Connecting to MySQL Using Connector/Python

The connect() constructor creates a connection to the MySQL server and returns a MySQLConnection object.

The following example shows how to connect to the MySQL server:

import mysql.connector

cnx = mysql.connector.connect(user='scott', password='*password*',

host='127.0.0.1',

database='employees')

cnx.close()

It is also possible to create connection objects using the [connection.MySQLConnection()](https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlconnection.html) class:

from mysql.connector import (connection)

cnx = connection.MySQLConnection(user='scott', password='password',

host='127.0.0.1',

database='employees')

cnx.close()

To handle connection errors, use the try statement and catch all errors using the [errors.Error](https://dev.mysql.com/doc/connector-python/en/connector-python-api-errors-error.html) exception:

import mysql.connector

from mysql.connector import errorcode

try:

cnx = mysql.connector.connect(user='scott',

database='testt')

except mysql.connector.Error as err:

if err.errno == errorcode.ER\_ACCESS\_DENIED\_ERROR:

print("Something is wrong with your user name or password")

elif err.errno == errorcode.ER\_BAD\_DB\_ERROR:

print("Database does not exist")

else:

print(err)

else:

cnx.close()

If you have lots of connection arguments, it's best to keep them in a dictionary and use the \*\* operator:

import mysql.connector

config = {

'user': 'scott',

'password': 'password',

'host': '127.0.0.1',

'database': 'employees',

'raise\_on\_warnings': True,

}

cnx = mysql.connector.connect(\*\*config)

cnx.close()

import mysql.connector

from mysql.connector import errorcode

DB\_NAME = 'employees'

TABLES = {}

TABLES['employees'] = (

"CREATE TABLE `employees` ("

" `emp\_no` int(11) NOT NULL AUTO\_INCREMENT,"

" `birth\_date` date NOT NULL,"

" `first\_name` varchar(14) NOT NULL,"

" `last\_name` varchar(16) NOT NULL,"

" `gender` enum('M','F') NOT NULL,"

" `hire\_date` date NOT NULL,"

" PRIMARY KEY (`emp\_no`)"

") ENGINE=InnoDB")

TABLES['departments'] = (

"CREATE TABLE `departments` ("

" `dept\_no` char(4) NOT NULL,"

" `dept\_name` varchar(40) NOT NULL,"

" PRIMARY KEY (`dept\_no`), UNIQUE KEY `dept\_name` (`dept\_name`)"

") ENGINE=InnoDB")

TABLES['salaries'] = (

"CREATE TABLE `salaries` ("

" `emp\_no` int(11) NOT NULL,"

" `salary` int(11) NOT NULL,"

" `from\_date` date NOT NULL,"

" `to\_date` date NOT NULL,"

" PRIMARY KEY (`emp\_no`,`from\_date`), KEY `emp\_no` (`emp\_no`),"

" CONSTRAINT `salaries\_ibfk\_1` FOREIGN KEY (`emp\_no`) "

" REFERENCES `employees` (`emp\_no`) ON DELETE CASCADE"

") ENGINE=InnoDB")

TABLES['dept\_emp'] = (

"CREATE TABLE `dept\_emp` ("

" `emp\_no` int(11) NOT NULL,"

" `dept\_no` char(4) NOT NULL,"

" `from\_date` date NOT NULL,"

" `to\_date` date NOT NULL,"

" PRIMARY KEY (`emp\_no`,`dept\_no`), KEY `emp\_no` (`emp\_no`),"

" KEY `dept\_no` (`dept\_no`),"

" CONSTRAINT `dept\_emp\_ibfk\_1` FOREIGN KEY (`emp\_no`) "

" REFERENCES `employees` (`emp\_no`) ON DELETE CASCADE,"

" CONSTRAINT `dept\_emp\_ibfk\_2` FOREIGN KEY (`dept\_no`) "

" REFERENCES `departments` (`dept\_no`) ON DELETE CASCADE"

