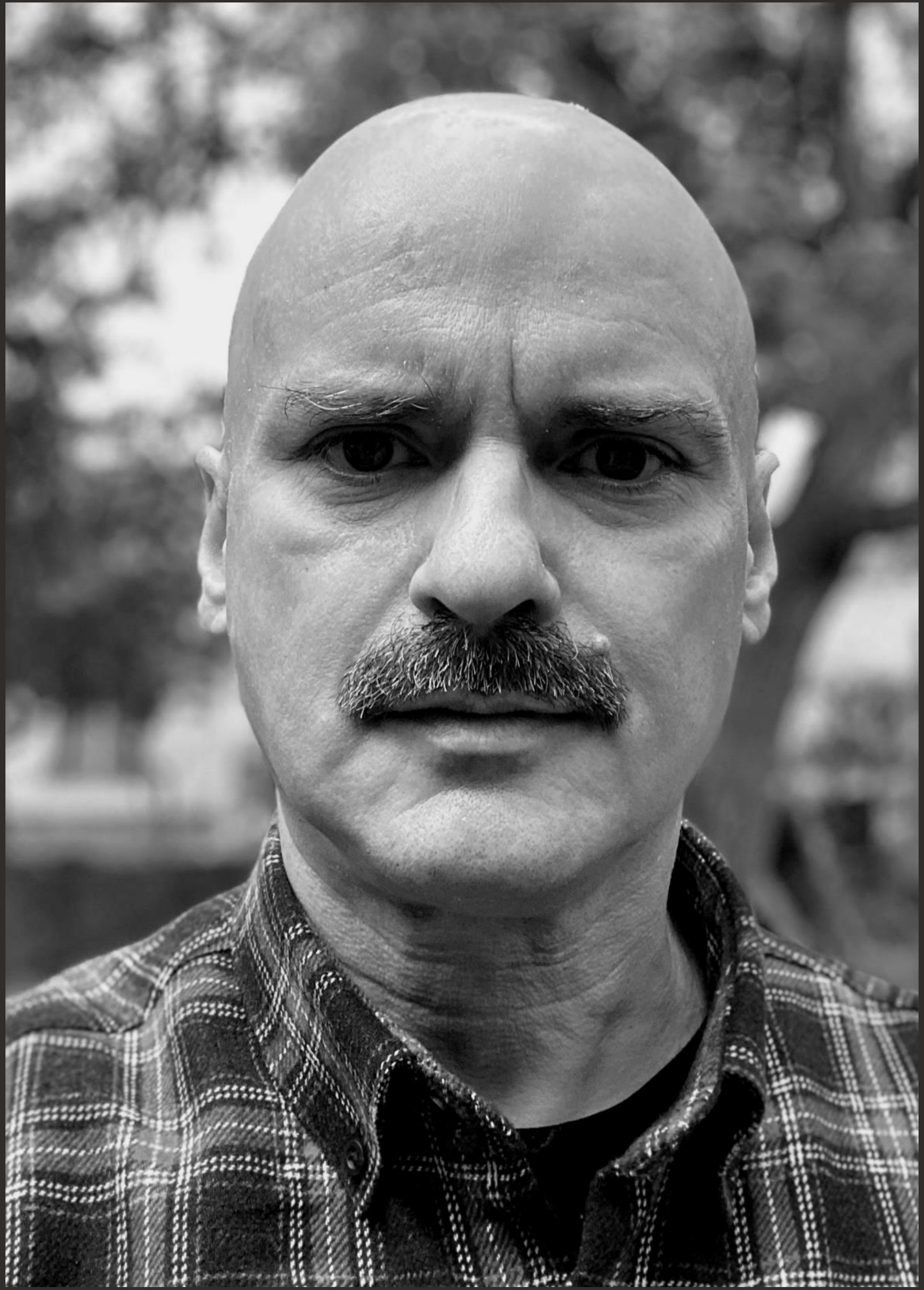
ORACLE

MySQL Protocol's Query Attributes

Passing extra (meta-)data about/to queries over the wire

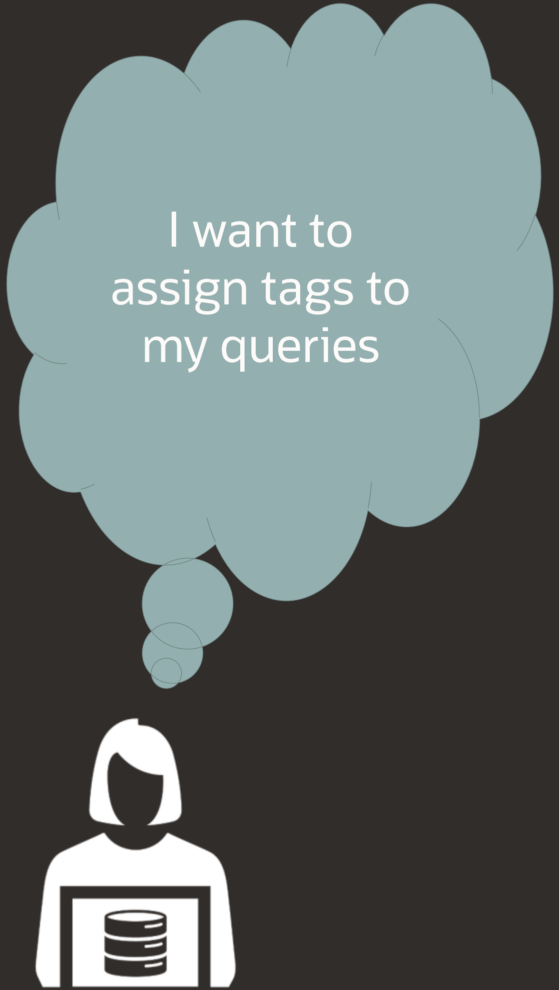
Georgi Kodinov
Team Lead,
MySQL Server General team

