9 HAILDB\_API `ib_err_t ib_trx_commit ( ib_trx_t ib_trx )`**

Commit a transaction. This function will release the schema latches too. It will also free the transaction handle.

**Parameters**

*ib\_trx* is thr transaction handle

**Returns**

DB\_SUCCESS or err code

**4.3.1.10 HAILDB\_API `ib_err_t ib_trx_release ( ib_trx_t ib_trx )`**

Release the resources of the transaction. If the transaction was selected as a victim by InnoDB and rolled back then use this function to free the transaction handle.

**Parameters**

*ib\_trx* is the transaction handle

**Returns**

DB\_SUCCESS or err code

#### 4.3.1.11 HAILDB\_API `ib_err_t ib_trx_rollback ( ib_trx_t ib_trx )`

Rollback a transaction. This function will release the schema latches too. It will also free the transaction handle.

##### Parameters

*ib\_trx* is the transaction handle

##### Returns

DB\_SUCCESS or err code

#### 4.3.1.12 HAILDB\_API `void ib_trx_set_client_data ( ib_trx_t ib_trx, void * client_data )`

Set client data for a transaction. This is passed back to the client in the `trx_is_interrupted` callback. HailDB will only ever pass this around, it will never dereference it.

##### Parameters

*ib\_trx* is the transaction to set the client data for

*client\_data* is client program's data about this transaction

##### Parameters

*ib\_trx* in: transaction

#### 4.3.1.13 HAILDB\_API `ib_err_t ib_trx_start ( ib_trx_t ib_trx, ib_trx_level_t ib_trx_level )`

Start a transaction that's been rolled back. This special function exists for the case when InnoDB's deadlock detector has rolled back a transaction. While the transaction has been rolled back the handle is still valid and can be reused by calling this function. If you don't want to reuse the transaction handle then you can free the handle by calling `ib_trx_release()`.

##### Parameters

*ib\_trx* is the transaction to restart

*ib\_trx\_level* is the transaction isolation level

##### Returns

innobase txn handle

#### 4.3.1.14 HAILDB\_API `ib_trx_state_t ib_trx_state ( ib_trx_t ib_trx )`

Query the transaction's state. This function can be used to check for the state of the transaction in case it has been rolled back by the InnoDB deadlock detector. Note that when a transaction is selected as a victim for rollback, InnoDB will always return an appropriate error code indicating this.

##### See also

[DB\\_DEADLOCK](#),  
[DB\\_LOCK\\_TABLE\\_FULL](#) and  
[DB\\_LOCK\\_WAIT\\_TIMEOUT](#)

**Parameters**

*ib\_trx* is the transaction handle

**Returns**

transaction state

## 4.4 Configuration functions

**Defines**

- #define `ib_cfg_set_int`(name, value) `ib_cfg_set`(name, value)
- #define `ib_cfg_set_text`(name, value) `ib_cfg_set`(name, value)
- #define `ib_cfg_set_bool_on`(name) `ib_cfg_set`(name, IB\_TRUE)
- #define `ib_cfg_set_bool_off`(name) `ib_cfg_set`(name, IB\_FALSE)
- #define `ib_cfg_set_callback`(name, value) `ib_cfg_set`(name, value)

**Functions**

- HAILDB\_API `ib_err_t ib_cfg_var_get_type` (const char \*name, `ib_cfg_type_t` \*type) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_cfg_set` (const char \*name,...) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_cfg_get` (const char \*name, void \*value) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_cfg_get_all` (const char \*\*\*names, `ib_u32_t` \*names\_num) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_status_get_all` (const char \*\*\*names, `ib_u32_t` \*names\_num) UNIV\_NO\_IGNORE
- HAILDB\_API void `ib_set_panic_handler` (`ib_panic_handler_t` new\_panic\_handler)
- HAILDB\_API void `ib_set_trx_is_interrupted_handler` (`ib_trx_is_interrupted_handler_t` handler)

### 4.4.1 Define Documentation

#### 4.4.1.1 #define `ib_cfg_set_bool_off`( *name* ) `ib_cfg_set`(name, IB\_FALSE)

Set a boolean configuration variable to IB\_FALSE.

**Parameters**

*name* is the config variable name

**Returns**

DB\_SUCCESS or error code

#### 4.4.1.2 #define `ib_cfg_set_bool_on`( *name* ) `ib_cfg_set`(name, IB\_TRUE)

Set a boolean configuration variable to IB\_TRUE.

**Parameters**

*name* is the config variable name

**Returns**

DB\_SUCCESS or error code

**4.4.1.3 #define ib\_cfg\_set\_callback( *name*, *value* ) ib\_cfg\_set(name, value)**

Set a generic ib\_cb\_t callback function.

**Parameters**

*name* is the config variable name

*value* is a pointer to a callback function

**Returns**

DB\_SUCCESS or error code

**4.4.1.4 #define ib\_cfg\_set\_int( *name*, *value* ) ib\_cfg\_set(name, value)**

Set an int configuration variable.

**Parameters**

*name* is the config variable name

*value* is the integer value of the variable

**Returns**

DB\_SUCCESS or error code

**4.4.1.5 #define ib\_cfg\_set\_text( *name*, *value* ) ib\_cfg\_set(name, value)**

Set a text configuration variable.

**Parameters**

*name* is the config variable name

*value* is the char\* value of the variable

**Returns**

DB\_SUCCESS or error code

## 4.4.2 Function Documentation

### 4.4.2.1 HAILDB\_API `ib_err_t ib_cfg_get ( const char * name, void * value )`

Get the value of a configuration variable. The type of the returned value depends on the type of the configuration variable. `DB_SUCCESS` is returned if the variable with name "name" was found and "value" was set.

#### Parameters

*name* is the variable name whose value needs to be accessed

[out] *value* is the value of the variable if found

#### Returns

`DB_SUCCESS` if retrieved successfully

### 4.4.2.2 HAILDB\_API `ib_err_t ib_cfg_get_all ( const char *** names, ib_u32_t * names_num )`

Get a list of the names of all configuration variables. The caller is responsible for `free(3)`ing the returned array of strings when it is not needed anymore and for not modifying the individual strings.

#### Parameters

[out] *names* pointer to array of strings

[out] *names\_num* number of strings returned

#### Returns

`DB_SUCCESS` or error code

### 4.4.2.3 HAILDB\_API `ib_err_t ib_cfg_set ( const char * name, ... )`

Set a configuration variable. The second argument's type depends on the type of the variable with the given "name". Returns `DB_SUCCESS` if the variable with name "name" was found and if its value was set.

#### Parameters

*name* is the config variable name whose value is to be set

#### Returns

`DB_SUCCESS` if set

### 4.4.2.4 HAILDB\_API `ib_err_t ib_cfg_var_get_type ( const char * name, ib_cfg_type_t * type )`

Get the type of a configuration variable. Returns `DB_SUCCESS` if the variable with name "name" was found and "type" was set.

#### Parameters

*name* is the variable name to look up

[out] *type* is the type of the variable name if found

### Returns

DB\_SUCCESS if successful

#### 4.4.2.5 HAILDB\_API void `ib_set_panic_handler ( ib_panic_handler_t new_panic_handler )`

Set panic handler.

HailDB will "panic" upon finding certain forms of corruption. By setting a panic handler, you can implement your own notification to the end user of this corruption (e.g. popping up a dialog box).

### Parameters

*new\_panic\_handler* your panic handler

#### 4.4.2.6 HAILDB\_API void `ib_set_trx_is_interrupted_handler ( ib_trx_is_interrupted_handler_t handler )`

Set `trx_is_interrupted_handler`.

You may specify a callback that HailDB will check during certain wait situations to see if it should abort the operation or not. This lets you implement MySQL/Drizzle KILL command style functionality.

### Parameters

*handler* the `trx_is_interrupted` callback

#### 4.4.2.7 HAILDB\_API `ib_err_t ib_status_get_all ( const char *** names, ib_u32_t * names_num )`

Get a list of the names of all status variables. The caller is responsible for `free(3)`ing the returned array of strings when it is not needed anymore and for not modifying the individual strings.

### Parameters

[out] *names* pointer to array of strings

[out] *names\_num* number of strings returned

### Returns

DB\_SUCCESS or error code

## 4.5 DDL functions

### Functions

- HAILDB\_API `ib_err_t ib_table_schema_add_col ( ib_tbl_sch_t ib_tbl_sch, const char *name, ib_col_type_t ib_col_type, ib_col_attr_t ib_col_attr, ib_u16_t client_type, ib_uint_t len) UNIV_NO_IGNORE`

- HAILDB\_API `ib_err_t ib_table_schema_add_index (ib_tbl_sch_t ib_tbl_sch, const char *name, ib_idx_sch_t *ib_idx_sch) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_table_schema_delete (ib_tbl_sch_t ib_tbl_sch)`
- HAILDB\_API `ib_err_t ib_table_schema_create (const char *name, ib_tbl_sch_t *ib_tbl_sch, ib_tbl_fmt_t ib_tbl_fmt, ib_ulint_t page_size) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_index_schema_add_col (ib_idx_sch_t ib_idx_sch, const char *name, ib_ulint_t prefix_len) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_index_schema_create (ib_trx_t ib_usr_trx, const char *name, const char *table_name, ib_idx_sch_t *ib_idx_sch) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_index_schema_set_clustered (ib_idx_sch_t ib_idx_sch) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_index_schema_set_unique (ib_idx_sch_t ib_idx_sch) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_index_schema_delete (ib_idx_sch_t ib_idx_sch)`
- HAILDB\_API `ib_err_t ib_table_create (ib_trx_t ib_trx, const ib_tbl_sch_t ib_tbl_sch, ib_id_t *id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_table_rename (ib_trx_t ib_trx, const char *old_name, const char *new_name) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_index_create (ib_idx_sch_t ib_idx_sch, ib_id_t *index_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_table_drop (ib_trx_t trx, const char *name) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_index_drop (ib_trx_t trx, ib_id_t index_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_truncate (ib_crsr_t *ib_crsr, ib_id_t *table_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_table_truncate (const char *table_name, ib_id_t *table_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_table_get_id (const char *table_name, ib_id_t *table_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_index_get_id (const char *table_name, const char *index_name, ib_id_t *index_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_bool_t ib_database_create (const char *dbname) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_database_drop (const char *dbname) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_schema_lock_shared (ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_schema_lock_exclusive (ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_bool_t ib_schema_lock_is_exclusive (const ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_schema_unlock (ib_trx_t ib_trx)`
- HAILDB\_API `ib_err_t ib_table_schema_visit (ib_trx_t ib_trx, const char *name, const ib_schema_visitor_t *visitor, void *arg) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_schema_tables_iterate (ib_trx_t ib_trx, ib_schema_visitor_table_all_t visitor, void *arg) UNIV_NO_IGNORE`

## 4.5.1 Function Documentation

### 4.5.1.1 HAILDB\_API `ib_err_t ib_cursor_truncate ( ib_crsr_t * ib_crsr, ib_id_t * table_id )`

Truncate a table. The cursor handle will be closed and set to NULL on success.

#### Parameters

[out] *ib\_crsr* is the cursor for table to truncate

[out] *table\_id* is the new table id

**Returns**

DB\_SUCCESS or error code

**4.5.1.2 HAILDB\_API ib\_bool\_t ib\_database\_create ( const char \* *dbname* )**

Create a database if it doesn't exist.

**Parameters**

*dbname* is the name of the database to create

**Returns**

IB\_TRUE on success

**4.5.1.3 HAILDB\_API ib\_err\_t ib\_database\_drop ( const char \* *dbname* )**

Drop a database if it exists. This function will also drop all tables within the database.

**Parameters**

*dbname* is the name of the database to drop

**Returns**

DB\_SUCCESS or error code

**4.5.1.4 HAILDB\_API ib\_err\_t ib\_index\_create ( ib\_idx\_sch\_t *ib\_idx\_sch*, ib\_id\_t \* *index\_id* )**

Create a secondary index. The index id encodes the table id in the high 4 bytes and the index id in the lower 4 bytes.

**Parameters**

[in, out] *ib\_idx\_sch* the schema for the index

[out] *index\_id* is the new index id that was created

**Returns**

DB\_SUCCESS or err code

**4.5.1.5 HAILDB\_API ib\_err\_t ib\_index\_drop ( ib\_trx\_t *trx*, ib\_id\_t *index\_id* )**

Drop a secondary index. Ensure that you have acquired the schema lock in exclusive mode.

**Parameters**

*trx* is the covering transaction.

*index\_id* is the id of the index to drop

**Returns**

DB\_SUCCESS or err code



**4.5.1.6 HAILDB\_API ib\_err\_t ib\_index\_get\_id ( const char \* *table\_name*, const char \* *index\_name*, ib\_id\_t \* *index\_id* )**

Get an index id.

**Parameters**

*table\_name* is the name of the table that contains the index

*index\_name* is the name of the index to lookup

[out] *index\_id* contains the index id if found

**Returns**

DB\_SUCCESS if found

**4.5.1.7 HAILDB\_API ib\_err\_t ib\_index\_schema\_add\_col ( ib\_idx\_sch\_t *ib\_idx\_sch*, const char \* *name*, ib\_ulint\_t *prefix\_len* )**

Add columns to an index schema definition.

**Parameters**

[in, out] *ib\_idx\_sch* is the index schema instance

*name* is the name of the column to add to the index schema

*prefix\_len* is the prefix length of the index or 0 if no prefix

**Returns**

DB\_SUCCESS or err code

**4.5.1.8 HAILDB\_API ib\_err\_t ib\_index\_schema\_create ( ib\_trx\_t *ib\_usr\_trx*, const char \* *name*, const char \* *table\_name*, ib\_idx\_sch\_t \* *ib\_idx\_sch* )**

Create an index schema instance.

**Parameters**

*ib\_usr\_trx* is the current user transaction

*name* is the name of the index to create

*table\_name* is the name of the table the index belongs to

[out] *ib\_idx\_sch* is the newly created index schema instance

**Returns**

DB\_SUCCESS or err code

**4.5.1.9 HAILDB\_API void ib\_index\_schema\_delete ( ib\_idx\_sch\_t *ib\_idx\_sch* )**

Destroy an index schema.

**Parameters**

*ib\_idx\_sch* is the index schema to delete

**4.5.1.10 HAILDB\_API ib\_err\_t ib\_index\_schema\_set\_clustered ( ib\_idx\_sch\_t ib\_idx\_sch )**

Set index as clustered index. Implies UNIQUE.

**Parameters**

[in, out] *ib\_idx\_sch* is the index schema to update

**Returns**

DB\_SUCCESS or err code

**4.5.1.11 HAILDB\_API ib\_err\_t ib\_index\_schema\_set\_unique ( ib\_idx\_sch\_t ib\_idx\_sch )**

Set index as a unique index.

**Parameters**

[in, out] *ib\_idx\_sch* is the index schema to update

**Returns**

DB\_SUCCESS or err code

**4.5.1.12 HAILDB\_API ib\_err\_t ib\_schema\_lock\_exclusive ( ib\_trx\_t ib\_trx )**

Latches the data dictionary in exclusive mode.

**Parameters**

*ib\_trx* is the transaction instance

**Returns**

DB\_SUCCESS or error code

**4.5.1.13 HAILDB\_API ib\_bool\_t ib\_schema\_lock\_is\_exclusive ( const ib\_trx\_t ib\_trx )**

Checks if the data dictionary is latched in exclusive mode by a user transaction.

**Parameters**

*ib\_trx* is a transaction instance

**Returns**

TRUE if exclusive latch

**4.5.1.14 HAILDB\_API ib\_err\_t ib\_schema\_lock\_shared ( ib\_trx\_t ib\_trx )**

Latches the data dictionary in shared mode.

