Python: how to use prepared queries in MySQL Connector

In this article we will see how to make prepared queries with MySQL Connector in Python.

A prepared query, as its name suggests, is a query in which the values are not inserted directly within the string but the latter are replaced by placeholders (indicated with the sequence `%s`).

Upon query execution, MySQL Connector will filter, sanitize and validate the values passed as tuples by replacing them with the specified placeholder characters in the same order as they appear in the query string.

To use this feature, a cursor must be initialized using the `prepared` parameter set to `True`.

```
import mysql.connector

db = mysql.connector.connect(host='localhost',
                                 user='username',
                                 password='password',
                                 database='test')

cursor = db.cursor(prepared=True, dictionary=True)
query = 'SELECT * FROM items WHERE id = %s'
values = (10,)
cursor.execute(query,values)

item = cursor.fetchone()
cursor.close()
```

```
if item is not None:
    print(item['name'])
```

In this example we are retrieving a single row from the `items` table using the value of its identifier (`id`). This value is passed dynamically to the query, so there may be a risk of SQL injection.

For this reason we use a prepared statement specifying the `prepared` parameter set to `True` when instantiating a `cursor` object. So in the query string we use the placeholder `%s` to indicate that that value will have to be validated and replaced. At that point in the tuple `values` we specify the actual value to replace and execute the query. If there is a match in the table, the `fetchone()` method will return the found row as a dictionary. In fact, we will obtain a dictionary because in this case the `cursor` object has been instantiated with the boolean parameter `dictionary` set to `True`.

The same technique also applies to other types of queries. For example if we wanted to insert a new row in the same table, we could write:

```
cursor = db.cursor(prepared=True)
query = 'INSERT INTO items (name,price) VALUES
(%s,%s)'
values = ('New Item',9.50)
cursor.execute(query,values)
db.commit()
```

In this case, not having to find any data, the cursor is initialized with only the `prepared` parameter. In summary, prepared queries turn out to be an essential feature to be able to write secure queries using the MySQL Connector in Python.