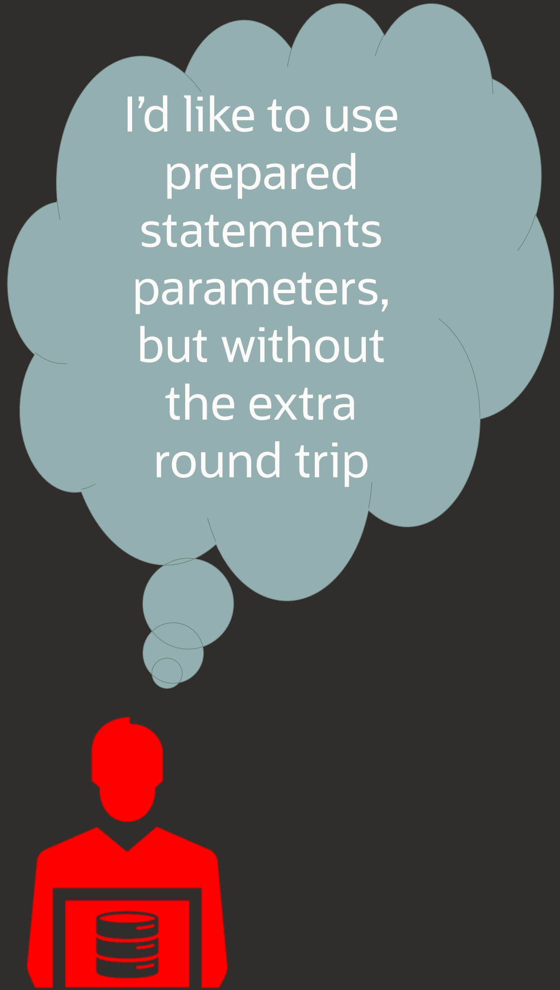


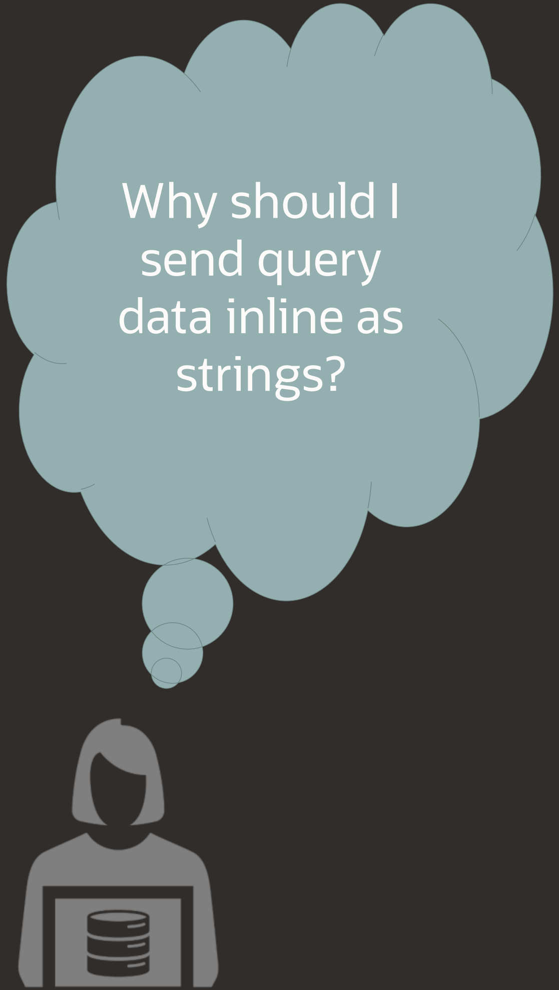
About Me: Georgi “Joro” Kodinov

- Working on MySQL since 2006
- Based in Plovdiv, Bulgaria
- Background in Banking IT

So Many Questions!