**Parameters**

*ib\_trx* is the transaction instance

**Returns**

DB\_SUCCESS or error code

**4.5.1.15 HAILDB\_API ib\_err\_t ib\_schema\_tables\_iterate ( ib\_trx\_t ib\_trx, ib\_schema\_visitor\_table\_all\_t visitor, void \* arg )**

List all the tables in the InnoDB's data dictionary. It will abort if visitor returns a non-zero value.

It will call the function: visitor.tables(arg, const char\* name, int name\_len);

The function will abort if visitor.tables() returns non-zero.

**Parameters**

*ib\_trx* is the transaction that owns the schema lock

*visitor* is the visitor function

*arg* argument passed to the visitor function

**Returns**

DB\_SUCCESS if successful

**4.5.1.16 HAILDB\_API ib\_err\_t ib\_schema\_unlock ( ib\_trx\_t ib\_trx )**

Unlocks the data dictionary.

**Parameters**

*ib\_trx* is a transaction instance

**Returns**

DB\_SUCCESS or error code

**4.5.1.17 HAILDB\_API ib\_err\_t ib\_table\_create ( ib\_trx\_t ib\_trx, const ib\_tbl\_sch\_t ib\_tbl\_sch, ib\_id\_t \* id )**

Create a table in the InnoDB data dictionary using the schema definition. If the table exists in the database then this function will return DB\_TABLE\_IS\_BEING\_USED and id will contain that table's id.

**Parameters**

[in, out] *ib\_trx* the current user transaction

*ib\_tbl\_sch* the the schema for the table to create

[out] *id* table id that was created

#### Returns

DB\_SUCCESS or err code

#### 4.5.1.18 HAILDB\_API *ib\_err\_t* *ib\_table\_drop* ( *ib\_trx\_t* *trx*, *const char \** *name* )

Drop a table. Ensure that you have acquired the schema lock in exclusive mode.

#### Parameters

*trx* is the covering transaction.

*name* is the name of the table to drop

#### Returns

DB\_SUCCESS or err code

#### 4.5.1.19 HAILDB\_API *ib\_err\_t* *ib\_table\_get\_id* ( *const char \** *table\_name*, *ib\_id\_t \** *table\_id* )

Get a table id.

#### Parameters

*table\_name* is the name of the table to lookup

[out] *table\_id* is the new table id if found

#### Returns

DB\_SUCCESS if found

#### 4.5.1.20 HAILDB\_API *ib\_err\_t* *ib\_table\_rename* ( *ib\_trx\_t* *ib\_trx*, *const char \** *old\_name*, *const char \** *new\_name* )

Rename a table. Ensure that you have acquired the schema lock in exclusive mode.

#### Parameters

[in, out] *ib\_trx* is the current user transaction

*old\_name* the current name of the table

*new\_name* the new name for the table

#### Returns

DB\_SUCCESS or err code

**4.5.1.21 HAILDB\_API** `ib_err_t ib_table_schema_add_col ( ib_tbl_sch_t ib_tbl_sch, const char * name, ib_col_type_t ib_col_type, ib_col_attr_t ib_col_attr, ib_u16_t client_type, ib_ulint_t len )`

Add columns to a table schema. Tables are created in InnoDB by first creating a table schema which is identified by a handle. Then you add the column definitions to the table schema.

#### Parameters

*ib\_tbl\_sch* is the table schema instance

*name* is the name of the column to add

*ib\_col\_type* is the type of the column

*ib\_col\_attr* are the attributes of the column, including constraints

*client\_type* is any 16 bit number relevant only to the client

*len* is the maximum length of the column

#### Returns

DB\_SUCCESS or err code

**4.5.1.22 HAILDB\_API** `ib_err_t ib_table_schema_add_index ( ib_tbl_sch_t ib_tbl_sch, const char * name, ib_idx_sch_t * ib_idx_sch )`

Create and add an index key definition to a table schema. The index schema is owned by the table schema instance and will be freed when the table schema instance is freed.

#### Parameters

[in, out] *ib\_tbl\_sch* is the schema instance

*name* name of the key definition to create

[out] *ib\_idx\_sch* is the key definition schema instance

#### Returns

DB\_SUCCESS or err code

**4.5.1.23 HAILDB\_API** `ib_err_t ib_table_schema_create ( const char * name, ib_tbl_sch_t * ib_tbl_sch, ib_tbl_fmt_t ib_tbl_fmt, ib_ulint_t page_size )`

Create a table schema.

#### Parameters

*name* is the table name for which to create the schema

[out] *ib\_tbl\_sch* is the schema instance that is created

*ib\_tbl\_fmt* is the format of the table to be created

*page\_size* is the page size for the table or 0 for default

#### Returns

DB\_SUCCESS or err code

#### 4.5.1.24 HAILDB\_API void `ib_table_schema_delete` ( `ib_tbl_sch_t` `ib_tbl_sch` )

Destroy a schema. The handle is freed by this function.

##### Parameters

`ib_tbl_sch` is the table schema to delte

#### 4.5.1.25 HAILDB\_API `ib_err_t` `ib_table_schema_visit` ( `ib_trx_t` `ib_trx`, `const char *` `name`, `const ib_schema_visitor_t *` `visitor`, `void *` `arg` )

Read a table's schema using the visitor pattern. It will make the following sequence of calls:

visitor->table() visitor->table\_col() for each user column visitor->index() for each user index visitor->index\_col() for each column in user index

It will stop if any of the above functions returns a non-zero value. The caller must have an exclusive lock on the InnoDB data dictionary

##### Parameters

`ib_trx` transaction that owns the schema lock

`name` is the table name to read

`visitor` visitor functions to invoke on each definition

`arg` is the argument passed to the visitor functions.

##### Returns

DB\_SUCCESS or DB\_ERROR

#### 4.5.1.26 HAILDB\_API `ib_err_t` `ib_table_truncate` ( `const char *` `table_name`, `ib_id_t *` `table_id` )

Truncate a table.

##### Parameters

`table_name` is the name of the table to truncate

[out] `table_id` is the new table id

##### Returns

DB\_SUCCESS or error code

## 4.6 Miscellaneous functions

### Functions

- HAILDB\_API `ib_u64_t` `ib_api_version` (void) UNIV\_NO\_IGNORE
- HAILDB\_API void `ib_set_client_compare` (`ib_client_cmp_t` `client_cmp_func`)
- HAILDB\_API void `ib_logger_set` (`ib_msg_log_t` `ib_msg_log`, `ib_msg_stream_t` `ib_msg_stream`)
- HAILDB\_API `const char *` `ib_strerror` (`ib_err_t` `db_errno`) UNIV\_NO\_IGNORE

- HAILDB\_API `ib_err_t ib_status_get_i64` (const char \*name, `ib_i64_t` \*dst) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_get_table_statistics` (`ib_csr_t` ib\_csr, `ib_table_stats_t` \*table\_stats, size\_t sizeof\_ib\_table\_stats\_t)
- HAILDB\_API `ib_err_t ib_get_index_stat_n_diff_key_vals` (`ib_csr_t` ib\_csr, const char \*index\_name, `ib_u64_t` \*ncols, `ib_i64_t` \*\*n\_diff)
- HAILDB\_API `ib_err_t ib_update_table_statistics` (`ib_csr_t` csr)

## 4.6.1 Function Documentation

### 4.6.1.1 HAILDB\_API `ib_u64_t ib_api_version` ( void )

Return the API version number, the version number format is: | 16 bits future use | 16 bits current | 16 bits revision | 16 bits age |

- If the library source code has changed at all since the last release, then revision will be incremented ('c:r:a' becomes 'c:r+1:a').
- If any interfaces have been added, removed, or changed since the last update, current will be incremented, and revision will be set to 0.
- If any interfaces have been added (but not changed or removed) since the last release, then age will be incremented.
- If any interfaces have been changed or removed since the last release, then age will be set to 0.

#### Returns

API version number

### 4.6.1.2 HAILDB\_API `ib_err_t ib_get_index_stat_n_diff_key_vals` ( `ib_csr_t` *ib\_csr*, const char \* *index\_name*, `ib_u64_t` \* *ncols*, `ib_i64_t` \*\* *n\_diff* )

Get statistics on number of different key values per index part

This function returns the approximate different key values for this index. They are periodically recalculated.

#### Parameters

*ib\_csr* A Cursor that is opened to a table

*index\_name* name of the index

*ncols* returns the number of elements in *n\_diff*

*n\_diff* An array allocated with malloc() (user needs to free()) containing the statistics

#### Returns

`DB_SUCCESS` or error. `DB_NOT_FOUND` if index is not found

#### 4.6.1.3 HAILDB\_API `ib_err_t ib_get_table_statistics ( ib_crsr_t ib_crsr, ib_table_stats_t * table_stats, size_t sizeof_ib_table_stats_t )`

Get table statistics.

This function will fill out the provided `ib_table_stats_t` with statistics about the table on the currently opened `ib_crsr_t`

##### Parameters

*ib\_crsr* A Cursor that is opened to a table

*table\_stats* a `ib_table_stats_t` to be filled out by HailDB

*sizeof\_ib\_table\_stats\_t* `sizeof(ib_table_stats_t)`. This allows for ABI compatible changes to the size of `ib_table_stats_t`.

##### Returns

`DB_SUCCESS` or error

#### 4.6.1.4 HAILDB\_API `void ib_logger_set ( ib_msg_log_t ib_msg_log, ib_msg_stream_t ib_msg_stream )`

Set the message logging function.

##### Parameters

*ib\_msg\_log* is the message logging function

*ib\_msg\_stream* is the message stream, this is the first argument to the loggingfunction

#### 4.6.1.5 HAILDB\_API `void ib_set_client_compare ( ib_client_cmp_t client_cmp_func )`

Set the client comparison function for BLOBs and client types.

##### Parameters

*client\_cmp\_func* is the index key compare callback function

#### 4.6.1.6 HAILDB\_API `ib_err_t ib_status_get_i64 ( const char * name, ib_i64_t * dst )`

Get the value of an INT status variable.

##### Parameters

*name* is the status variable name

[out] *dst* is where the output value is copied if name is found

##### Returns

`DB_SUCCESS` if found and type is INT, `DB_DATA_MISMATCH` if found but type is not INT, `DB_NOT_FOUND` otherwise.



**4.6.1.7 HAILDB\_API const char\* ib\_strerror ( ib\_err\_t db\_errno )**

Convert an error number to a human readable text message. The returned string is static and should not be freed or modified.

**Parameters**

*db\_errno* is the error number

**Returns**

string, describing the error

**4.6.1.8 HAILDB\_API ib\_err\_t ib\_update\_table\_statistics ( ib\_cursor\_t csr )**

Force an update of table and index statistics

This function forces an update to the table and index statistics for the table *csr* is opened on.

**Parameters**

*csr* A Cursor that is opened to a table

**Returns**

[DB\\_SUCCESS](#) or error.

## 4.7 Tuple functions

### Functions

- HAILDB\_API [ib\\_tpl\\_t ib\\_tuple\\_clear](#) (ib\_tpl\_t ib\_tpl) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_err\\_t ib\\_tuple\\_get\\_cluster\\_key](#) (ib\_cursor\_t ib\_csr, [ib\\_tpl\\_t](#) \*ib\_dst\_tpl, const [ib\\_tpl\\_t](#) ib\_src\_tpl) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_err\\_t ib\\_tuple\\_copy](#) (ib\_tpl\_t ib\_dst\_tpl, const [ib\\_tpl\\_t](#) ib\_src\_tpl) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_tpl\\_t ib\\_sec\\_search\\_tuple\\_create](#) (ib\_cursor\_t ib\_csr) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_tpl\\_t ib\\_sec\\_read\\_tuple\\_create](#) (ib\_cursor\_t ib\_csr) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_tpl\\_t ib\\_clust\\_search\\_tuple\\_create](#) (ib\_cursor\_t ib\_csr) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_tpl\\_t ib\\_clust\\_read\\_tuple\\_create](#) (ib\_cursor\_t ib\_csr) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_ulint\\_t ib\\_tuple\\_get\\_n\\_user\\_cols](#) (const [ib\\_tpl\\_t](#) ib\_tpl) UNIV\_NO\_IGNORE
- HAILDB\_API [ib\\_ulint\\_t ib\\_tuple\\_get\\_n\\_cols](#) (const [ib\\_tpl\\_t](#) ib\_tpl) UNIV\_NO\_IGNORE
- HAILDB\_API void [ib\\_tuple\\_delete](#) (ib\_tpl\_t ib\_tpl)

### 4.7.1 Function Documentation

**4.7.1.1 HAILDB\_API ib\_tpl\_t ib\_clust\_read\_tuple\_create ( ib\_cursor\_t ib\_csr )**

Create an InnoDB tuple for table row operations.

**Parameters**

*ib\_csr* is the cursor instance

**Returns**

tTuple for current table

**4.7.1.2 HAILDB\_API ib\_tpl\_t ib\_clust\_search\_tuple\_create ( ib\_crsr\_t ib\_crsr )**

Create an InnoDB tuple used for table key operations.

**Parameters**

*ib\_crsr* is the cursor instance

**Returns**

tuple for current table

**4.7.1.3 HAILDB\_API ib\_tpl\_t ib\_sec\_read\_tuple\_create ( ib\_crsr\_t ib\_crsr )**

Create an InnoDB tuple used for index/table search.

**Parameters**

*ib\_crsr* is the cursor instance

**Returns**

tuple for current index

**4.7.1.4 HAILDB\_API ib\_tpl\_t ib\_sec\_search\_tuple\_create ( ib\_crsr\_t ib\_crsr )**

Create an InnoDB tuple used for index/table search.

**Parameters**

*ib\_crsr* is the cursor instance

**Returns**

tuple for current index

**4.7.1.5 HAILDB\_API ib\_tpl\_t ib\_tuple\_clear ( ib\_tpl\_t ib\_tpl )**

"Clear" or reset an InnoDB tuple. We free the heap and recreate the tuple.

**Parameters**

*ib\_tpl* is the tuple to be freed

**Returns**

new tuple, or NULL

**4.7.1.6 HAILDB\_API `ib_err_t ib_tuple_copy ( ib_tpl_t ib_dst_tpl, const ib_tpl_t ib_src_tpl )`**

Copy the contents of source tuple to destination tuple. The tuples must be of the same type and belong to the same table/index.

**Parameters**

*ib\_dst\_tpl* is the destination tuple

*ib\_src\_tpl* is the source tuple

**Returns**

DB\_SUCCESS or error code

**4.7.1.7 HAILDB\_API `void ib_tuple_delete ( ib_tpl_t ib_tpl )`**

Destroy an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple instance to delete

**4.7.1.8 HAILDB\_API `ib_err_t ib_tuple_get_cluster_key ( ib_csr_t ib_csr, ib_tpl_t * ib_dst_tpl, const ib_tpl_t ib_src_tpl )`**

Create a new cluster key search tuple and copy the contents of the secondary index key tuple columns that refer to the cluster index record to the cluster key. It does a deep copy of the column data.

**Parameters**

*ib\_csr* is a cursor opened on a secondary index

[out] *ib\_dst\_tpl* is the tuple where the key data will be copied

*ib\_src\_tpl* is the source secondary index tuple to copy from

**Returns**

DB\_SUCCESS or error code

**4.7.1.9 HAILDB\_API `ib_ulint_t ib_tuple_get_n_cols ( const ib_tpl_t ib_tpl )`**

Return the number of columns in the tuple definition.

**Parameters**

*ib\_tpl* is a tuple

**Returns**

number of columns

#### 4.7.1.10 HAILDB\_API `ib_ulint_t ib_tuple_get_n_user_cols ( const ib_tpl_t ib_tpl )`

Return the number of user columns in the tuple definition.

##### Parameters

*ib\_tpl* is a tuple

##### Returns

number of user columns

## 4.8 SQL functions

### Functions

- HAILDB\_API void `ib_cursor_set_simple_select (ib_csr_t ib_csr)`

#### 4.8.1 Function Documentation

##### 4.8.1.1 HAILDB\_API void `ib_cursor_set_simple_select ( ib_csr_t ib_csr )`

Set to true if it's a simple select.

