How to connect Python programs to MariaDB

Posted on November 14, 2014 by MariaDB (/resources/blog)

Download (https://mariadb.com/downloads/) **Contact (https://mariadb.com/contact/)**

You can use the popular programming language Python to manage data stored in MariaDB. Here is everything you need to know about connecting to MariaDB from Python for retrieving, updating, and inserting information.

Preparation and installation

If you don't already have MariaDB installed on your system, check the official MariaDB installation instructions (https://mariadb.com/kb/en/mariadb/documentation/getting-started/getting-installing-and-upgrading-mariadb/) or read the blog post on How to install MariaDB on CentOS 7 (https://mariadb.com/blog/installing-mariadb-10-centos-7-rhel-7).

If you want to try out Python integration with MariaDB but you don't have a database to play with, you can use the popular employees example database (https://launchpad.net/test-db/).

MariaDB provides Python support through the MySQL Python package (https://pypi.python.org/pypi/MySQL-python), which does not come installed with the default Python installation on most distros. To add it, use any of the installation packages from the official package page or your distribution's repository.

The basic Python code

To connect to MariaDB using the MySQL Python module in your program, you have to import it first, just as you would any other module. For clarity and ease of use, import the connector class only under the name mariadb: `import mysql.connector as mariadb`. I'll use the class under the mariadb name in the following examples.

Next, establish a database connection with code like `mariadb_connection = mariadb.connect(user='python_user', password='some_pass', database='employees')`, where you assign real values for user, password, and database.

Finally, to start interacting with the database and running queries, you need to instantiate the cursor object with the code `cursor = mariadb_connection.cursor()`. So far your initial code should look like this:

Download (https://mariadb.com/downloads/) Contact (https://mariadb.com/contact/)

```
#!/usr/bin/python
import mysql.connector as mariadb

mariadb_connection = mariadb.connect(user='python_user', password='some
cursor = mariadb_connection.cursor()
```

Retrieving information

Once you have the initial code in place you can start working with the data. The first thing you should do is try to retrieve information from the database. Here is code for a query against the employees database:

```
cursor.execute("SELECT first_name,last_name FROM employees WHERE first_
```

This code uses a variable string (`%s`) which is assigned from the `some_name` variable that follows in parentheses. You should have assigned the variable already to a name. Use exactly this syntax with the same number of arguments to ensure that your code works.

(https://mariadb.com) (https://mariadb.com)

The result of the query is stored in a list called "cursor." To test the result you can print it with a simple `for` loop, but for better formatting use Python's string formatting method:

```
for first_name, last_name in cursor:
    print("First name: {}, Last name: {}").format(first_name,last_name)
```

Inserting rows

Download (https://mariadb.com/downloads/) **Contact (https://mariadb.com/contact/)**

You can insert rows into a table in a way similar to retrieving it by using the `cursor.execute` method: `cursor.execute("INSERT INTO employees (first_name,last_name) VALUES (%s,%s)", (first_name, last_name))`. Here you should have already assigned the `first_name` and `last_name` variables. By default AUTOCOMMIT is disabled (https://mariadb.com/blog/every-select-your-python-program-may-acquire-metadata-lock), meaning queries are not committed, so no data will be saved until you manually commit with the connection `commit` method: `mariadb_connection.commit()`.

You should commit as soon as you are certain that the data is correct and should be recorded. This allows you to continue with a new transaction if needed. MariaDB allows you to run multiple concurrent transaction on the same table without locking it when you use XtraDB (InnoDB) engine.

Just as in SQL, the opposite method to commit is rollback. Thus, if you wish to discard the changes from the last queries, you can use the `rollback()` method: `mariadb_connection.rollback()`.

While inserting rows you may wish to find the ID of the last inserted row when it is automatically generated, as with autoincrement values. You can acquire this useful information with the `insert_id()` method of the connection class: `mariadb_connection.insert_id()`.

Updating and deleting rows is done similarly to inserting them. The only difference is in the query used.

Trapping errors

For any of your SQL actions (querying, updating, deleting, or inserting records) you should try to trap errors, so you can verify that your actions are being executed as expected and you know about any problems as they occur. To trap errors, use the `Error` class:

```
try:
    cursor.execute("some MariaDB query"))
except mariadb.Error as error:
    print("Error: {}".format(error))
```

Download (https://mariadb.com/downloads/) **Contact (https://mariadb.com/contact/)**

If the query in the `try` clause of the above code fails, MariaDB will raise an SQL exception and you will see it printed, properly formatted. This programming best practice for trapping errors is especially important when you're working with a database, because you need to ensure the integrity of the information.

Once you finish working with the database make sure that you close this connection to avoid keeping unused connections open and thus wasting resources. You can close the connection with the `close()` method: `mariadb_connection.close()`

This is how easy and straightforward it is to connect your Python code to a MariaDB database. Here is how a complete script should look like:

```
#!/usr/bin/python
import mysql.connector as mariadb

mariadb_connection = mariadb.connect(user='python_user', password='some
cursor = mariadb_connection.cursor()

#retrieving information
some_name = 'Georgi'
cursor.execute("SELECT first_name,last_name FROM employees WHERE first_

for first_name, last_name in cursor:
    print("First name: {}, Last name: {}").format(first_name,last_name)

#insert information
try:
    cursor.execute("INSERT INTO employees (first_name,last_name) VALUES
except mariadb.Error as error:
    print("Error: {}".format(error))

mariadb_connection.commit()
print "The last inserted id was: ", cursor.lastrowid

mariadb_connection.close()
```

Download (https://mariadb.com/downloads/) **Contact (https://mariadb.com/contact/)**

BACK TO BLOG (/RESOURCES/BLOG)

(https://mariadb.com) (https://mariadb.com)

PRODUCTS (HTTPS://MARIADB.COM/PRODUCTS/)

MariaDB Platform (https://mariadb.com/products/mariadb-platform/)

Platform Managed Service (https://mariadb.com/products/mariadb-platform-managed-service/)

ClustrixDB (https://mariadb.com/products/clustrixdb/)

Pricing (https://mariadb.com/pricing/)

Download MariaDB (https://mariadb.com/downloads/)

SERVICES (HTTPS://MARIADB.COM/SERVICES/)

Remote DBA (https://mariadb.com/services/remote-db

Enterprise Architect (https://mariadb.com/services/enterprise-architect/)

Technical Support (https://mariadb.com/services/technical-support-services/)

Migration Practice (https://mariadb.com/services/migration-practice/)

Consulting (https://mariadb.com/services/consulting/)

Training (https://mariadb.com/services/training/)

RESOURCES (HTTPS://MARI

Customer Stories

tor

Blog (https://mariad

ari

OpenWorks 2019 (ht

Roadshow (https://n

Support (https://sup

Subscribe to our newsletter!

YOUR EMAIL ADDRES

-- Please Select Coun

ADD ME

Download (https://mariadb.com/downloads/)

Contact (https://mariadb.com/contact/)

Legal (https://mariadb.com/legal/) | Privacy Policy (https://mariadb.com/privacy-policy/) | Cookie Policy (https://mariadb.com/cookie-policy/)| Sitemap (https://mariadb.com/sitemap.xml)

(https://www.facebook.com/M

(https://twitter.com/mariadb)

(https://www.linkedin.com/co
corporation?trk=