I'd like to use prepared statements parameters, but without the extra round trip



Seeking Answers

Where All Good Queries Go

Simple!
Fast!

Payload

Type	Name	Description
int<1>	command	0x03: COM_QUERY
string<EOF>	query	the text of the SQL query to execute

Payload

Type	Name	Description
int<1>	command	0x16: COM_STMT_PREPARE
string<EOF>	query	The query to prepare

Payload

Type	Name	Description
int<1>	status	[0x17] COM_STMT_EXECUTE
int<4>	statement_id	ID of the prepared statement to execute
int<1>	flags	Flags. See enum_cursor_type
int<4>	iteration_count	Number of times to execute the statement. Currently always 1.
if num_params > 0 {		
binary<var>	null_bitmap	NULL bitmap, length= (num_params + 7) / 8
int<1>	new_params_bind_flag	Flag if parameters must be re-bound
if new_params_bind_flag {		
binary<var>	parameter_types	Type of each parameter, length: num_params * 2
binary<var>	parameter_values	value of each parameter

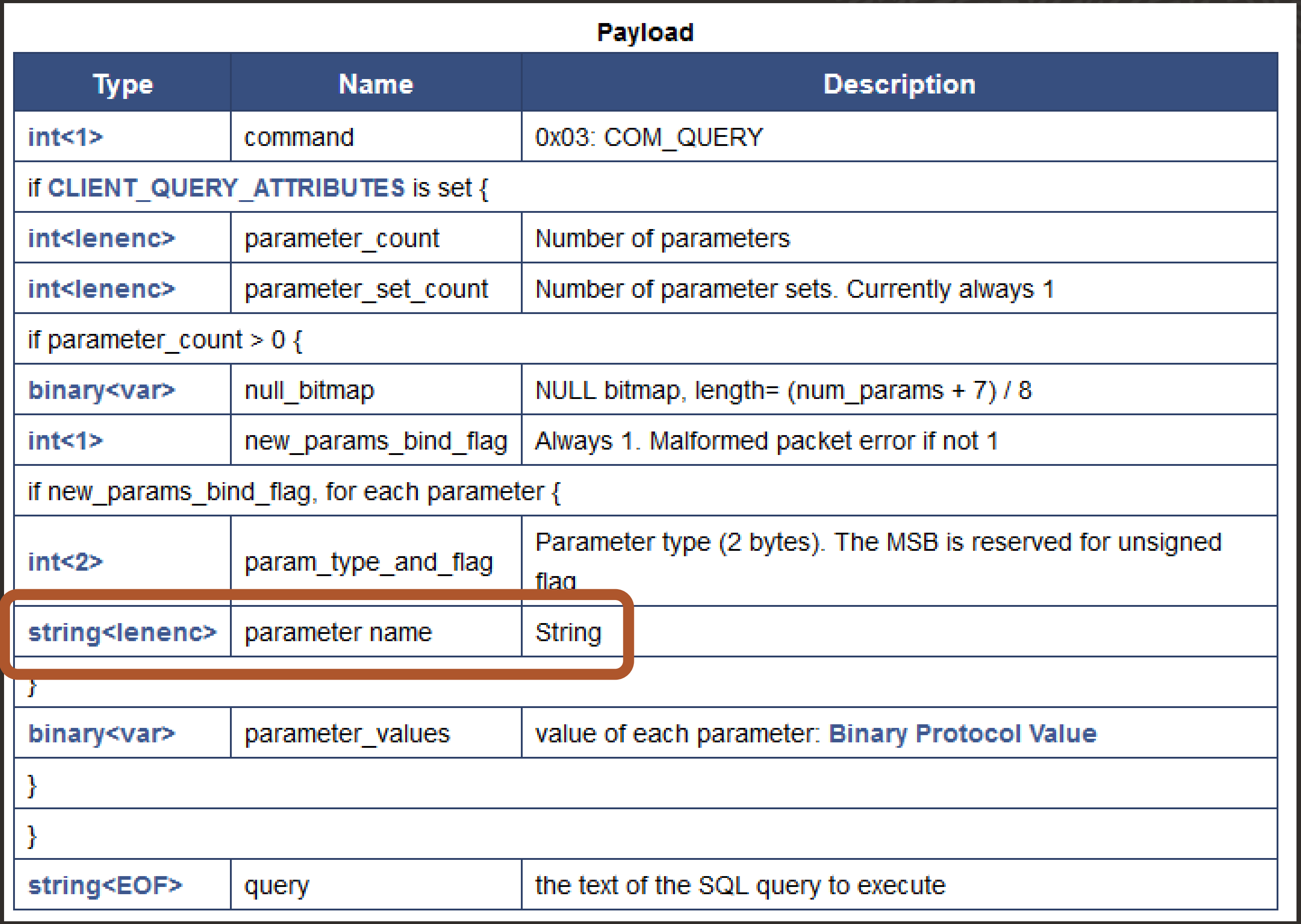
Powerful!
Secure!