##### Parameters

[in, out] *ib\_csr* is the cursor to update

## 4.9 DML functions

### Functions

- HAILDB\_API `ib_err_t ib_cursor_insert_row (ib_csr_t ib_csr, const ib_tpl_t ib_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_update_row (ib_csr_t ib_csr, const ib_tpl_t ib_old_tpl, const ib_tpl_t ib_new_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_delete_row (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_read_row (ib_csr_t ib_csr, ib_tpl_t ib_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_col_set_value (ib_tpl_t ib_tpl, ib_ulint_t col_no, const void *src, ib_ulint_t len) UNIV_NO_IGNORE`
- HAILDB\_API `ib_ulint_t ib_col_get_len (ib_tpl_t ib_tpl, ib_ulint_t i) UNIV_NO_IGNORE`
- HAILDB\_API `ib_ulint_t ib_col_copy_value (ib_tpl_t ib_tpl, ib_ulint_t i, void *dst, ib_ulint_t len)`
- HAILDB\_API `ib_err_t ib_tuple_read_i8 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_i8_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_u8 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_u8_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_i16 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_i16_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_u16 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_u16_t *ival) UNIV_NO_IGNORE`

- HAILDB\_API `ib_err_t ib_tuple_read_i32 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_i32_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_u32 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_u32_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_i64 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_i64_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_u64 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_u64_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `const void * ib_col_get_value (ib_tpl_t ib_tpl, ib_ulint_t i) UNIV_NO_IGNORE`
- HAILDB\_API `ib_ulint_t ib_col_get_meta (ib_tpl_t ib_tpl, ib_ulint_t i, ib_col_meta_t *ib_col_meta)`
- HAILDB\_API `void ib_cursor_set_cluster_access (ib_crsr_t ib_crsr)`
- HAILDB\_API `ib_err_t ib_tuple_write_i8 (ib_tpl_t ib_tpl, int col_no, ib_i8_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_i16 (ib_tpl_t ib_tpl, int col_no, ib_i16_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_i32 (ib_tpl_t ib_tpl, int col_no, ib_i32_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_i64 (ib_tpl_t ib_tpl, int col_no, ib_i64_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_u8 (ib_tpl_t ib_tpl, int col_no, ib_u8_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_u16 (ib_tpl_t ib_tpl, int col_no, ib_u16_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_u32 (ib_tpl_t ib_tpl, int col_no, ib_u32_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_u64 (ib_tpl_t ib_tpl, int col_no, ib_u64_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_double (ib_tpl_t ib_tpl, int col_no, double val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_double (ib_tpl_t ib_tpl, ib_ulint_t col_no, double *dval) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_float (ib_tpl_t ib_tpl, int col_no, float val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_float (ib_tpl_t ib_tpl, ib_ulint_t col_no, float *fval) UNIV_NO_IGNORE`

## 4.9.1 Function Documentation

### 4.9.1.1 HAILDB\_API `ib_ulint_t ib_col_copy_value ( ib_tpl_t ib_tpl, ib_ulint_t i, void * dst, ib_ulint_t len )`

Copy a column value from the tuple.

#### Parameters

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *dst* is where the data will be copied

*len* is the maximum number of bytes that can be copied to *dst*

#### Returns

bytes copied or `IB_SQL_NULL`

**4.9.1.2 HAILDB\_API `ib_ulint_t ib_col_get_len ( ib_tpl_t ib_tpl, ib_ulint_t i )`**

Get the size of the data available in the column the tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

**Returns**

bytes avail or IB\_SQL\_NULL

**4.9.1.3 HAILDB\_API `ib_ulint_t ib_col_get_meta ( ib_tpl_t ib_tpl, ib_ulint_t i, ib_col_meta_t * ib_col_meta )`**

Get a column type, length and attributes from the tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ib\_col\_meta* the column meta data

**Returns**

len of column data

**4.9.1.4 HAILDB\_API `const void* ib_col_get_value ( ib_tpl_t ib_tpl, ib_ulint_t i )`**

Get a column value pointer from the tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

**Returns**

NULL or pointer to buffer

**4.9.1.5 HAILDB\_API `ib_err_t ib_col_set_value ( ib_tpl_t ib_tpl, ib_ulint_t col_no, const void * src, ib_ulint_t len )`**

Set a column of the tuple. Make a copy using the tuple's heap.

**Parameters**

*ib\_tpl* is the tuple instance

*col\_no* is the column index in the tuple

*src* is the data value to set

*len* is the data value (*src*) length in bytes

#### Returns

DB\_SUCCESS or error code

#### 4.9.1.6 HAILDB\_API *ib\_err\_t* *ib\_cursor\_delete\_row* ( *ib\_csr\_t* *ib\_csr* )

Delete a row in a table.

#### Parameters

*ib\_csr* is the cursor instance

#### Returns

DB\_SUCCESS or err code

#### 4.9.1.7 HAILDB\_API *ib\_err\_t* *ib\_cursor\_insert\_row* ( *ib\_csr\_t* *ib\_csr*, const *ib\_tpl\_t* *ib\_tpl* )

Insert a row to a table.

#### Parameters

*ib\_csr* is an open cursor

*ib\_tpl* is the tuple to insert

#### Returns

DB\_SUCCESS or err code

#### 4.9.1.8 HAILDB\_API *ib\_err\_t* *ib\_cursor\_read\_row* ( *ib\_csr\_t* *ib\_csr*, *ib\_tpl\_t* *ib\_tpl* )

Read current row.

#### Parameters

*ib\_csr* is the cursor instance

[out] *ib\_tpl* is the tuple to read the column values

#### Returns

DB\_SUCCESS or err code

#### 4.9.1.9 HAILDB\_API void *ib\_cursor\_set\_cluster\_access* ( *ib\_csr\_t* *ib\_csr* )

Set need to access clustered index record flag.

#### Parameters

*ib\_csr* is the cursor instance for which we want to set the flag

**4.9.1.10 HAILDB\_API *ib\_err\_t* *ib\_cursor\_update\_row* ( *ib\_crsr\_t* *ib\_crsr*, const *ib\_tpl\_t* *ib\_old\_tpl*, const *ib\_tpl\_t* *ib\_new\_tpl* )**

Update a row in a table.

**Parameters**

*ib\_crsr* is the cursor instance

*ib\_old\_tpl* is the old tuple in the table

*ib\_new\_tpl* is the new tuple with the updated values

**Returns**

DB\_SUCCESS or err code

**4.9.1.11 HAILDB\_API *ib\_err\_t* *ib\_tuple\_read\_double* ( *ib\_tpl\_t* *ib\_tpl*, *ib\_ulint\_t* *col\_no*, double \* *dval* )**

Read a double column value from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple to read from

*col\_no* is the column number to read

[out] *dval* is where the value is copied

**Returns**

DB\_SUCCESS or error

**4.9.1.12 HAILDB\_API *ib\_err\_t* *ib\_tuple\_read\_float* ( *ib\_tpl\_t* *ib\_tpl*, *ib\_ulint\_t* *col\_no*, float \* *fval* )**

Read a float value from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple to read from

*col\_no* is the column number to read

[out] *fval* is where the value is copied

**Returns**

DB\_SUCCESS or error

**4.9.1.13 HAILDB\_API *ib\_err\_t* *ib\_tuple\_read\_i16* ( *ib\_tpl\_t* *ib\_tpl*, *ib\_ulint\_t* *i*, *ib\_i16\_t* \* *ival* )**

Read a signed int 16 bit column from an InnoDB tuple.



**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ival* is the integer value

**Returns**

DB\_SUCCESS or error

**4.9.1.14 HAILDB\_API ib\_err\_t ib\_tuple\_read\_i32 ( ib\_tpl\_t *ib\_tpl*, ib\_ulint\_t *i*, ib\_i32\_t \* *ival* )**

Read a signed int 32 bit column from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ival* is the integer value

**Returns**

DB\_SUCCESS or error

**4.9.1.15 HAILDB\_API ib\_err\_t ib\_tuple\_read\_i64 ( ib\_tpl\_t *ib\_tpl*, ib\_ulint\_t *i*, ib\_i64\_t \* *ival* )**

Read a signed int 64 bit column from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ival* is the integer value

**Returns**

DB\_SUCCESS or error

**4.9.1.16 HAILDB\_API ib\_err\_t ib\_tuple\_read\_i8 ( ib\_tpl\_t *ib\_tpl*, ib\_ulint\_t *i*, ib\_i8\_t \* *ival* )**

Read a signed int 8 bit column from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ival* is the integer value

**Returns**

DB\_SUCCESS or error

**4.9.1.17 HAILDB\_API** `ib_err_t ib_tuple_read_u16 ( ib_tpl_t ib_tpl, ib_ulint_t i, ib_u16_t * ival )`

Read an unsigned int 16 bit column from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ival* is the integer value

**Returns**

DB\_SUCCESS or error

**4.9.1.18 HAILDB\_API** `ib_err_t ib_tuple_read_u32 ( ib_tpl_t ib_tpl, ib_ulint_t i, ib_u32_t * ival )`

Read an unsigned int 32 bit column from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ival* is the integer value

**Returns**

DB\_SUCCESS or error

**4.9.1.19 HAILDB\_API** `ib_err_t ib_tuple_read_u64 ( ib_tpl_t ib_tpl, ib_ulint_t i, ib_u64_t * ival )`

Read an unsigned int 64 bit column from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ival* is the integer value

**Returns**

DB\_SUCCESS or error

**4.9.1.20 HAILDB\_API** `ib_err_t ib_tuple_read_u8 ( ib_tpl_t ib_tpl, ib_ulint_t i, ib_u8_t * ival )`

Read an unsigned int 8 bit column from an InnoDB tuple.

**Parameters**

*ib\_tpl* is the tuple instance

*i* is the index (ordinal position) of the column within the tuple

[out] *ival* is the integer value

**Returns**

DB\_SUCCESS or error

**4.9.1.21 HAILDB\_API ib\_err\_t ib\_tuple\_write\_double ( ib\_tpl\_t *ib\_tpl*, int *col\_no*, double *val* )**

Write a double value to a column.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to

*col\_no* is the column number to update

*val* is the value to write

**Returns**

DB\_SUCCESS or error

**4.9.1.22 HAILDB\_API ib\_err\_t ib\_tuple\_write\_float ( ib\_tpl\_t *ib\_tpl*, int *col\_no*, float *val* )**

Write a float value to a column.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to

*col\_no* is the column number to update

*val* is the value to write

**Returns**

DB\_SUCCESS or error

**4.9.1.23 HAILDB\_API ib\_err\_t ib\_tuple\_write\_i16 ( ib\_tpl\_t *ib\_tpl*, int *col\_no*, ib\_i16\_t *val* )**

Write an integer value to a column. Integers are stored in big-endian format and will need to be converted from the host format.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to

*col\_no* is the column number to update

*val* is the value to write

**Returns**

DB\_SUCCESS or error

**4.9.1.24 HAILDB\_API `ib_err_t ib_tuple_write_i32 ( ib_tpl_t ib_tpl, int col_no, ib_i32_t val )`**

Write an integer value to a column. Integers are stored in big-endian format and will need to be converted from the host format.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to  
*col\_no* is the column number to update  
*val* is the value to write

**Returns**

DB\_SUCESS or error

**4.9.1.25 HAILDB\_API `ib_err_t ib_tuple_write_i64 ( ib_tpl_t ib_tpl, int col_no, ib_i64_t val )`**

Write an integer value to a column. Integers are stored in big-endian format and will need to be converted from the host format.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to  
*col\_no* is the column number to update  
*val* is the value to write

**Returns**

DB\_SUCESS or error

**4.9.1.26 HAILDB\_API `ib_err_t ib_tuple_write_i8 ( ib_tpl_t ib_tpl, int col_no, ib_i8_t val )`**

Write an integer value to a column. Integers are stored in big-endian format and will need to be converted from the host format.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to  
*col\_no* is the column number to update  
*val* is the value to write

**Returns**

DB\_SUCESS or error

**4.9.1.27 HAILDB\_API `ib_err_t ib_tuple_write_u16 ( ib_tpl_t ib_tpl, int col_no, ib_u16_t val )`**

Write an integer value to a column. Integers are stored in big-endian format and will need to be converted from the host format.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to  
*col\_no* is the column number to update  
*val* is the value to write

**Returns**

DB\_SUCESS or error

**4.9.1.28 HAILDB\_API ib\_err\_t ib\_tuple\_write\_u32 ( ib\_tpl\_t ib\_tpl, int col\_no, ib\_u32\_t val )**

Write an integer value to a column. Integers are stored in big-endian format and will need to be converted from the host format.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to  
*col\_no* is the column number to update  
*val* is the value to write

**Returns**

DB\_SUCESS or error

**4.9.1.29 HAILDB\_API ib\_err\_t ib\_tuple\_write\_u64 ( ib\_tpl\_t ib\_tpl, int col\_no, ib\_u64\_t val )**

Write an integer value to a column. Integers are stored in big-endian format and will need to be converted from the host format.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to  
*col\_no* is the column number to update  
*val* is the value to write

**Returns**

DB\_SUCESS or error

**4.9.1.30 HAILDB\_API ib\_err\_t ib\_tuple\_write\_u8 ( ib\_tpl\_t ib\_tpl, int col\_no, ib\_u8\_t val )**

Write an integer value to a column. Integers are stored in big-endian format and will need to be converted from the host format.

**Parameters**

[in, out] *ib\_tpl* is the tuple to write to  
*col\_no* is the column number to update  
*val* is the value to write

**Returns**

DB\_SUCESS or error

## 4.10 Debug and Testing functions

### Functions

- HAILDB\_API [ib\\_err\\_t ib\\_error\\_inject](#) (int error\_to\_inject)

#### 4.10.1 Function Documentation

##### 4.10.1.1 HAILDB\_API [ib\\_err\\_t ib\\_error\\_inject](#) ( int *error\_to\_inject* )

Inject an error into HailDB

This function will simulate an error condition inside HailDB. You should not rely on this function. It is for HailDB test suite use only, parts may only be compiled into debug libraries and this function can quite legitimately just return DB\_ERROR and cause Voldemort to pay you a visit.

#### Parameters

*error\_to\_inject* The error inject code to insert.

# Chapter 5

## Data Structure Documentation

### 5.1 `ib_col_meta_t` Struct Reference

#### Data Fields

- [ib\\_col\\_type\\_t](#) type
- [ib\\_col\\_attr\\_t](#) attr
- [ib\\_u32\\_t](#) type\_len
- [ib\\_u16\\_t](#) client\_type
- [ib\\_charset\\_t](#) \* charset

#### 5.1.1 Detailed Description

meta data.

#### 5.1.2 Field Documentation

##### 5.1.2.1 `ib_col_attr_t` attr

Column attributes

##### 5.1.2.2 `ib_charset_t`\* charset

Column charset

##### 5.1.2.3 `ib_u16_t` client\_type

16 bits of data relevant only to the client. InnoDB doesn't care

##### 5.1.2.4 `ib_col_type_t` type

Type of the column

### 5.1.2.5 `ib_u32_t type_len`

Length of type

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

- [haildb.h](#)

## 5.2 `ib_schema_visitor_t` Struct Reference

```
#include <haildb.h>
```

### Data Fields

- [ib\\_schema\\_visitor\\_version\\_t version](#)
- [ib\\_schema\\_visitor\\_table\\_t table](#)
- [ib\\_schema\\_visitor\\_table\\_col\\_t table\\_col](#)
- [ib\\_schema\\_visitor\\_index\\_t index](#)
- [ib\\_schema\\_visitor\\_index\\_col\\_t index\\_col](#)

### 5.2.1 Detailed Description

Callback functions to traverse the schema of a table.

