Name

mariadb_stmt_execute_direct - prepares and executes a prepared statement

Synopsis

```
#include <mysql.h>

int mariadb_stmt_execute_direct(MYSQL_STMT * stmt,
                                const char *query,
                                size_t length);
```

Description

Prepares and executes a statement which was previously allocated by **mysql_stmt_init(3)**, using the current values of the parameter variables if any parameters exist in the statement.

Parameters

- stmt - A statement handle, which was previously allocated by **mysql_stmt_init(3)**.

- query SQL statement

- length Length of SQL statement

Return value

Returns zero on success, non-zero on failure.

Notes

- Since the number of parameter of the statement is unknown before execution it is mandatory to set the number of parameters via the **mysql_stmt_attr_set(3)** function.

- If the SQL statement is a zero-terminated string, you can also pass -1 as length.

- The statement handle is intended for one-time execution. Reusing the statement handle might lead to unexpected behavior.

History

This function was added in Connector/C 3.0 and requires MariaDB 10.2 or later versions.

See Also

- **mysql_stmt_attr_set(3)**

- **mysql_stmt_bind_param(3)**

Example

‘‘‘C static int execute_direct_example(MYSQL *mysql) { MYSQL_STMT* stmt= mysql_stmt_init(mysql); MYSQL_BIND bind[2]; int intval= 1; int param_count= 2; char *strval= "execute_direct_example";

/* Direct execution without parameters */ if (mariadb_stmt_execute_direct(stmt, "CREATE TABLE execute_direct (a int, b varchar(30))", -1)) goto error;

memset(&bind, 0, sizeof(MYSQL_BIND) * 2); bind[0].buffer_type= MYSQL_TYPE_SHORT; bind[0].buffer= &intval; bind[1].buffer_type= MYSQL_TYPE_STRING; bind[1].buffer= strval; bind[1].buffer_length= strlen(strval);

/* set number of parameters */ if (mysql_stmt_attr_set(stmt, STMT_ATTR_PREBIND_PARAMS, &param_count)) goto error;

/* bind parameters */ if (mysql_stmt_bind_param(stmt, bind)) goto error;

if (mariadb_stmt_execute_direct(stmt, "INSERT INTO execute_direct VALUES (?,?)", -1)) goto error;

mysql_stmt_close(stmt); return 0; error: printf("Error: %s", mysql_stmt_error(stmt)); mysql_stmt_close(stmt); return 1; }