We select a possibility and we walk
until we reach it.
So, in a sense, we create it.

Roger Zelazny

The Chronicles of Amber

Taaa Daaa! Magic!



Payload

Type	Name	Description
int<1>	command	0x03: COM_QUERY
if CLIENT_QUERY_ATTRIBUTES is set {		
int<lenenc>	parameter_count	Number of parameters
int<lenenc>	parameter_set_count	Number of parameter sets. Currently always 1
if parameter_count > 0 {		
binary<var>	null_bitmap	NULL bitmap, length= (num_params + 7) / 8
int<1>	new_params_bind_flag	Always 1. Malformed packet error if not 1
if new_params_bind_flag, for each parameter {		
int<2>	param_type_and_flag	Parameter type (2 bytes). The MSB is reserved for unsigned flag
string<lenenc>	parameter name	String
}		
binary<var>	parameter_values	value of each parameter: Binary Protocol Value
}		
}		
string<EOF>	query	the text of the SQL query to execute

Except This one!

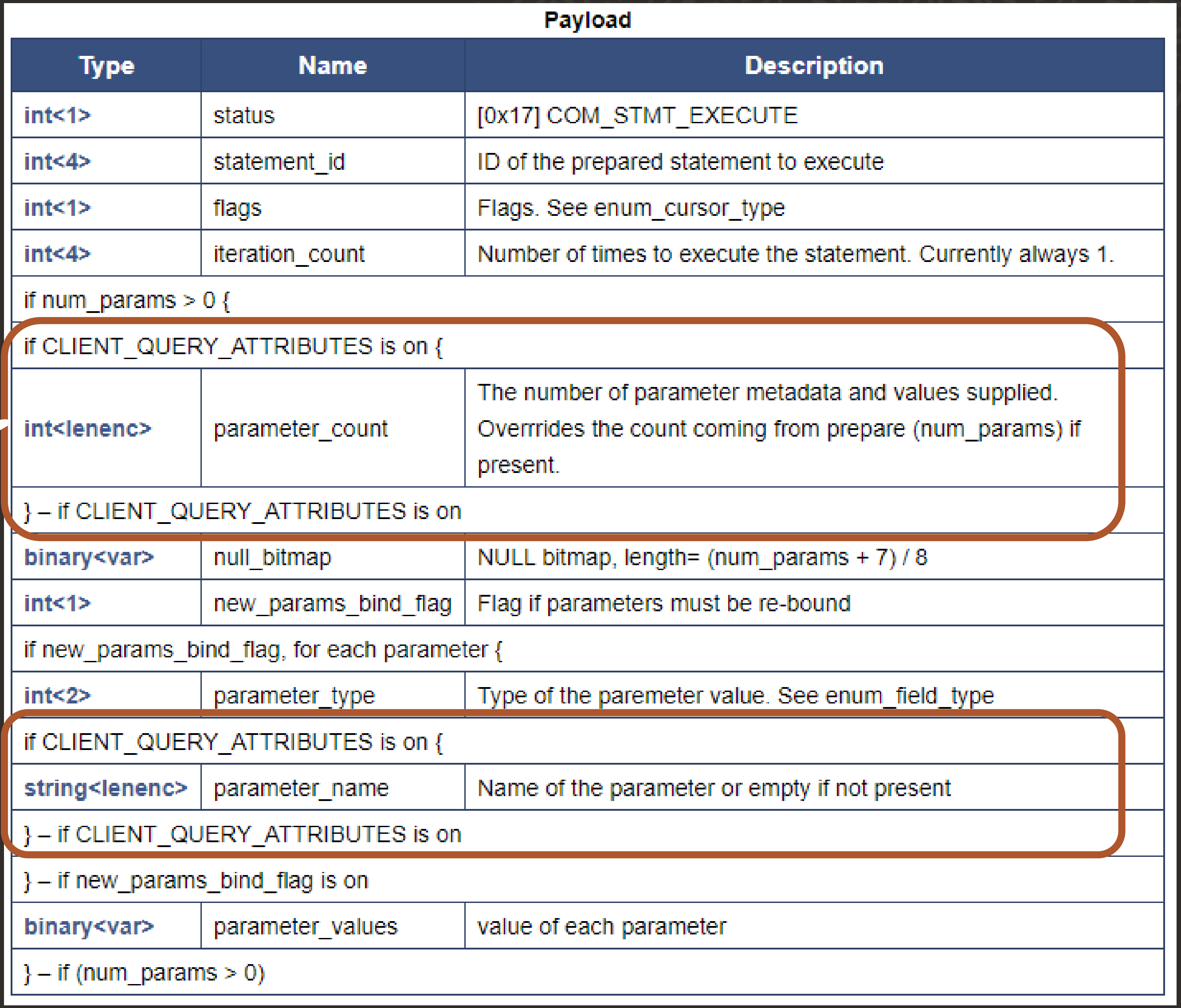
NEW !
Standard COM_STMT_EXECUTE! In a COM_QUERY!