### 5.2.2 Field Documentation

#### 5.2.2.1 `ib_schema_visitor_index_t index`

For traversing index info

#### 5.2.2.2 `ib_schema_visitor_index_col_t index_col`

For traversing index column info

#### 5.2.2.3 `ib_schema_visitor_table_t table`

For traversing table info

#### 5.2.2.4 `ib_schema_visitor_table_col_t table_col`

For traversing table column info

#### 5.2.2.5 `ib_schema_visitor_version_t version`

Visitor version

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



- [haildb.h](#)

## 5.3 `ib_table_stats_t` Struct Reference

### Data Fields

- [ib\\_i64\\_t stat\\_n\\_rows](#)
- [ib\\_u64\\_t stat\\_clustered\\_index\\_size](#)
- [ib\\_u64\\_t stat\\_sum\\_of\\_other\\_index\\_sizes](#)
- [ib\\_u64\\_t stat\\_modified\\_counter](#)

### 5.3.1 Detailed Description

and index statistics.

### 5.3.2 Field Documentation

#### 5.3.2.1 `ib_u64_t stat_clustered_index_size`

approximate clustered index size in bytes.

#### 5.3.2.2 `ib_u64_t stat_modified_counter`

when a row is inserted, updated, or deleted, we add 1 to this number; we calculate new estimates for the `stat_...` values for the table and the indexes at an interval of 2 GB or when about 1 / 16 of table has been modified; also when an estimate operation is called for; the counter is reset to zero at statistics calculation; this counter is not protected by any latch, because this is only used for heuristics

#### 5.3.2.3 `ib_i64_t stat_n_rows`

approximate number of rows in the table; we periodically calculate new estimates

#### 5.3.2.4 `ib_u64_t stat_sum_of_other_index_sizes`

other indexes in bytes

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

- [haildb.h](#)



# Chapter 6

## File Documentation

### 6.1 haildb.h File Reference

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

#### Data Structures

- struct [ib\\_col\\_meta\\_t](#)
- struct [ib\\_schema\\_visitor\\_t](#)
- struct [ib\\_table\\_stats\\_t](#)

#### Defines

- #define [HAILDB\\_API](#)
- #define [HAILDB\\_LOCAL](#)
- #define [UNIV\\_NO\\_IGNORE](#)
- #define [IB\\_TRUE](#) 0x1UL
- #define [IB\\_FALSE](#) 0x0UL
- #define [IB\\_SQL\\_NULL](#) 0xFFFFFFFF
- #define [IB\\_N\\_SYS\\_COLS](#) 3
- #define [MAX\\_TEXT\\_LEN](#) 4096
- #define [IB\\_MAX\\_COL\\_NAME\\_LEN](#) (64 \* 3)
- #define [IB\\_MAX\\_TABLE\\_NAME\\_LEN](#) (64 \* 3)
- #define [ib\\_tbl\\_sch\\_add\\_blob\\_col\(s, n\)](#) [ib\\_table\\_schema\\_add\\_col\(s, n, IB\\_BLOB, IB\\_COL\\_NONE, 0, 0\)](#)
- #define [ib\\_tbl\\_sch\\_add\\_text\\_col\(s, n\)](#) [ib\\_table\\_schema\\_add\\_col\(s, n, IB\\_VARCHAR, IB\\_COL\\_NONE, 0, MAX\\_TEXT\\_LEN\)](#)
- #define [ib\\_tbl\\_sch\\_add\\_varchar\\_col\(s, n, l\)](#) [ib\\_table\\_schema\\_add\\_col\(s, n, IB\\_VARCHAR, IB\\_COL\\_NONE, 0, l\)](#)
- #define [ib\\_tbl\\_sch\\_add\\_u32\\_col\(s, n\)](#) [ib\\_table\\_schema\\_add\\_col\(s, n, IB\\_INT, IB\\_COL\\_UNSIGNED, 0, 4\)](#)
- #define [ib\\_tbl\\_sch\\_add\\_u64\\_col\(s, n\)](#) [ib\\_table\\_schema\\_add\\_col\(s, n, IB\\_INT, IB\\_COL\\_UNSIGNED, 0, 8\)](#)

- #define `ib_tbl_sch_add_u64_notnull_col(s, n)`
- #define `ib_cfg_set_int(name, value) ib_cfg_set(name, value)`
- #define `ib_cfg_set_text(name, value) ib_cfg_set(name, value)`
- #define `ib_cfg_set_bool_on(name) ib_cfg_set(name, IB_TRUE)`
- #define `ib_cfg_set_bool_off(name) ib_cfg_set(name, IB_FALSE)`
- #define `ib_cfg_set_callback(name, value) ib_cfg_set(name, value)`

## Typedefs

- typedef enum `db_err` `ib_err_t`
- typedef unsigned char `ib_byte_t`
- typedef unsigned long int `ib_ulint_t`
- typedef void \* `ib_opaque_t`
- typedef `ib_ulint_t` `ib_bool_t`
- typedef `ib_opaque_t` `ib_charset_t`
- typedef int8\_t `ib_i8_t`
- typedef uint8\_t `ib_u8_t`
- typedef int16\_t `ib_i16_t`
- typedef uint16\_t `ib_u16_t`
- typedef int32\_t `ib_i32_t`
- typedef uint32\_t `ib_u32_t`
- typedef int64\_t `ib_i64_t`
- typedef uint64\_t `ib_u64_t`
- typedef `ib_u64_t` `ib_id_t`
- typedef void(\* `ib_cb_t`)(void)
- typedef FILE \* `ib_msg_stream_t`
- typedef int(\* `ib_msg_log_t`)(`ib_msg_stream_t`, const char \*,...)
- typedef struct `ib_tpl_struct` \* `ib_tpl_t`
- typedef struct `ib_trx_struct` \* `ib_trx_t`
- typedef struct `ib_crsr_struct` \* `ib_crsr_t`
- typedef struct `ib_tbl_sch_struct` \* `ib_tbl_sch_t`
- typedef struct `ib_idx_sch_struct` \* `ib_idx_sch_t`
- typedef int(\* `ib_schema_visitor_table_all_t`)(void \*arg, const char \*name, int name\_len)
- typedef int(\* `ib_schema_visitor_table_t`)(void \*arg, const char \*name, `ib_tbl_fmt_t` tbl\_fmt, `ib_ulint_t` page\_size, int n\_cols, int n\_indexes)
- typedef int(\* `ib_schema_visitor_table_col_t`)(void \*arg, const char \*name, `ib_col_type_t` col\_type, `ib_ulint_t` len, `ib_col_attr_t` attr)
- typedef int(\* `ib_schema_visitor_index_t`)(void \*arg, const char \*name, `ib_bool_t` clustered, `ib_bool_t` unique, int n\_cols)
- typedef int(\* `ib_schema_visitor_index_col_t`)(void \*arg, const char \*name, `ib_ulint_t` prefix\_len)
- typedef int(\* `ib_client_cmp_t`)(const `ib_col_meta_t` \*col\_meta, const `ib_byte_t` \*p1, `ib_ulint_t` p1\_len, const `ib_byte_t` \*p2, `ib_ulint_t` p2\_len)
- typedef void(\* `ib_panic_handler_t`)(void \*, int, char \*,...)
- typedef int(\* `ib_trx_is_interrupted_handler_t`)(void \*)

## Enumerations

- enum `db_err` {
  - `DB_SUCCESS` = 10, `DB_ERROR`, `DB_INTERRUPTED`, `DB_OUT_OF_MEMORY`,
  - `DB_OUT_OF_FILE_SPACE`, `DB_LOCK_WAIT`, `DB_DEADLOCK`, `DB_ROLLBACK`,
  - `DB_DUPLICATE_KEY`, `DB_QUE_THR_SUSPENDED`, `DB_MISSING_HISTORY`, `DB_CLUSTER_NOT_FOUND` = 30,
  - `DB_TABLE_NOT_FOUND`, `DB_MUST_GET_MORE_FILE_SPACE`, `DB_TABLE_IS_BEING_USED`, `DB_TOO_BIG_RECORD`,
  - `DB_LOCK_WAIT_TIMEOUT`, `DB_NO_REFERENCED_ROW`, `DB_ROW_IS_REFERENCED`, `DB_CANNOT_ADD_CONSTRAINT`,
  - `DB_CORRUPTION`, `DB_COL_APPEARS_TWICE_IN_INDEX`, `DB_CANNOT_DROP_CONSTRAINT`, `DB_NO_SAVEPOINT`,
  - `DB_TABLESPACE_ALREADY_EXISTS`, `DB_TABLESPACE_DELETED`, `DB_LOCK_TABLE_FULL`, `DB_FOREIGN_DUPLICATE_KEY`,
  - `DB_TOO_MANY_CONCURRENT_TRXS`, `DB_UNSUPPORTED`, `DB_PRIMARY_KEY_IS_NULL`, `DB_FATAL`,
  - `DB_FAIL` = 1000, `DB_OVERFLOW`, `DB_UNDERFLOW`, `DB_STRONG_FAIL`,
  - `DB_ZIP_OVERFLOW`, `DB_RECORD_NOT_FOUND` = 1500, `DB_END_OF_INDEX`, `DB_SCHEMA_ERROR` = 2000,
  - `DB_DATA_MISMATCH`, `DB_SCHEMA_NOT_LOCKED`, `DB_NOT_FOUND`, `DB_READONLY`,
  - `DB_INVALID_INPUT` }
- enum `ib_cfg_type_t` {
  - `IB_CFG_IBOOL`, `IB_CFG_ULINT`, `IB_CFG_ULONG`, `IB_CFG_TEXT`,
  - `IB_CFG_CB` }
- enum `ib_col_type_t` {
  - `IB_VARCHAR` = 1, `IB_CHAR` = 2, `IB_BINARY` = 3, `IB_VARBINARY` = 4,
  - `IB_BLOB` = 5, `IB_INT` = 6, `IB_SYS` = 8, `IB_FLOAT` = 9,
  - `IB_DOUBLE` = 10, `IB_DECIMAL` = 11, `IB_VARCHAR_ANYCHARSET` = 12, `IB_CHAR_ANYCHARSET` = 13 }
- enum `ib_tbl_fmt_t` { `IB_TBL_REDUNDANT`, `IB_TBL_COMPACT`, `IB_TBL_DYNAMIC`, `IB_TBL_COMPRESSED` }
- enum `ib_col_attr_t` {
  - `IB_COL_NONE` = 0, `IB_COL_NOT_NULL` = 1, `IB_COL_UNSIGNED` = 2, `IB_COL_NOT_USED` = 4,
  - `IB_COL_CUSTOM1` = 8, `IB_COL_CUSTOM2` = 16, `IB_COL_CUSTOM3` = 32 }
- enum `ib_lck_mode_t` {
  - `IB_LOCK_IS` = 0, `IB_LOCK_IX`, `IB_LOCK_S`, `IB_LOCK_X`,
  - `IB_LOCK_NOT_USED`, `IB_LOCK_NONE`, `IB_LOCK_NUM` = `IB_LOCK_NONE` }
- enum `ib_srch_mode_t` { `IB_CUR_G` = 1, `IB_CUR_GE` = 2, `IB_CUR_L` = 3, `IB_CUR_LE` = 4 }
- enum `ib_match_mode_t` { `IB_CLOSEST_MATCH`, `IB_EXACT_MATCH`, `IB_EXACT_PREFIX` }
- enum `ib_trx_state_t` { `IB_TRX_NOT_STARTED`, `IB_TRX_ACTIVE`, `IB_TRX_COMMITTED_IN_MEMORY`, `IB_TRX_PREPARED` }
- enum `ib_trx_level_t` { `IB_TRX_READ_UNCOMMITTED` = 0, `IB_TRX_READ_COMMITTED` = 1, `IB_TRX_REPEATABLE_READ` = 2, `IB_TRX_SERIALIZABLE` = 3 }

- enum `ib_shutdown_t` { `IB_SHUTDOWN_NORMAL`, `IB_SHUTDOWN_NO_IBUFMERGE_PURGE`, `IB_SHUTDOWN_NO_BUFPOOL_FLUSH` }
- enum `ib_schema_visitor_version_t` { `IB_SCHEMA_VISITOR_TABLE = 1`, `IB_SCHEMA_VISITOR_TABLE_COL = 2`, `IB_SCHEMA_VISITOR_TABLE_AND_INDEX = 3`, `IB_SCHEMA_VISITOR_TABLE_AND_INDEX_COL = 4` }

## Functions

- HAILDB\_API `ib_u64_t ib_api_version` (void) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_init` (void) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_startup` (const char \*format) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_shutdown` (`ib_shutdown_t` flag) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_trx_start` (`ib_trx_t` `ib_trx`, `ib_trx_level_t` `ib_trx_level`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_trx_t ib_trx_begin` (`ib_trx_level_t` `ib_trx_level`) UNIV\_NO\_IGNORE
- HAILDB\_API void `ib_trx_set_client_data` (`ib_trx_t` `ib_trx`, void \*`client_data`)
- HAILDB\_API `ib_trx_state_t ib_trx_state` (`ib_trx_t` `ib_trx`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_trx_release` (`ib_trx_t` `ib_trx`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_trx_commit` (`ib_trx_t` `ib_trx`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_trx_rollback` (`ib_trx_t` `ib_trx`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_table_schema_add_col` (`ib_tbl_sch_t` `ib_tbl_sch`, const char \*`name`, `ib_col_type_t` `ib_col_type`, `ib_col_attr_t` `ib_col_attr`, `ib_u16_t` `client_type`, `ib_ulint_t` `len`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_table_schema_add_index` (`ib_tbl_sch_t` `ib_tbl_sch`, const char \*`name`, `ib_idx_sch_t` \*`ib_idx_sch`) UNIV\_NO\_IGNORE
- HAILDB\_API void `ib_table_schema_delete` (`ib_tbl_sch_t` `ib_tbl_sch`)
- HAILDB\_API `ib_err_t ib_table_schema_create` (const char \*`name`, `ib_tbl_sch_t` \*`ib_tbl_sch`, `ib_tbl_fmt_t` `ib_tbl_fmt`, `ib_ulint_t` `page_size`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_index_schema_add_col` (`ib_idx_sch_t` `ib_idx_sch`, const char \*`name`, `ib_ulint_t` `prefix_len`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_index_schema_create` (`ib_trx_t` `ib_usr_trx`, const char \*`name`, const char \*`table_name`, `ib_idx_sch_t` \*`ib_idx_sch`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_index_schema_set_clustered` (`ib_idx_sch_t` `ib_idx_sch`) UNIV\_NO\_IGNORE
- HAILDB\_API void `ib_cursor_set_simple_select` (`ib_crsr_t` `ib_crsr`)
- HAILDB\_API `ib_err_t ib_index_schema_set_unique` (`ib_idx_sch_t` `ib_idx_sch`) UNIV\_NO\_IGNORE
- HAILDB\_API void `ib_index_schema_delete` (`ib_idx_sch_t` `ib_idx_sch`)
- HAILDB\_API `ib_err_t ib_table_create` (`ib_trx_t` `ib_trx`, const `ib_tbl_sch_t` `ib_tbl_sch`, `ib_id_t` \*`id`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_table_rename` (`ib_trx_t` `ib_trx`, const char \*`old_name`, const char \*`new_name`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_index_create` (`ib_idx_sch_t` `ib_idx_sch`, `ib_id_t` \*`index_id`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_table_drop` (`ib_trx_t` `trx`, const char \*`name`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_index_drop` (`ib_trx_t` `trx`, `ib_id_t` `index_id`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_cursor_open_table_using_id` (`ib_id_t` `table_id`, `ib_trx_t` `ib_trx`, `ib_crsr_t` \*`ib_crsr`) UNIV\_NO\_IGNORE
- HAILDB\_API `ib_err_t ib_cursor_open_index_using_id` (`ib_id_t` `index_id`, `ib_trx_t` `ib_trx`, `ib_crsr_t` \*`ib_crsr`) UNIV\_NO\_IGNORE