Uniformity is Important!

Two data sets now:

- Parameters
- Metadata

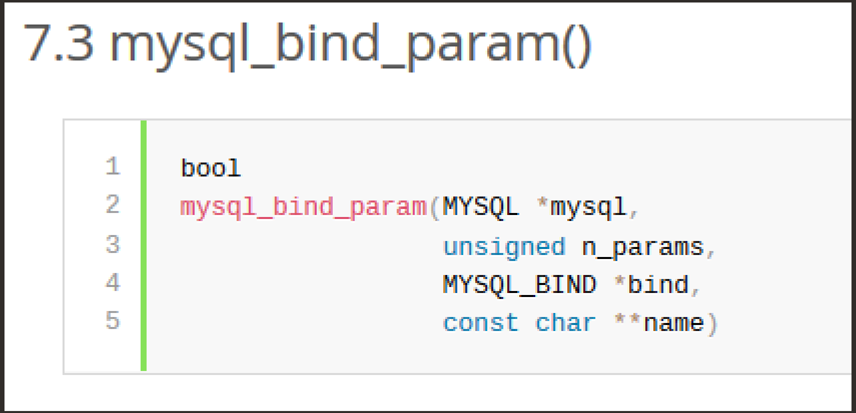
And of course:
The Names!



Payload

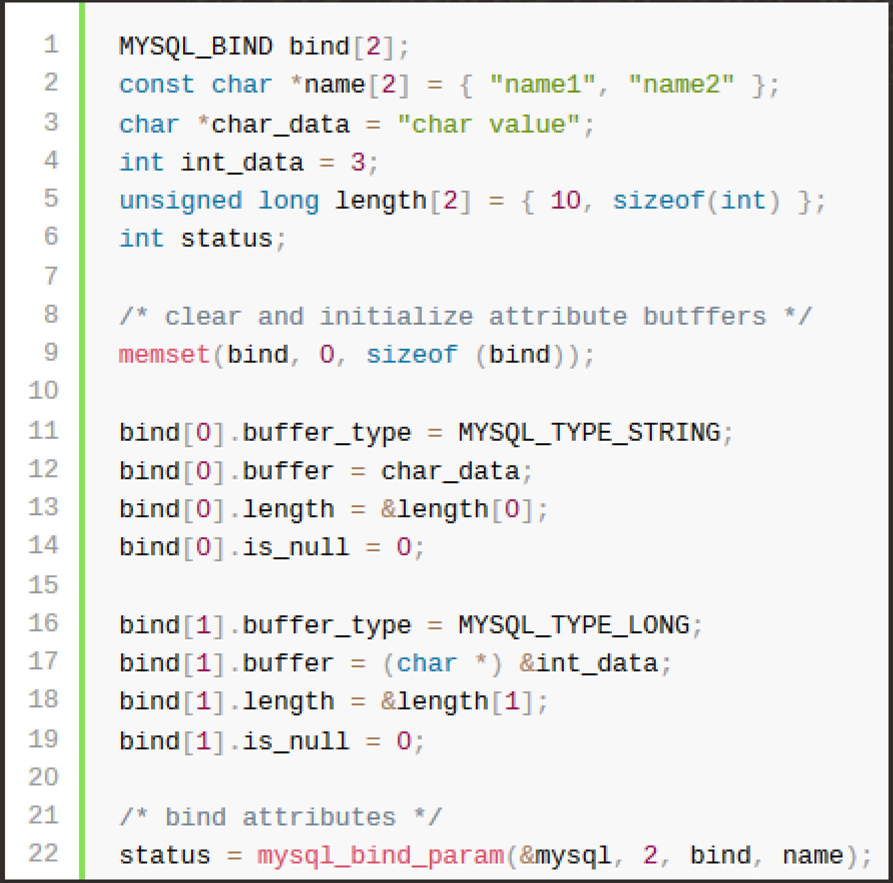
Type	Name	Description
int<1>	status	[0x17] COM_STMT_EXECUTE
int<4>	statement_id	ID of the prepared statement to execute
int<1>	flags	Flags. See enum_cursor_type
int<4>	iteration_count	Number of times to execute the statement. Currently always 1.
if num_params > 0 {		
if CLIENT_QUERY_ATTRIBUTES is on {		
int<lenenc>	parameter_count	The number of parameter metadata and values supplied. Overrrides the count coming from prepare (num_params) if present.
} – if CLIENT_QUERY_ATTRIBUTES is on		
binary<var>	null_bitmap	NULL bitmap, length= (num_params + 7) / 8
int<1>	new_params_bind_flag	Flag if parameters must be re-bound
if new_params_bind_flag, for each parameter {		
int<2>	parameter_type	Type of the paremeter value. See enum_field_type
if CLIENT_QUERY_ATTRIBUTES is on {		
string<lenenc>	parameter_name	Name of the parameter or empty if not present
} – if CLIENT_QUERY_ATTRIBUTES is on		
} – if new_params_bind_flag is on		
binary<var>	parameter_values	value of each parameter
} – if (num_params > 0)		

In the C API

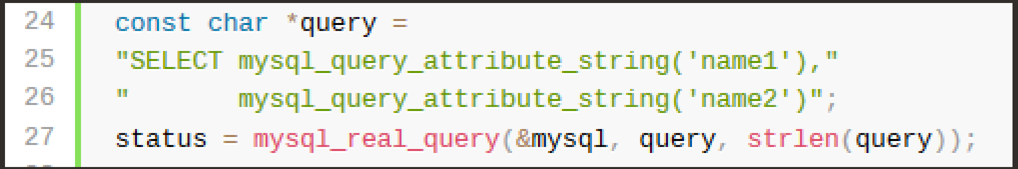


7.3 mysql_bind_param()

```c
bool
mysql_bind_param(MYSQL *mysql,
                 unsigned n_params,
                 MYSQL_BIND *bind,
                 const char **name)
```



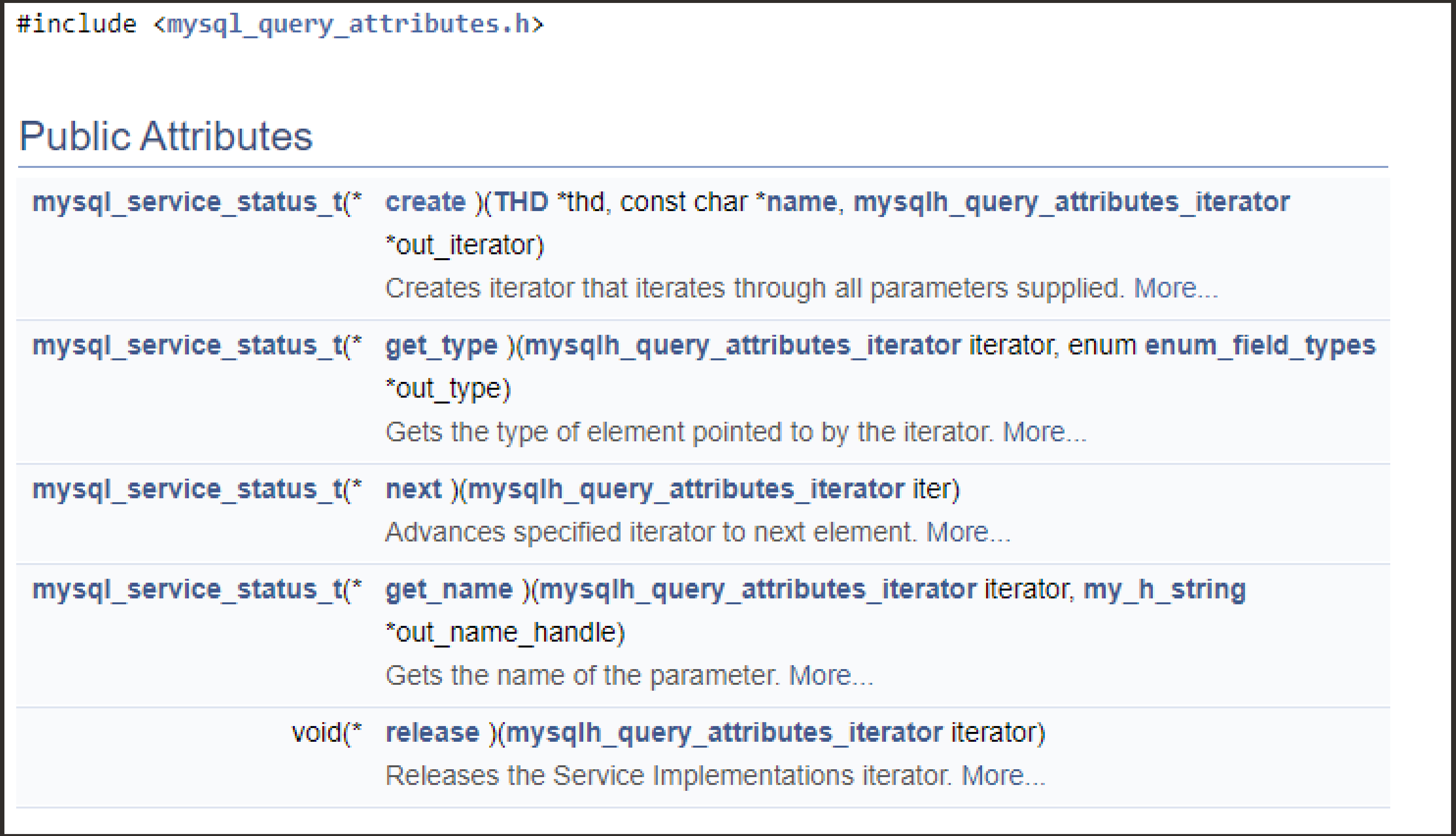
```c
MYSQL_BIND bind[2];
const char *name[2] = { "name1", "name2" };
char *char_data = "char value";
int int_data = 3;
unsigned long length[2] = { 10, sizeof(int) };
int status;

/* clear and initialize attribute butffers */
memset(bind, 0, sizeof (bind));

bind[0].buffer_type = MYSQL_TYPE_STRING;