- HAILDB\_API `ib_err_t ib_cursor_open_index_using_name (ib_csr_t ib_open_csr, const char *index_name, ib_csr_t *ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_open_table (const char *name, ib_trx_t ib_trx, ib_csr_t *ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_reset (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_close (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_insert_row (ib_csr_t ib_csr, const ib_tpl_t ib_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_update_row (ib_csr_t ib_csr, const ib_tpl_t ib_old_tpl, const ib_tpl_t ib_new_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_delete_row (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_read_row (ib_csr_t ib_csr, ib_tpl_t ib_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_prev (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_next (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_first (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_last (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_moveto (ib_csr_t ib_csr, ib_tpl_t ib_tpl, ib_srch_mode_t ib_srch_mode, int *result) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_cursor_attach_trx (ib_csr_t ib_csr, ib_trx_t ib_trx)`
- HAILDB\_API `void ib_set_client_compare (ib_client_cmp_t client_cmp_func)`
- HAILDB\_API `void ib_cursor_set_match_mode (ib_csr_t ib_csr, ib_match_mode_t match_mode)`
- HAILDB\_API `ib_err_t ib_col_set_value (ib_tpl_t ib_tpl, ib_ulint_t col_no, const void *src, ib_ulint_t len) UNIV_NO_IGNORE`
- HAILDB\_API `ib_ulint_t ib_col_get_len (ib_tpl_t ib_tpl, ib_ulint_t i) UNIV_NO_IGNORE`
- HAILDB\_API `ib_ulint_t ib_col_copy_value (ib_tpl_t ib_tpl, ib_ulint_t i, void *dst, ib_ulint_t len)`
- HAILDB\_API `ib_err_t ib_tuple_read_i8 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_i8_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_u8 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_u8_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_i16 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_i16_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_u16 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_u16_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_i32 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_i32_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_u32 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_u32_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_i64 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_i64_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_u64 (ib_tpl_t ib_tpl, ib_ulint_t i, ib_u64_t *ival) UNIV_NO_IGNORE`
- HAILDB\_API `const void * ib_col_get_value (ib_tpl_t ib_tpl, ib_ulint_t i) UNIV_NO_IGNORE`
- HAILDB\_API `ib_ulint_t ib_col_get_meta (ib_tpl_t ib_tpl, ib_ulint_t i, ib_col_meta_t *ib_col_meta)`
- HAILDB\_API `ib_tpl_t ib_tuple_clear (ib_tpl_t ib_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_get_cluster_key (ib_csr_t ib_csr, ib_tpl_t *ib_dst_tpl, const ib_tpl_t ib_src_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_copy (ib_tpl_t ib_dst_tpl, const ib_tpl_t ib_src_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_tpl_t ib_sec_search_tuple_create (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_tpl_t ib_sec_read_tuple_create (ib_csr_t ib_csr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_tpl_t ib_clust_search_tuple_create (ib_csr_t ib_csr) UNIV_NO_IGNORE`

- HAILDB\_API `ib_tpl_t ib_clust_read_tuple_create (ib_crsr_t ib_crsr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_ulint_t ib_tuple_get_n_user_cols (const ib_tpl_t ib_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `ib_ulint_t ib_tuple_get_n_cols (const ib_tpl_t ib_tpl) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_tuple_delete (ib_tpl_t ib_tpl)`
- HAILDB\_API `ib_err_t ib_cursor_truncate (ib_crsr_t *ib_crsr, ib_id_t *table_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_table_truncate (const char *table_name, ib_id_t *table_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_table_get_id (const char *table_name, ib_id_t *table_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_index_get_id (const char *table_name, const char *index_name, ib_id_t *index_id) UNIV_NO_IGNORE`
- HAILDB\_API `ib_bool_t ib_database_create (const char *dbname) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_database_drop (const char *dbname) UNIV_NO_IGNORE`
- HAILDB\_API `ib_bool_t ib_cursor_is_positioned (const ib_crsr_t ib_crsr) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_schema_lock_shared (ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_schema_lock_exclusive (ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_bool_t ib_schema_lock_is_exclusive (const ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_bool_t ib_schema_lock_is_shared (const ib_trx_t ib_trx) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_schema_unlock (ib_trx_t ib_trx)`
- HAILDB\_API `ib_err_t ib_cursor_lock (ib_crsr_t ib_crsr, ib_lck_mode_t ib_lck_mode) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_table_lock (ib_trx_t ib_trx, ib_id_t table_id, ib_lck_mode_t ib_lck_mode) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cursor_set_lock_mode (ib_crsr_t ib_crsr, ib_lck_mode_t ib_lck_mode) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_cursor_set_cluster_access (ib_crsr_t ib_crsr)`
- HAILDB\_API `ib_err_t ib_table_schema_visit (ib_trx_t ib_trx, const char *name, const ib_schema_visitor_t *visitor, void *arg) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_schema_tables_iterate (ib_trx_t ib_trx, ib_schema_visitor_table_all_t visitor, void *arg) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cfg_var_get_type (const char *name, ib_cfg_type_t *type) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cfg_set (const char *name,...) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cfg_get (const char *name, void *value) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_cfg_get_all (const char ***names, ib_u32_t *names_num) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_savepoint_take (ib_trx_t ib_trx, const void *name, ib_ulint_t name_len)`
- HAILDB\_API `ib_err_t ib_savepoint_release (ib_trx_t ib_trx, const void *name, ib_ulint_t name_len) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_savepoint_rollback (ib_trx_t ib_trx, const void *name, ib_ulint_t name_len) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_i8 (ib_tpl_t ib_tpl, int col_no, ib_i8_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_i16 (ib_tpl_t ib_tpl, int col_no, ib_i16_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_i32 (ib_tpl_t ib_tpl, int col_no, ib_i32_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_i64 (ib_tpl_t ib_tpl, int col_no, ib_i64_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_u8 (ib_tpl_t ib_tpl, int col_no, ib_u8_t val) UNIV_NO_IGNORE`



- HAILDB\_API `ib_err_t ib_tuple_write_u16 (ib_tpl_t ib_tpl, int col_no, ib_u16_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_u32 (ib_tpl_t ib_tpl, int col_no, ib_u32_t val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_u64 (ib_tpl_t ib_tpl, int col_no, ib_u64_t val) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_cursor_stmt_begin (ib_csr_t ib_csr)`
- HAILDB\_API `ib_err_t ib_tuple_write_double (ib_tpl_t ib_tpl, int col_no, double val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_double (ib_tpl_t ib_tpl, ib_ulint_t col_no, double *dval) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_write_float (ib_tpl_t ib_tpl, int col_no, float val) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_tuple_read_float (ib_tpl_t ib_tpl, ib_ulint_t col_no, float *fval) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_logger_set (ib_msg_log_t ib_msg_log, ib_msg_stream_t ib_msg_stream)`
- HAILDB\_API `const char * ib_strerror (ib_err_t db_errno) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_status_get_i64 (const char *name, ib_i64_t *dst) UNIV_NO_IGNORE`
- HAILDB\_API `ib_err_t ib_status_get_all (const char ***names, ib_u32_t *names_num) UNIV_NO_IGNORE`
- HAILDB\_API `void ib_set_panic_handler (ib_panic_handler_t new_panic_handler)`
- HAILDB\_API `void ib_set_trx_is_interrupted_handler (ib_trx_is_interrupted_handler_t handler)`
- HAILDB\_API `ib_err_t ib_get_duplicate_key (ib_trx_t ib_trx, const char **table_name, const char **index_name)`
- HAILDB\_API `ib_err_t ib_get_table_statistics (ib_csr_t ib_csr, ib_table_stats_t *table_stats, size_t sizeof_ib_table_stats_t)`
- HAILDB\_API `ib_err_t ib_get_index_stat_n_diff_key_vals (ib_csr_t ib_csr, const char *index_name, ib_u64_t *ncols, ib_i64_t **n_diff)`
- HAILDB\_API `ib_err_t ib_update_table_statistics (ib_csr_t csr)`
- HAILDB\_API `ib_err_t ib_error_inject (int error_to_inject)`

## Variables

- `ib_client_cmp_t ib_client_compare`

### 6.1.1 Detailed Description

<

### 6.1.2 Define Documentation

#### 6.1.2.1 #define IB\_FALSE 0x0UL

The boolean value of "false" used internally within InnoDB

#### 6.1.2.2 #define IB\_MAX\_COL\_NAME\_LEN (64 \* 3)

The maximum length of a column name in a table schema.

**6.1.2.3 #define IB\_MAX\_TABLE\_NAME\_LEN (64 \* 3)**

The maximum length of a table name (plus database name).

**6.1.2.4 #define IB\_N\_SYS\_COLS 3**

The number of system columns in a row.

**6.1.2.5 #define IB\_SQL\_NULL 0xFFFFFFFF**

Represents SQL\_NULL length

**6.1.2.6 #define ib\_tbl\_sch\_add\_blob\_col( s, n ) ib\_table\_schema\_add\_col(s, n, IB\_BLOB, IB\_COL\_NONE, 0, 0)**

Add a BLOB column to a table schema.

**Parameters**

*s* is the the schema handle

*n* is the column name

**6.1.2.7 #define ib\_tbl\_sch\_add\_text\_col( s, n ) ib\_table\_schema\_add\_col(s, n, IB\_VARCHAR, IB\_COL\_NONE, 0, MAX\_TEXT\_LEN)**

Add a BLOB column to a table schema.

**Parameters**

*s* is the the schema handle

*n* is the column name Add a TEXT column to a table schema.

**6.1.2.8 #define ib\_tbl\_sch\_add\_u32\_col( s, n ) ib\_table\_schema\_add\_col(s, n, IB\_INT, IB\_COL\_UNSIGNED, 0, 4)**

Add an UNSIGNED INT column to a table schema.

**Parameters**

*s* is the schema handle

*n* is the column name

**Returns**

DB\_SUCCESS or error code

**6.1.2.9 #define `ib_tbl_sch_add_u64_col( s, n )` `ib_table_schema_add_col(s, n, IB_INT, IB_COL_UNSIGNED, 0, 8)`**

Add an UNSIGNED BIGINT column to a table schema.

**Parameters**

*s* is the schema handle  
*n* is the column name

**Returns**

DB\_SUCCESS or error code

**6.1.2.10 #define `ib_tbl_sch_add_u64_notnull_col( s, n )`****Value:**

```
ib_table_schema_add_col( s, n, IB_INT, \
                          IB_COL_NOT_NULL | IB_COL_UNSIGNED, 0, \
                          8)
```

Add an UNSIGNED BIGINT NOT NULL column to a table schema.

**Parameters**

*s* is the schema handle  
*n* is the column name

**Returns**

DB\_SUCCESS or error code

**6.1.2.11 #define `ib_tbl_sch_add_varchar_col( s, n, l )` `ib_table_schema_add_col(s, n, IB_VARCHAR, IB_COL_NONE, 0, l)`**

Add a VARCHAR column to a table schema.

**Parameters**

*s* is the schema handle  
*n* is the column name  
*l* the max length of the VARCHAR column

**Returns**

DB\_SUCCESS or error code

**6.1.2.12 #define `IB_TRUE` 0x1UL**

The boolean value of "true" used internally within InnoDB

### 6.1.2.13 `#define MAX_TEXT_LEN 4096`

The maximum length of a text column.

### 6.1.2.14 `#define UNIV_NO_IGNORE`

Some HailDB methods will produce warnings if the result of them is not checked in client code. This uses a GCC compiler extension.

## 6.1.3 Typedef Documentation

### 6.1.3.1 `typedef ib_ulint_t ib_bool_t`

Representation of a "boolean" type within InnoDB

### 6.1.3.2 `typedef unsigned char ib_byte_t`

Representation of a byte within InnoDB

### 6.1.3.3 `typedef void(* ib_cb_t)(void)`

Generical InnoDB callback prototype.

### 6.1.3.4 `typedef ib_opaque_t ib_charset_t`

A character set pointer

### 6.1.3.5 `typedef int(* ib_client_cmp_t)(const ib_col_meta_t *col_meta, const ib_byte_t *p1, ib_ulint_t p1_len, const ib_byte_t *p2, ib_ulint_t p2_len)`

This function is used to compare two data fields for which the data type is such that we must use the client code to compare them. in: key length

### 6.1.3.6 `typedef struct ib_csr_struct* ib_csr_t`

InnoDB cursor handle

### 6.1.3.7 `typedef enum db_err ib_err_t`

All InnoDB error codes are represented by `ib_err_t`. See [db\\_err](#) for a complete list of possible error codes.

### 6.1.3.8 `typedef int16_t ib_i16_t`

A signed 16 bit integral type.

**6.1.3.9 typedef int32\_t ib\_i32\_t**

A signed 32 bit integral type.

**6.1.3.10 typedef int64\_t ib\_i64\_t**

A signed 64 bit integral type.

**6.1.3.11 typedef int8\_t ib\_i8\_t**

A signed 8 bit integral type.

**6.1.3.12 typedef ib\_u64\_t ib\_id\_t**

The integral type that represents internal table and index ids.

**6.1.3.13 typedef struct ib\_idx\_sch\_struct\* ib\_idx\_sch\_t**

InnoDB index schema handle

**6.1.3.14 typedef int(\* ib\_msg\_log\_t)(ib\_msg\_stream\_t, const char \*,...)**

All log messages are written to this function. It should have the same behavior as fprintf(3).

**6.1.3.15 typedef FILE\* ib\_msg\_stream\_t**

The first argument to the InnoDB message logging function. By default it's set to stderr. You should treat ib\_msg\_stream\_t as a void\*, since it will probably change in the future.

**6.1.3.16 typedef void\* ib\_opaque\_t**

Representation of a void\* within InnoDB

**6.1.3.17 typedef void(\* ib\_panic\_handler\_t)(void \*, int, char \*,...)**

Type of callback in the event of HailDB panicing. Your callback should call exit() rather soon, as continuing after a panic will lead to errors returned from every API function. We have also not fully tested every possible outcome from not immediately calling exit().

**6.1.3.18 typedef int(\* ib\_schema\_visitor\_index\_col\_t)(void \*arg, const char \*name, ib\_ulint\_t prefix\_len)**

Index column visitor Prefix length

**6.1.3.19** `typedef int(* ib_schema_visitor_index_t)(void *arg,const char *name,ib_bool_t clustered,ib_bool_t unique,int n_cols)`

Index visitor No. of cols defined

**6.1.3.20** `typedef int(* ib_schema_visitor_table_all_t)(void *arg,const char *name,int name_len)`

Visit all tables in the InnoDB schem. Length of name in bytes

**6.1.3.21** `typedef int(* ib_schema_visitor_table_col_t)(void *arg,const char *name,ib_col_type_t col_type,ib_ulint_t len,ib_col_attr_t attr)`

Table column visitor Column attributes

**6.1.3.22** `typedef int(* ib_schema_visitor_table_t)(void *arg,const char *name,ib_tbl_fmt_t tbl_fmt,ib_ulint_t page_size,int n_cols,int n_indexes)`

Table visitor No. of indexes defined

**6.1.3.23** `typedef struct ib_tbl_sch_struct* ib_tbl_sch_t`

InnoDB table schema handle

**6.1.3.24** `typedef struct ib_tpl_struct* ib_tpl_t`

InnoDB tuple handle. This handle can refer to either a cluster index tuple or a secondary index tuple. There are two types of tuples for each type of index, making a total of four types of tuple handles. There is a tuple for reading the entire row contents and another for searching on the index key.

**6.1.3.25** `typedef int(* ib_trx_is_interrupted_handler_t)(void *)`

Callback for checking if a transaction has been interrupted. This callback lets you implement the MySQL KILL command kind of functionality. A transaction may block in the thread it's running in (for example, while acquiring row locks or doing IO) but other threads may do something that causes `ib_trx_is_interrupted_handler_t` to return true.