bind[0].buffer = char_data;
bind[0].length = &length[0];
bind[0].is_null = 0;

bind[1].buffer_type = MYSQL_TYPE_LONG;
bind[1].buffer = (char *) &int_data;
bind[1].length = &length[1];
bind[1].is_null = 0;

/* bind attributes */
status = mysql_bind_param(&mysql, 2, bind, name);
```



```c
const char *query =
"SELECT mysql_query_attribute_string('name1'),"
"       mysql_query_attribute_string('name2')";
status = mysql_real_query(&mysql, query, strlen(query));
```

In The Server

The iterator component service



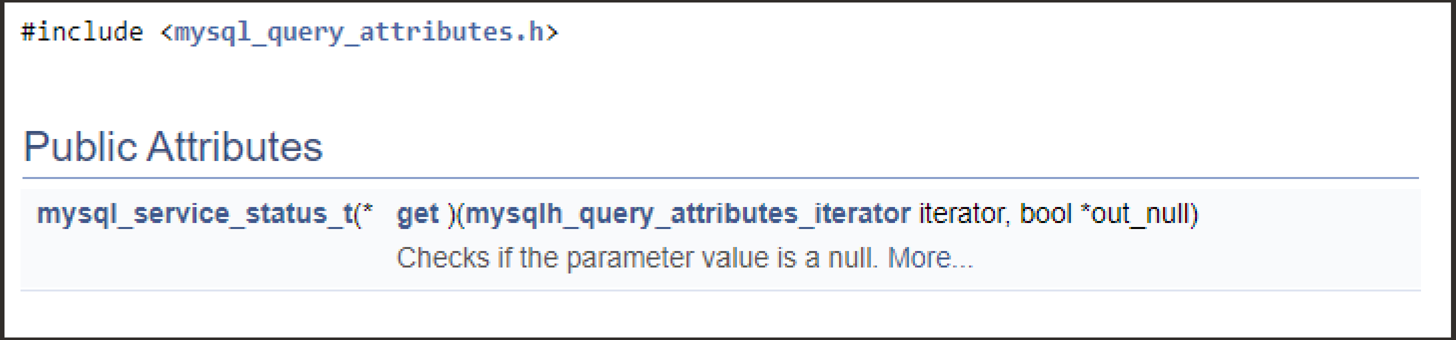
```
#include <mysql_query_attributes.h>
```