**6.1.3.26** `typedef struct ib_trx_struct* ib_trx_t`

InnoDB transaction handle, all database operations need to be covered by transactions. This handle represents a transaction. The handle can be created with `ib_trx_begin()`, you commit your changes with `ib_trx_commit()` and undo your changes using `ib_trx_rollback()`. If the InnoDB deadlock monitor rolls back the transaction then you need to free the transaction using the function `ib_trx_release()`. You can query the state of an InnoDB transaction by calling `ib_trx_state()`.

**6.1.3.27** `typedef uint16_t ib_u16_t`

An unsigned 16 bit integral type.

### 6.1.3.28 typedef uint32\_t ib\_u32\_t

An unsigned 32 bit integral type.

### 6.1.3.29 typedef uint64\_t ib\_u64\_t

An unsigned 64 bit integral type.

### 6.1.3.30 typedef uint8\_t ib\_u8\_t

An unsigned 8 bit integral type.

### 6.1.3.31 typedef unsigned long int ib\_ulint\_t

Representation of an unsigned long int within InnoDB

## 6.1.4 Enumeration Type Documentation

### 6.1.4.1 enum db\_err

InnoDB error codes. Most of the error codes are internal to the engine and will not be seen by user applications. The partial error codes reflect the sub-state of an operation within InnoDB. Some of the error codes are deprecated and are no longer used.

#### Enumerator:

***DB\_SUCCESS*** A successful result

***DB\_ERROR*** This is a generic error code. It is used to classify error conditions that can't be represented by other codes

***DB\_INTERRUPTED*** An operation was interrupted by a user.

***DB\_OUT\_OF\_MEMORY*** Operation caused an out of memory error. Within InnoDB core code this is normally a fatal error

***DB\_OUT\_OF\_FILE\_SPACE*** The operating system returned an out of file space error when trying to do an IO operation.

***DB\_LOCK\_WAIT*** A lock request by transaction resulted in a lock wait. The thread is suspended internally by InnoDB and is put on a lock wait queue.

***DB\_DEADLOCK*** A lock request by a transaction resulted in a deadlock. The transaction was rolled back

***DB\_ROLLBACK*** Not used

***DB\_DUPLICATE\_KEY*** A record insert or update violates a unique constraint.

***DB\_QUE\_THR\_SUSPENDED*** A query thread should be in state suspended but is trying to acquire a lock. Currently this is treated as a hard error and a violation of an invariant.

***DB\_MISSING\_HISTORY*** Required history data has been deleted due to lack of space in rollback segment

***DB\_CLUSTER\_NOT\_FOUND*** This error is not used

***DB\_TABLE\_NOT\_FOUND*** The table could not be found

**DB\_MUST\_GET\_MORE\_FILE\_SPACE** The database has to be stopped and restarted with more file space

**DB\_TABLE\_IS\_BEING\_USED** The user is trying to create a table in the InnoDB data dictionary but a table with that name already exists

**DB\_TOO\_BIG\_RECORD** A record in an index would not fit on a compressed page, or it would become bigger than 1/2 free space in an uncompressed page frame

**DB\_LOCK\_WAIT\_TIMEOUT** Lock wait lasted too long

**DB\_NO\_REFERENCED\_ROW** Referenced key value not found for a foreign key in an insert or update of a row

**DB\_ROW\_IS\_REFERENCED** Cannot delete or update a row because it contains a key value which is referenced

**DB\_CANNOT\_ADD\_CONSTRAINT** Adding a foreign key constraint to a table failed

**DB\_CORRUPTION** Data structure corruption noticed

**DB\_COL\_APPEARS\_TWICE\_IN\_INDEX** InnoDB cannot handle an index where same column appears twice

**DB\_CANNOT\_DROP\_CONSTRAINT** Dropping a foreign key constraint from a table failed

**DB\_NO\_SAVEPOINT** No savepoint exists with the given name

**DB\_TABLESPACE\_ALREADY\_EXISTS** We cannot create a new single-table tablespace because a file of the same name already exists

**DB\_TABLESPACE\_DELETED** Tablespace does not exist or is being dropped right now

**DB\_LOCK\_TABLE\_FULL** Lock structs have exhausted the buffer pool (for big transactions, InnoDB stores the lock structs in the buffer pool)

**DB\_FOREIGN\_DUPLICATE\_KEY** Foreign key constraints activated but the operation would lead to a duplicate key in some table

**DB\_TOO\_MANY\_CONCURRENT\_TRXS** When InnoDB runs out of the preconfigured undo slots, this can only happen when there are too many concurrent transactions

**DB\_UNSUPPORTED** When InnoDB sees any artefact or a feature that it can't recognize or work with e.g., FT indexes created by a later version of the engine.

**DB\_PRIMARY\_KEY\_IS\_NULL** A column in the PRIMARY KEY was found to be NULL

**DB\_FATAL** The application should clean up and quite ASAP. Fatal error, InnoDB cannot continue operation without risking database corruption.

**DB\_FAIL** Partial failure code.

**DB\_OVERFLOW** If an update or insert of a record doesn't fit in a Btree page

**DB\_UNDERFLOW** If an update or delete of a record causes a Btree page to be below a minimum threshold

**DB\_STRONG\_FAIL** Failure to insert a secondary index entry to the insert buffer

**DB\_ZIP\_OVERFLOW** Failure trying to compress a page

**DB\_RECORD\_NOT\_FOUND** Record not found

**DB\_END\_OF\_INDEX** A cursor operation or search operation scanned to the end of the index.

**DB\_SCHEMA\_ERROR** Generic schema error

**DB\_DATA\_MISMATCH** Column update or read failed because the types mismatch

**DB\_SCHEMA\_NOT\_LOCKED** If an API function expects the schema to be locked in exclusive mode and if it's not then that API function will return this error code

**DB\_NOT\_FOUND** Generic error code for "Not found" type of errors

**DB\_READONLY** Generic error code for "Readonly" type of errors

**DB\_INVALID\_INPUT** Generic error code for "Invalid input" type of errors



#### 6.1.4.2 enum ib\_cfg\_type\_t

Possible types for a configuration variable.

**Enumerator:**

- IB\_CFG\_IBOOL* The configuration parameter is of type ibool
- IB\_CFG\_ULINT* The configuration parameter is of type ulint
- IB\_CFG\_ULONG* The configuration parameter is of type ulong
- IB\_CFG\_TEXT* The configuration parameter is of type char\*
- IB\_CFG\_CB* The configuration parameter is a callback parameter

#### 6.1.4.3 enum ib\_col\_attr\_t

InnoDB column attributes

**Enumerator:**

- IB\_COL\_NONE* No special attributes.
- IB\_COL\_NOT\_NULL* Column data can't be NULL.
- IB\_COL\_UNSIGNED* Column is IB\_INT and unsigned.
- IB\_COL\_NOT\_USED* Future use, reserved.
- IB\_COL\_CUSTOM1* Custom precision type, this is a bit that is ignored by InnoDB and so can be set and queried by users.
- IB\_COL\_CUSTOM2* Custom precision type, this is a bit that is ignored by InnoDB and so can be set and queried by users.
- IB\_COL\_CUSTOM3* Custom precision type, this is a bit that is ignored by InnoDB and so can be set and queried by users.

#### 6.1.4.4 enum ib\_col\_type\_t

column types that are supported.

**Enumerator:**

- IB\_VARCHAR* Character varying length. The column is not padded.
- IB\_CHAR* Fixed length character string. The column is padded to the right.
- IB\_BINARY* Fixed length binary, similar to IB\_CHAR but the column is not padded to the right.
- IB\_VARBINARY* Variable length binary
- IB\_BLOB* Binary large object, or a TEXT type
- IB\_INT* Integer: can be any size from 1 - 8 bytes. If the size is 1, 2, 4 and 8 bytes then you can use the typed read and write functions. For other sizes you will need to use the [ib\\_col\\_get\\_value\(\)](#) function and do the conversion yourself.
- IB\_SYS* System column, this column can be one of DATA\_TRX\_ID, DATA\_ROLL\_PTR or DATA\_ROW\_ID.
- IB\_FLOAT* C (float) floating point value.
- IB\_DECIMAL* > C (double) floating point value. Decimal stored as an ASCII string
- IB\_VARCHAR\_ANYCHARSET* Any charset, varying length
- IB\_CHAR\_ANYCHARSET* Any charset, fixed length

#### 6.1.4.5 enum `ib_lck_mode_t`

InnoDB lock modes.

##### Enumerator:

- IB\_LOCK\_IS* Intention shared, an intention lock should be used to lock tables
- IB\_LOCK\_IX* Intention exclusive, an intention lock should be used to lock tables
- IB\_LOCK\_S* Shared locks should be used to lock rows
- IB\_LOCK\_X* Exclusive locks should be used to lock rows
- IB\_LOCK\_NOT\_USED* Future use, reserved
- IB\_LOCK\_NONE* This is used internally to note consistent read
- IB\_LOCK\_NUM* number of lock modes

#### 6.1.4.6 enum `ib_match_mode_t`

Various match modes used by `ib_cursor_moveto()`

##### Enumerator:

- IB\_CLOSEST\_MATCH* Closest match possible
- IB\_EXACT\_MATCH* Search using a complete key value
- IB\_EXACT\_PREFIX* Search using a key prefix which must match to rows: the prefix may contain an incomplete field (the last field in prefix may be just a prefix of a fixed length column)

#### 6.1.4.7 enum `ib_schema_visitor_version_t`

Currently, this is also the number of callback functions in the struct.

#### 6.1.4.8 enum `ib_shutdown_t`

When `ib_shutdown()` is called InnoDB may take a long time to shutdown because of background tasks e.g., purging deleted records. The following flags allow the user to control the shutdown behavior.

##### Enumerator:

- IB\_SHUTDOWN\_NORMAL* Normal shutdown, do insert buffer merge and purge before complete shutdown.
- IB\_SHUTDOWN\_NO\_IBUFMERGE\_PURGE* Do not do a purge and index buffer merge at shutdown.
- IB\_SHUTDOWN\_NO\_BUFPOOL\_FLUSH* Same as *NO\_IBUFMERGE\_PURGE* and in addition do not even flush the buffer pool to data files. No committed transactions are lost

#### 6.1.4.9 enum `ib_srch_mode_t`

InnoDB cursor search modes for `ib_cursor_moveto()`. Note: Values must match those found in `page0cur.h`

**Enumerator:**

- IB\_CUR\_G* If search key is not found then position the cursor on the row that is greater than the search key
- IB\_CUR\_GE* If the search key not found then position the cursor on the row that is greater than or equal to the search key
- IB\_CUR\_L* If search key is not found then position the cursor on the row that is less than the search key
- IB\_CUR\_LE* If search key is not found then position the cursor on the row that is less than or equal to the search key

**6.1.4.10 enum ib\_tbl\_fmt\_t**

InnoDB table format types

**Enumerator:**

- IB\_TBL\_REDUNDANT* Redundant row format, the column type and length is stored in the row.
- IB\_TBL\_COMPACT* Compact row format, the column type is not stored in the row. The length is stored in the row but the storage format uses a compact format to store the length of the column data and record data storage format also uses less storage.
- IB\_TBL\_DYNAMIC* Compact row format. BLOB prefixes are not stored in the clustered index
- IB\_TBL\_COMPRESSED* Similar to dynamic format but with pages compressed

**6.1.4.11 enum ib\_trx\_level\_t**

Transaction isolation levels

**Enumerator:**

- IB\_TRX\_READ\_UNCOMMITTED* Dirty read: non-locking SELECTs are performed so that we do not look at a possible earlier version of a record; thus they are not 'consistent' reads under this isolation level; otherwise like level 2
- IB\_TRX\_READ\_COMMITTED* Somewhat Oracle-like isolation, except that in range UPDATE and DELETE we must block phantom rows with next-key locks; SELECT ... FOR UPDATE and ... LOCK IN SHARE MODE only lock the index records, NOT the gaps before them, and thus allow free inserting; each consistent read reads its own snapshot
- IB\_TRX\_REPEATABLE\_READ* All consistent reads in the same trx read the same snapshot; full next-key locking used in locking reads to block insertions into gaps
- IB\_TRX\_SERIALIZABLE* All plain SELECTs are converted to LOCK IN SHARE MODE reads

**6.1.4.12 enum ib\_trx\_state\_t**

The transaction state can be queried using the [ib\\_trx\\_state\(\)](#) function. The InnoDB deadlock monitor can roll back a transaction and users should be prepared for this, especially where there is high contention. The way to determine the state of the transaction is to query it's state and check.

**Enumerator:**

- IB\_TRX\_NOT\_STARTED* Has not started yet, the transaction has not ben started yet.

***IB\_TRX\_ACTIVE*** The transaction is currently active and needs to be either committed or rolled back.

***IB\_TRX\_COMMITTED\_IN\_MEMORY*** Not committed to disk yet

***IB\_TRX\_PREPARED*** Support for 2PC/XA

## 6.1.5 Function Documentation

### 6.1.5.1 HAILDB\_API `ib_bool_t ib_schema_lock_is_shared ( const ib_trx_t ib_trx )`

Checks if the data dictionary is latched in shared mode.

#### Parameters

*ib\_trx* is a transaction instance

#### Returns

TRUE if shared latch

## 6.1.6 Variable Documentation

### 6.1.6.1 `ib_client_cmp_t ib_client_compare`

Callback function to compare InnoDB key columns in an index.

# Index

- attr
  - [ib\\_col\\_meta\\_t](#), 45
- charset
  - [ib\\_col\\_meta\\_t](#), 45
- client\_type
  - [ib\\_col\\_meta\\_t](#), 45
- config
  - [ib\\_cfg\\_get](#), 19
  - [ib\\_cfg\\_get\\_all](#), 19
  - [ib\\_cfg\\_set](#), 19
  - [ib\\_cfg\\_set\\_bool\\_off](#), 17
  - [ib\\_cfg\\_set\\_bool\\_on](#), 17
  - [ib\\_cfg\\_set\\_callback](#), 18
  - [ib\\_cfg\\_set\\_int](#), 18
  - [ib\\_cfg\\_set\\_text](#), 18
  - [ib\\_cfg\\_var\\_get\\_type](#), 19
  - [ib\\_set\\_panic\\_handler](#), 20
  - [ib\\_set\\_trx\\_is\\_interrupted\\_handler](#), 20
  - [ib\\_status\\_get\\_all](#), 20
- Configuration functions, 17
- cursor
  - [ib\\_cursor\\_attach\\_trx](#), 8
  - [ib\\_cursor\\_close](#), 8
  - [ib\\_cursor\\_first](#), 9
  - [ib\\_cursor\\_is\\_positioned](#), 9
  - [ib\\_cursor\\_last](#), 9
  - [ib\\_cursor\\_moveto](#), 9
  - [ib\\_cursor\\_next](#), 10
  - [ib\\_cursor\\_open\\_index\\_using\\_id](#), 10
  - [ib\\_cursor\\_open\\_index\\_using\\_name](#), 10
  - [ib\\_cursor\\_open\\_table](#), 11
  - [ib\\_cursor\\_open\\_table\\_using\\_id](#), 11
  - [ib\\_cursor\\_prev](#), 11
  - [ib\\_cursor\\_reset](#), 11
  - [ib\\_cursor\\_set\\_match\\_mode](#), 12
  - [ib\\_cursor\\_stmt\\_begin](#), 12
- Cursor functions, 8
- DB\_CANNOT\_ADD\_CONSTRAINT
  - [haiddb.h](#), 62
- DB\_CANNOT\_DROP\_CONSTRAINT
  - [haiddb.h](#), 62
- DB\_CLUSTER\_NOT\_FOUND
  - [haiddb.h](#), 61
- DB\_COL\_APPEARS\_TWICE\_IN\_INDEX
  - [haiddb.h](#), 62
- DB\_CORRUPTION
  - [haiddb.h](#), 62
- DB\_DATA\_MISMATCH
  - [haiddb.h](#), 62
- DB\_DEADLOCK
  - [haiddb.h](#), 61
- DB\_DUPLICATE\_KEY
  - [haiddb.h](#), 61
- DB\_END\_OF\_INDEX
  - [haiddb.h](#), 62
- DB\_ERROR
  - [haiddb.h](#), 61
- DB\_FAIL
  - [haiddb.h](#), 62
- DB\_FATAL
  - [haiddb.h](#), 62
- DB\_FOREIGN\_DUPLICATE\_KEY
  - [haiddb.h](#), 62
- DB\_INTERRUPTED
  - [haiddb.h](#), 61
- DB\_INVALID\_INPUT
  - [haiddb.h](#), 62
- DB\_LOCK\_TABLE\_FULL
  - [haiddb.h](#), 62
- DB\_LOCK\_WAIT
  - [haiddb.h](#), 61
- DB\_LOCK\_WAIT\_TIMEOUT
  - [haiddb.h](#), 62
- DB\_MISSING\_HISTORY
  - [haiddb.h](#), 61
- DB\_MUST\_GET\_MORE\_FILE\_SPACE
  - [haiddb.h](#), 61
- DB\_NO\_REFERENCED\_ROW
  - [haiddb.h](#), 62
- DB\_NO\_SAVEPOINT
  - [haiddb.h](#), 62
- DB\_NOT\_FOUND
  - [haiddb.h](#), 62
- DB\_OUT\_OF\_FILE\_SPACE
  - [haiddb.h](#), 61
- DB\_OUT\_OF\_MEMORY
  - [haiddb.h](#), 61
- DB\_OVERFLOW
  - [haiddb.h](#), 61

- haildb.h, 62
- DB\_PRIMARY\_KEY\_IS\_NULL
  - haildb.h, 62
- DB\_QUE\_THR\_SUSPENDED
  - haildb.h, 61
- DB\_READONLY
  - haildb.h, 62
- DB\_RECORD\_NOT\_FOUND
  - haildb.h, 62
- DB\_ROLLBACK
  - haildb.h, 61
- DB\_ROW\_IS\_REFERENCED
  - haildb.h, 62
- DB\_SCHEMA\_ERROR
  - haildb.h, 62
- DB\_SCHEMA\_NOT\_LOCKED
  - haildb.h, 62
- DB\_STRONG\_FAIL
  - haildb.h, 62
- DB\_SUCCESS
  - haildb.h, 61
- DB\_TABLE\_IS\_BEING\_USED
  - haildb.h, 62
- DB\_TABLE\_NOT\_FOUND
  - haildb.h, 61
- DB\_TABLESPACE\_ALREADY\_EXISTS
  - haildb.h, 62
- DB\_TABLESPACE\_DELETED
  - haildb.h, 62
- DB\_TOO\_BIG\_RECORD
  - haildb.h, 62
- DB\_TOO\_MANY\_CONCURRENT\_TRXS
  - haildb.h, 62
- DB\_UNDERFLOW
  - haildb.h, 62
- DB\_UNSUPPORTED
  - haildb.h, 62
- DB\_ZIP\_OVERFLOW
  - haildb.h, 62
- db\_err
  - haildb.h, 61
- ddl
  - ib\_cursor\_truncate, 21
  - ib\_database\_create, 22
  - ib\_database\_drop, 22
  - ib\_index\_create, 22
  - ib\_index\_drop, 22
  - ib\_index\_get\_id, 22
  - ib\_index\_schema\_add\_col, 23
  - ib\_index\_schema\_create, 23
  - ib\_index\_schema\_delete, 23
  - ib\_index\_schema\_set\_clustered, 23
  - ib\_index\_schema\_set\_unique, 24
  - ib\_schema\_lock\_exclusive, 24
  - ib\_schema\_lock\_is\_exclusive, 24
  - ib\_schema\_lock\_shared, 24
  - ib\_schema\_tables\_iterate, 25
  - ib\_schema\_unlock, 25
  - ib\_table\_create, 25
  - ib\_table\_drop, 26
  - ib\_table\_get\_id, 26
  - ib\_table\_rename, 26
  - ib\_table\_schema\_add\_col, 26
  - ib\_table\_schema\_add\_index, 27
  - ib\_table\_schema\_create, 27
  - ib\_table\_schema\_delete, 27
  - ib\_table\_schema\_visit, 28
  - ib\_table\_truncate, 28
- DDL functions, 20
- debug
  - ib\_error\_inject, 44
- Debug and Testing functions, 44
- dml
  - ib\_col\_copy\_value, 35
  - ib\_col\_get\_len, 35
  - ib\_col\_get\_meta, 36
  - ib\_col\_get\_value, 36
  - ib\_col\_set\_value, 36
  - ib\_cursor\_delete\_row, 37
  - ib\_cursor\_insert\_row, 37
  - ib\_cursor\_read\_row, 37
  - ib\_cursor\_set\_cluster\_access, 37
  - ib\_cursor\_update\_row, 37
  - ib\_tuple\_read\_double, 38
  - ib\_tuple\_read\_float, 38
  - ib\_tuple\_read\_i16, 38
  - ib\_tuple\_read\_i32, 39
  - ib\_tuple\_read\_i64, 39
  - ib\_tuple\_read\_i8, 39
  - ib\_tuple\_read\_u16, 39
  - ib\_tuple\_read\_u32, 40
  - ib\_tuple\_read\_u64, 40
  - ib\_tuple\_read\_u8, 40
  - ib\_tuple\_write\_double, 41
  - ib\_tuple\_write\_float, 41
  - ib\_tuple\_write\_i16, 41
  - ib\_tuple\_write\_i32, 41
  - ib\_tuple\_write\_i64, 42
  - ib\_tuple\_write\_i8, 42
  - ib\_tuple\_write\_u16, 42
  - ib\_tuple\_write\_u32, 43
  - ib\_tuple\_write\_u64, 43
  - ib\_tuple\_write\_u8, 43
- DML functions, 34
- haildb.h, 49
  - DB\_CANNOT\_ADD\_CONSTRAINT, 62
  - DB\_CANNOT\_DROP\_CONSTRAINT, 62

- DB\_CLUSTER\_NOT\_FOUND, 61
- DB\_COL\_APPEARS\_TWICE\_IN\_INDEX, 62
- DB\_CORRUPTION, 62
- DB\_DATA\_MISMATCH, 62
- DB\_DEADLOCK, 61
- DB\_DUPLICATE\_KEY, 61
- DB\_END\_OF\_INDEX, 62
- DB\_ERROR, 61
- DB\_FAIL, 62
- DB\_FATAL, 62
- DB\_FOREIGN\_DUPLICATE\_KEY, 62
- DB\_INTERRUPTED, 61
- DB\_INVALID\_INPUT, 62
- DB\_LOCK\_TABLE\_FULL, 62
- DB\_LOCK\_WAIT, 61
- DB\_LOCK\_WAIT\_TIMEOUT, 62
- DB\_MISSING\_HISTORY, 61
- DB\_MUST\_GET\_MORE\_FILE\_SPACE, 61
- DB\_NO\_REFERENCED\_ROW, 62
- DB\_NO\_SAVEPOINT, 62
- DB\_NOT\_FOUND, 62
- DB\_OUT\_OF\_FILE\_SPACE, 61
- DB\_OUT\_OF\_MEMORY, 61
- DB\_OVERFLOW, 62
- DB\_PRIMARY\_KEY\_IS\_NULL, 62
- DB\_QUEUE\_THR\_SUSPENDED, 61
- DB\_READONLY, 62
- DB\_RECORD\_NOT\_FOUND, 62
- DB\_ROLLBACK, 61
- DB\_ROW\_IS\_REFERENCED, 62
- DB\_SCHEMA\_ERROR, 62
- DB\_SCHEMA\_NOT\_LOCKED, 62
- DB\_STRONG\_FAIL, 62
- DB\_SUCCESS, 61
- DB\_TABLE\_IS\_BEING\_USED, 62
- DB\_TABLE\_NOT\_FOUND, 61
- DB\_TABLESPACE\_ALREADY\_EXISTS, 62
- DB\_TABLESPACE\_DELETED, 62
- DB\_TOO\_BIG\_RECORD, 62
- DB\_TOO\_MANY\_CONCURRENT\_TRXS, 62
- DB\_UNDERFLOW, 62
- DB\_UNSUPPORTED, 62
- DB\_ZIP\_OVERFLOW, 62
- db\_err, 61
- IB\_BINARY, 63
- IB\_BLOB, 63
- IB\_CFG\_CB, 63
- IB\_CFG\_IBOOL, 63
- IB\_CFG\_TEXT, 63
- IB\_CFG\_ULINT, 63
- IB\_CFG\_ULONG, 63
- IB\_CHAR, 63
- IB\_CHAR\_ANYCHARSET, 63
- IB\_CLOSEST\_MATCH, 64
- IB\_COL\_CUSTOM1, 63
- IB\_COL\_CUSTOM2, 63
- IB\_COL\_CUSTOM3, 63
- IB\_COL\_NONE, 63
- IB\_COL\_NOT\_NULL, 63
- IB\_COL\_NOT\_USED, 63
- IB\_COL\_UNSIGNED, 63
- IB\_CUR\_G, 65
- IB\_CUR\_GE, 65
- IB\_CUR\_L, 65
- IB\_CUR\_LE, 65
- IB\_DECIMAL, 63
- IB\_EXACT\_MATCH, 64
- IB\_EXACT\_PREFIX, 64
- IB\_FLOAT, 63
- IB\_INT, 63
- IB\_LOCK\_IS, 64
- IB\_LOCK\_IX, 64
- IB\_LOCK\_NONE, 64
- IB\_LOCK\_NOT\_USED, 64
- IB\_LOCK\_NUM, 64
- IB\_LOCK\_S, 64
- IB\_LOCK\_X, 64
- IB\_SHUTDOWN\_NO\_BUFPOOL\_FLUSH, 64
- IB\_SHUTDOWN\_NO\_IBUFMERGE\_PURGE, 64
- IB\_SHUTDOWN\_NORMAL, 64
- IB\_SYS, 63
- IB\_TBL\_COMPACT, 65
- IB\_TBL\_COMPRESSED, 65
- IB\_TBL\_DYNAMIC, 65
- IB\_TBL\_REDUNDANT, 65
- IB\_TRX\_ACTIVE, 65
- IB\_TRX\_COMMITTED\_IN\_MEMORY, 66
- IB\_TRX\_NOT\_STARTED, 65
- IB\_TRX\_PREPARED, 66
- IB\_TRX\_READ\_COMMITTED, 65
- IB\_TRX\_READ\_UNCOMMITTED, 65
- IB\_TRX\_REPEATABLE\_READ, 65
- IB\_TRX\_SERIALIZABLE, 65
- IB\_VARBINARY, 63
- IB\_VARCHAR, 63
- IB\_VARCHAR\_ANYCHARSET, 63
- ib\_bool\_t, 58
- ib\_byte\_t, 58
- ib\_cb\_t, 58
- ib\_cfg\_type\_t, 62
- ib\_charset\_t, 58
- ib\_client\_cmp\_t, 58
- ib\_client\_compare, 66
- ib\_col\_attr\_t, 63

- ib\_col\_type\_t, 63
- ib\_crsr\_t, 58
- ib\_err\_t, 58
- IB\_FALSE, 55
- ib\_i16\_t, 58
- ib\_i32\_t, 58
- ib\_i64\_t, 59
- ib\_i8\_t, 59
- ib\_id\_t, 59
- ib\_idx\_sch\_t, 59
- ib\_lck\_mode\_t, 63
- ib\_match\_mode\_t, 64
- IB\_MAX\_COL\_NAME\_LEN, 55
- IB\_MAX\_TABLE\_NAME\_LEN, 55
- ib\_msg\_log\_t, 59
- ib\_msg\_stream\_t, 59
- IB\_N\_SYS\_COLS, 56
- ib\_opaque\_t, 59
- ib\_panic\_handler\_t, 59
- ib\_schema\_lock\_is\_shared, 66
- ib\_schema\_visitor\_index\_col\_t, 59
- ib\_schema\_visitor\_index\_t, 59
- ib\_schema\_visitor\_table\_all\_t, 60
- ib\_schema\_visitor\_table\_col\_t, 60
- ib\_schema\_visitor\_table\_t, 60
- ib\_schema\_visitor\_version\_t, 64
- ib\_shutdown\_t, 64
- IB\_SQL\_NULL, 56
- ib\_srch\_mode\_t, 64
- ib\_tbl\_fmt\_t, 65
- ib\_tbl\_sch\_add\_blob\_col, 56
- ib\_tbl\_sch\_add\_text\_col, 56
- ib\_tbl\_sch\_add\_u32\_col, 56
- ib\_tbl\_sch\_add\_u64\_col, 56
- ib\_tbl\_sch\_add\_u64\_notnull\_col, 57
- ib\_tbl\_sch\_add\_varchar\_col, 57
- ib\_tbl\_sch\_t, 60
- ib\_tpl\_t, 60
- IB\_TRUE, 57
- ib\_trx\_is\_interrupted\_handler\_t, 60
- ib\_trx\_level\_t, 65
- ib\_trx\_state\_t, 65
- ib\_trx\_t, 60
- ib\_u16\_t, 60
- ib\_u32\_t, 60
- ib\_u64\_t, 61
- ib\_u8\_t, 61
- ib\_ulint\_t, 61
- MAX\_TEXT\_LEN, 57
- UNIV\_NO\_IGNORE, 58
- hailldb.h, 63
- IB\_CFG\_CB
  - hailldb.h, 63
- IB\_CFG\_IBOOL
  - hailldb.h, 63
- IB\_CFG\_TEXT
  - hailldb.h, 63
- IB\_CFG\_ULINT
  - hailldb.h, 63
- IB\_CFG\_ULONG
  - hailldb.h, 63
- IB\_CHAR
  - hailldb.h, 63
- IB\_CHAR\_ANYCHARSET
  - hailldb.h, 63
- IB\_CLOSEST\_MATCH
  - hailldb.h, 64
- IB\_COL\_CUSTOM1
  - hailldb.h, 63
- IB\_COL\_CUSTOM2
  - hailldb.h, 63
- IB\_COL\_CUSTOM3
  - hailldb.h, 63
- IB\_COL\_NONE
  - hailldb.h, 63
- IB\_COL\_NOT\_NULL
  - hailldb.h, 63
- IB\_COL\_NOT\_USED
  - hailldb.h, 63
- IB\_COL\_UNSIGNED
  - hailldb.h, 63
- IB\_CUR\_G
  - hailldb.h, 65
- IB\_CUR\_GE
  - hailldb.h, 65
- IB\_CUR\_L
  - hailldb.h, 65
- IB\_CUR\_LE
  - hailldb.h, 65
- IB\_DECIMAL
  - hailldb.h, 63
- IB\_EXACT\_MATCH
  - hailldb.h, 64
- IB\_EXACT\_PREFIX
  - hailldb.h, 64
- IB\_FLOAT
  - hailldb.h, 63
- IB\_INT
  - hailldb.h, 63
- IB\_LOCK\_IS
  - hailldb.h, 64
- IB\_LOCK\_IX
  - hailldb.h, 64
- IB\_LOCK\_NONE
- IB\_BINARY
  - hailldb.h, 63
- IB\_BLOB