Public Attributes

mysql_service_status_t(*	create)(THD *thd, const char *name, mysqlh_query_attributes_iterator *out_iterator) Creates iterator that iterates through all parameters supplied. More...
mysql_service_status_t(*	get_type)(mysqlh_query_attributes_iterator iterator, enum enum_field_types *out_type) Gets the type of element pointed to by the iterator. More...
mysql_service_status_t(*	next)(mysqlh_query_attributes_iterator iter) Advances specified iterator to next element. More...
mysql_service_status_t(*	get_name)(mysqlh_query_attributes_iterator iterator, my_h_string *out_name_handle) Gets the name of the parameter. More...
void(*	release)(mysqlh_query_attributes_iterator iterator) Releases the Service Implementations iterator. More...

In The Server

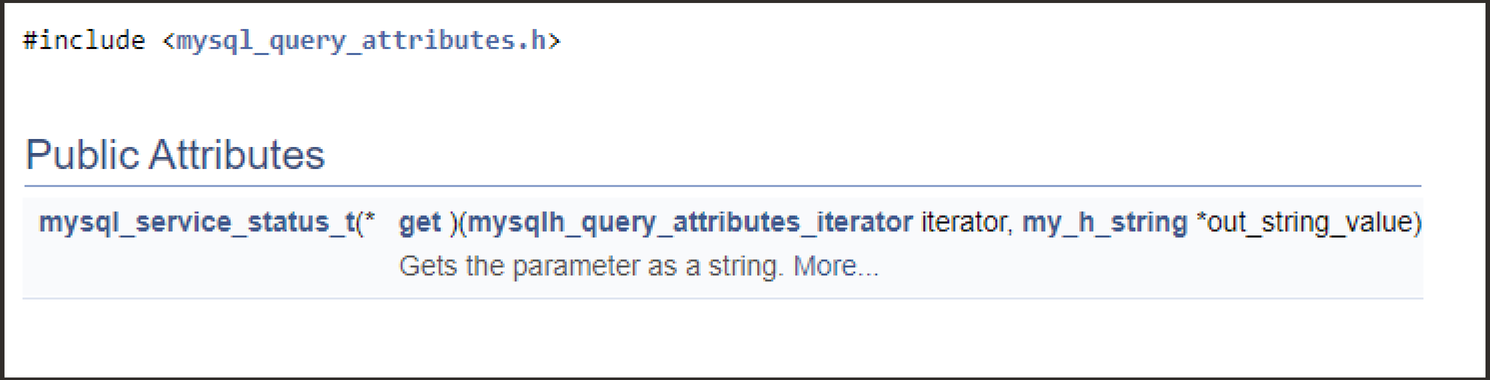
The Null check and the get-as-string component services



```
#include <mysql_query_attributes.h>
```

Public Attributes

mysql_service_status_t(* get)(mysqlh_query_attributes_iterator iterator, bool *out_null)
Checks if the parameter value is a null. More...



```
#include <mysql_query_attributes.h>
```

Public Attributes

mysql_service_status_t(* get)(mysqlh_query_attributes_iterator iterator, my_h_string *out_string_value)
Gets the parameter as a string. More...

In The Server

A UDF to serve the value to SQL

```
mysql> query_attributes n1 v1 n2 v2;
mysql> SELECT
    ->   mysql_query_attribute_string('n1') AS 'attr 1',
    ->   mysql_query_attribute_string('n2') AS 'attr 2',
    ->   mysql_query_attribute_string('n3') AS 'attr 3';
+--------+--------+--------+
| attr 1 | attr 2 | attr 3 |
+--------+--------+--------+
| v1     | v2     | NULL   |
+--------+--------+--------+
```

The Cost

- One new capability flag: CLIENT_QUERY_ATTRIBUTES: for backward compatibility
- One extra (zero) byte for COM_QUERY without query attributes: for the count
- One extra (zero) byte in COM_STMT_EXECUTE: for the count
- One extra (zero) byte per parameter value in COM_STMT_EXECUTE: for the name

The Road Ahead

- Make the SQL parser process named parameter markers syntax
- Extend the component service APIs to deal with more (which ones ?) data types
- Extend the C API to allow passing named parameters via COM_STMT_EXECUTE
- …
- **What else? Georgi.Kodinov@oracle.com**

Thanks!

- To the Facebook engineering team for the initial contribution!
- To the database engineers at Booking.com for championing the idea and developing my understanding of the needs

That’s all, folks!

Questions ?