- haildb.h, 64
- IB\_LOCK\_NOT\_USED
  - haildb.h, 64
- IB\_LOCK\_NUM
  - haildb.h, 64
- IB\_LOCK\_S
  - haildb.h, 64
- IB\_LOCK\_X
  - haildb.h, 64
- IB\_SHUTDOWN\_NO\_BUFPOOL\_FLUSH
  - haildb.h, 64
- IB\_SHUTDOWN\_NO\_IBUFMERGE\_PURGE
  - haildb.h, 64
- IB\_SHUTDOWN\_NORMAL
  - haildb.h, 64
- IB\_SYS
  - haildb.h, 63
- IB\_TBL\_COMPACT
  - haildb.h, 65
- IB\_TBL\_COMPRESSED
  - haildb.h, 65
- IB\_TBL\_DYNAMIC
  - haildb.h, 65
- IB\_TBL\_REDUNDANT
  - haildb.h, 65
- IB\_TRX\_ACTIVE
  - haildb.h, 65
- IB\_TRX\_COMMITTED\_IN\_MEMORY
  - haildb.h, 66
- IB\_TRX\_NOT\_STARTED
  - haildb.h, 65
- IB\_TRX\_PREPARED
  - haildb.h, 66
- IB\_TRX\_READ\_COMMITTED
  - haildb.h, 65
- IB\_TRX\_READ\_UNCOMMITTED
  - haildb.h, 65
- IB\_TRX\_REPEATABLE\_READ
  - haildb.h, 65
- IB\_TRX\_SERIALIZABLE
  - haildb.h, 65
- IB\_VARBINARY
  - haildb.h, 63
- IB\_VARCHAR
  - haildb.h, 63
- IB\_VARCHAR\_ANYCHARSET
  - haildb.h, 63
- ib\_api\_version
  - misc, 29
- ib\_bool\_t
  - haildb.h, 58
- ib\_byte\_t
  - haildb.h, 58
- ib\_cb\_t
  - haildb.h, 58
- ib\_cfg\_get
  - config, 19
- ib\_cfg\_get\_all
  - config, 19
- ib\_cfg\_set
  - config, 19
- ib\_cfg\_set\_bool\_off
  - config, 17
- ib\_cfg\_set\_bool\_on
  - config, 17
- ib\_cfg\_set\_callback
  - config, 18
- ib\_cfg\_set\_int
  - config, 18
- ib\_cfg\_set\_text
  - config, 18
- ib\_cfg\_type\_t
  - haildb.h, 62
- ib\_cfg\_var\_get\_type
  - config, 19
- ib\_charset\_t
  - haildb.h, 58
- ib\_client\_cmp\_t
  - haildb.h, 58
- ib\_client\_compare
  - haildb.h, 66
- ib\_clust\_read\_tuple\_create
  - tuple, 31
- ib\_clust\_search\_tuple\_create
  - tuple, 32
- ib\_col\_attr\_t
  - haildb.h, 63
- ib\_col\_copy\_value
  - dml, 35
- ib\_col\_get\_len
  - dml, 35
- ib\_col\_get\_meta
  - dml, 36
- ib\_col\_get\_value
  - dml, 36
- ib\_col\_meta\_t, 45
  - attr, 45
  - charset, 45
  - client\_type, 45
  - type, 45
  - type\_len, 45
- ib\_col\_set\_value
  - dml, 36
- ib\_col\_type\_t
  - haildb.h, 63
- ib\_crsr\_t
  - haildb.h, 58
- ib\_cursor\_attach\_trx
  - haildb.h, 58

- cursor, 8
- ib\_cursor\_close
  - cursor, 8
- ib\_cursor\_delete\_row
  - dml, 37
- ib\_cursor\_first
  - cursor, 9
- ib\_cursor\_insert\_row
  - dml, 37
- ib\_cursor\_is\_positioned
  - cursor, 9
- ib\_cursor\_last
  - cursor, 9
- ib\_cursor\_lock
  - trx, 13
- ib\_cursor\_moveto
  - cursor, 9
- ib\_cursor\_next
  - cursor, 10
- ib\_cursor\_open\_index\_using\_id
  - cursor, 10
- ib\_cursor\_open\_index\_using\_name
  - cursor, 10
- ib\_cursor\_open\_table
  - cursor, 11
- ib\_cursor\_open\_table\_using\_id
  - cursor, 11
- ib\_cursor\_prev
  - cursor, 11
- ib\_cursor\_read\_row
  - dml, 37
- ib\_cursor\_reset
  - cursor, 11
- ib\_cursor\_set\_cluster\_access
  - dml, 37
- ib\_cursor\_set\_lock\_mode
  - trx, 13
- ib\_cursor\_set\_match\_mode
  - cursor, 12
- ib\_cursor\_set\_simple\_select
  - sql, 34
- ib\_cursor\_stmt\_begin
  - cursor, 12
- ib\_cursor\_truncate
  - ddl, 21
- ib\_cursor\_update\_row
  - dml, 37
- ib\_database\_create
  - ddl, 22
- ib\_database\_drop
  - ddl, 22
- ib\_err\_t
  - haikdb.h, 58
- ib\_error\_inject
  - debug, 44
- IB\_FALSE
  - haikdb.h, 55
- ib\_get\_duplicate\_key
  - trx, 13
- ib\_get\_index\_stat\_n\_diff\_key\_vals
  - misc, 29
- ib\_get\_table\_statistics
  - misc, 29
- ib\_i16\_t
  - haikdb.h, 58
- ib\_i32\_t
  - haikdb.h, 58
- ib\_i64\_t
  - haikdb.h, 59
- ib\_i8\_t
  - haikdb.h, 59
- ib\_id\_t
  - haikdb.h, 59
- ib\_idx\_sch\_t
  - haikdb.h, 59
- ib\_index\_create
  - ddl, 22
- ib\_index\_drop
  - ddl, 22
- ib\_index\_get\_id
  - ddl, 22
- ib\_index\_schema\_add\_col
  - ddl, 23
- ib\_index\_schema\_create
  - ddl, 23
- ib\_index\_schema\_delete
  - ddl, 23
- ib\_index\_schema\_set\_clustered
  - ddl, 23
- ib\_index\_schema\_set\_unique
  - ddl, 24
- ib\_init
  - init, 7
- ib\_lck\_mode\_t
  - haikdb.h, 63
- ib\_logger\_set
  - misc, 30
- ib\_match\_mode\_t
  - haikdb.h, 64
- IB\_MAX\_COL\_NAME\_LEN
  - haikdb.h, 55
- IB\_MAX\_TABLE\_NAME\_LEN
  - haikdb.h, 55
- ib\_msg\_log\_t
  - haikdb.h, 59
- ib\_msg\_stream\_t
  - haikdb.h, 59
- IB\_N\_SYS\_COLS

- haikdb.h, 56
- ib\_opaque\_t
  - haikdb.h, 59
- ib\_panic\_handler\_t
  - haikdb.h, 59
- ib\_savepoint\_release
  - trx, 13
- ib\_savepoint\_rollback
  - trx, 14
- ib\_savepoint\_take
  - trx, 14
- ib\_schema\_lock\_exclusive
  - ddl, 24
- ib\_schema\_lock\_is\_exclusive
  - ddl, 24
- ib\_schema\_lock\_is\_shared
  - haikdb.h, 66
- ib\_schema\_lock\_shared
  - ddl, 24
- ib\_schema\_tables\_iterate
  - ddl, 25
- ib\_schema\_unlock
  - ddl, 25
- ib\_schema\_visitor\_index\_col\_t
  - haikdb.h, 59
- ib\_schema\_visitor\_index\_t
  - haikdb.h, 59
- ib\_schema\_visitor\_t, 46
  - index, 46
  - index\_col, 46
  - table, 46
  - table\_col, 46
  - version, 46
- ib\_schema\_visitor\_table\_all\_t
  - haikdb.h, 60
- ib\_schema\_visitor\_table\_col\_t
  - haikdb.h, 60
- ib\_schema\_visitor\_table\_t
  - haikdb.h, 60
- ib\_schema\_visitor\_version\_t
  - haikdb.h, 64
- ib\_sec\_read\_tuple\_create
  - tuple, 32
- ib\_sec\_search\_tuple\_create
  - tuple, 32
- ib\_set\_client\_compare
  - misc, 30
- ib\_set\_panic\_handler
  - config, 20
- ib\_set\_trx\_is\_interrupted\_handler
  - config, 20
- ib\_shutdown
  - init, 7
- ib\_shutdown\_t
  - haikdb.h, 64
- IB\_SQL\_NULL
  - haikdb.h, 56
- ib\_srch\_mode\_t
  - haikdb.h, 64
- ib\_startup
  - init, 7
- ib\_status\_get\_all
  - config, 20
- ib\_status\_get\_i64
  - misc, 30
- ib\_strerror
  - misc, 30
- ib\_table\_create
  - ddl, 25
- ib\_table\_drop
  - ddl, 26
- ib\_table\_get\_id
  - ddl, 26
- ib\_table\_lock
  - trx, 14
- ib\_table\_rename
  - ddl, 26
- ib\_table\_schema\_add\_col
  - ddl, 26
- ib\_table\_schema\_add\_index
  - ddl, 27
- ib\_table\_schema\_create
  - ddl, 27
- ib\_table\_schema\_delete
  - ddl, 27
- ib\_table\_schema\_visit
  - ddl, 28
- ib\_table\_stats\_t, 47
  - stat\_clustered\_index\_size, 47
  - stat\_modified\_counter, 47
  - stat\_n\_rows, 47
  - stat\_sum\_of\_other\_index\_sizes, 47
- ib\_table\_truncate
  - ddl, 28
- ib\_tbl\_fmt\_t
  - haikdb.h, 65
- ib\_tbl\_sch\_add\_blob\_col
  - haikdb.h, 56
- ib\_tbl\_sch\_add\_text\_col
  - haikdb.h, 56
- ib\_tbl\_sch\_add\_u32\_col
  - haikdb.h, 56
- ib\_tbl\_sch\_add\_u64\_col
  - haikdb.h, 56
- ib\_tbl\_sch\_add\_u64\_notnull\_col
  - haikdb.h, 57
- ib\_tbl\_sch\_add\_varchar\_col
  - haikdb.h, 57

- ib\_tbl\_sch\_t
  - haiddb.h, 60
- ib\_tpl\_t
  - haiddb.h, 60
- IB\_TRUE
  - haiddb.h, 57
- ib\_trx\_begin
  - trx, 15
- ib\_trx\_commit
  - trx, 15
- ib\_trx\_is\_interrupted\_handler\_t
  - haiddb.h, 60
- ib\_trx\_level\_t
  - haiddb.h, 65
- ib\_trx\_release
  - trx, 15
- ib\_trx\_rollback
  - trx, 15
- ib\_trx\_set\_client\_data
  - trx, 16
- ib\_trx\_start
  - trx, 16
- ib\_trx\_state
  - trx, 16
- ib\_trx\_state\_t
  - haiddb.h, 65
- ib\_trx\_t
  - haiddb.h, 60
- ib\_tuple\_clear
  - tuple, 32
- ib\_tuple\_copy
  - tuple, 32
- ib\_tuple\_delete
  - tuple, 33
- ib\_tuple\_get\_cluster\_key
  - tuple, 33
- ib\_tuple\_get\_n\_cols
  - tuple, 33
- ib\_tuple\_get\_n\_user\_cols
  - tuple, 33
- ib\_tuple\_read\_double
  - dml, 38
- ib\_tuple\_read\_float
  - dml, 38
- ib\_tuple\_read\_i16
  - dml, 38
- ib\_tuple\_read\_i32
  - dml, 39
- ib\_tuple\_read\_i64
  - dml, 39
- ib\_tuple\_read\_i8
  - dml, 39
- ib\_tuple\_read\_u16
  - dml, 39
- ib\_tuple\_read\_u32
  - dml, 40
- ib\_tuple\_read\_u64
  - dml, 40
- ib\_tuple\_read\_u8
  - dml, 40
- ib\_tuple\_write\_double
  - dml, 41
- ib\_tuple\_write\_float
  - dml, 41
- ib\_tuple\_write\_i16
  - dml, 41
- ib\_tuple\_write\_i32
  - dml, 41
- ib\_tuple\_write\_i64
  - dml, 42
- ib\_tuple\_write\_i8
  - dml, 42
- ib\_tuple\_write\_u16
  - dml, 42
- ib\_tuple\_write\_u32
  - dml, 43
- ib\_tuple\_write\_u64
  - dml, 43
- ib\_tuple\_write\_u8
  - dml, 43
- ib\_u16\_t
  - haiddb.h, 60
- ib\_u32\_t
  - haiddb.h, 60
- ib\_u64\_t
  - haiddb.h, 61
- ib\_u8\_t
  - haiddb.h, 61
- ib\_ulint\_t
  - haiddb.h, 61
- ib\_update\_table\_statistics
  - misc, 31
- index
  - ib\_schema\_visitor\_t, 46
- index\_col
  - ib\_schema\_visitor\_t, 46
- init
  - ib\_init, 7
  - ib\_shutdown, 7
  - ib\_startup, 7
- MAX\_TEXT\_LEN
  - haiddb.h, 57
- misc
  - ib\_api\_version, 29
  - ib\_get\_index\_stat\_n\_diff\_key\_vals, 29
  - ib\_get\_table\_statistics, 29
  - ib\_logger\_set, 30

- [ib\\_set\\_client\\_compare](#), 30
    - [ib\\_status\\_get\\_i64](#), 30
    - [ib\\_strerror](#), 30
    - [ib\\_update\\_table\\_statistics](#), 31
  - Miscellaneous functions, 28
  - sql
    - [ib\\_cursor\\_set\\_simple\\_select](#), 34
  - SQL functions, 34
  - Startup/Shutdown functions, 7
  - [stat\\_clustered\\_index\\_size](#)
    - [ib\\_table\\_stats\\_t](#), 47
  - [stat\\_modified\\_counter](#)
    - [ib\\_table\\_stats\\_t](#), 47
  - [stat\\_n\\_rows](#)
    - [ib\\_table\\_stats\\_t](#), 47
  - [stat\\_sum\\_of\\_other\\_index\\_sizes](#)
    - [ib\\_table\\_stats\\_t](#), 47
- table
- [ib\\_schema\\_visitor\\_t](#), 46
- table\_col
- [ib\\_schema\\_visitor\\_t](#), 46
- Transaction functions, 12
- trx
- [ib\\_cursor\\_lock](#), 13
  - [ib\\_cursor\\_set\\_lock\\_mode](#), 13
  - [ib\\_get\\_duplicate\\_key](#), 13
  - [ib\\_savepoint\\_release](#), 13
  - [ib\\_savepoint\\_rollback](#), 14
  - [ib\\_savepoint\\_take](#), 14
  - [ib\\_table\\_lock](#), 14
  - [ib\\_trx\\_begin](#), 15
  - [ib\\_trx\\_commit](#), 15
  - [ib\\_trx\\_release](#), 15
  - [ib\\_trx\\_rollback](#), 15
  - [ib\\_trx\\_set\\_client\\_data](#), 16
  - [ib\\_trx\\_start](#), 16
  - [ib\\_trx\\_state](#), 16
- tuple
- [ib\\_clust\\_read\\_tuple\\_create](#), 31
  - [ib\\_clust\\_search\\_tuple\\_create](#), 32
  - [ib\\_sec\\_read\\_tuple\\_create](#), 32
  - [ib\\_sec\\_search\\_tuple\\_create](#), 32
  - [ib\\_tuple\\_clear](#), 32
  - [ib\\_tuple\\_copy](#), 32
  - [ib\\_tuple\\_delete](#), 33
  - [ib\\_tuple\\_get\\_cluster\\_key](#), 33
  - [ib\\_tuple\\_get\\_n\\_cols](#), 33
  - [ib\\_tuple\\_get\\_n\\_user\\_cols](#), 33
- Tuple functions, 31
- type
- [ib\\_col\\_meta\\_t](#), 45
- type\_len
- [ib\\_col\\_meta\\_t](#), 45
- UNIV\_NO\_IGNORE
- [haibdb.h](#), 58
- version
- [ib\\_schema\\_visitor\\_t](#), 46