") ENGINE=InnoDB")

TABLES['dept\_manager'] = (

" CREATE TABLE `dept\_manager` ("

" `dept\_no` char(4) NOT NULL,"

" `emp\_no` int(11) NOT NULL,"

" `from\_date` date NOT NULL,"

" `to\_date` date NOT NULL,"

" PRIMARY KEY (`emp\_no`,`dept\_no`),"

" KEY `emp\_no` (`emp\_no`),"

" KEY `dept\_no` (`dept\_no`),"

" CONSTRAINT `dept\_manager\_ibfk\_1` FOREIGN KEY (`emp\_no`) "

" REFERENCES `employees` (`emp\_no`) ON DELETE CASCADE,"

" CONSTRAINT `dept\_manager\_ibfk\_2` FOREIGN KEY (`dept\_no`) "

" REFERENCES `departments` (`dept\_no`) ON DELETE CASCADE"

") ENGINE=InnoDB")

TABLES['titles'] = (

"CREATE TABLE `titles` ("

" `emp\_no` int(11) NOT NULL,"

" `title` varchar(50) NOT NULL,"

" `from\_date` date NOT NULL,"

" `to\_date` date DEFAULT NULL,"

" PRIMARY KEY (`emp\_no`,`title`,`from\_date`), KEY `emp\_no` (`emp\_no`),"

" CONSTRAINT `titles\_ibfk\_1` FOREIGN KEY (`emp\_no`)"

" REFERENCES `employees` (`emp\_no`) ON DELETE CASCADE"

") ENGINE=InnoDB")

The preceding code shows how we are storing the CREATE statements in a Python dictionary called TABLES. We also define the database in a global variable called DB\_NAME, which enables you to easily use a different schema.

cnx = mysql.connector.connect(user='scott')

cursor = cnx.cursor()

A single MySQL server can manage multiple [databases](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_database). Typically, you specify the database to switch to when connecting to the MySQL server. This example does not connect to the database upon connection, so that it can make sure the database exists, and create it if not:

def create\_database(cursor):

try:

cursor.execute(

"CREATE DATABASE {} DEFAULT CHARACTER SET 'utf8'".format(DB\_NAME))

except mysql.connector.Error as err:

print("Failed creating database: {}".format(err))

exit(1)

try:

cnx.database = DB\_NAME

except mysql.connector.Error as err:

if err.errno == errorcode.ER\_BAD\_DB\_ERROR:

create\_database(cursor)

cnx.database = DB\_NAME

else:

print(err)

exit(1)

We first try to change to a particular database using the database property of the connection object cnx. If there is an error, we examine the error number to check if the database does not exist. If so, we call the create\_database function to create it for us.

On any other error, the application exits and displays the error message.

After we successfully create or change to the target database, we create the tables by iterating over the items of the TABLES dictionary:

for name, ddl in TABLES.iteritems():

try:

print("Creating table {}: ".format(name), end='')

cursor.execute(ddl)

except mysql.connector.Error as err:

if err.errno == errorcode.ER\_TABLE\_EXISTS\_ERROR:

print("already exists.")

else:

print(err.msg)

else:

print("OK")

cursor.close()

cnx.close()

To handle the error when the table already exists, we notify the user that it was already there. Other errors are printed, but we continue creating tables. (The example shows how to handle the “table already exists” condition for illustration purposes. In a real application, we would typically avoid the error condition entirely by using the IF NOT EXISTS clause of the [CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html) statement.)

The output would be something like this:

Creating table employees: already exists.

Creating table salaries: already exists.

Creating table titles: OK

Creating table departments: already exists.

Creating table dept\_manager: already exists.

Creating table dept\_emp: already exists.

## Inserting Data Using Connector/Python

from \_\_future\_\_ import print\_function

from datetime import date, datetime, timedelta

import mysql.connector

cnx = mysql.connector.connect(user='scott', database='employees')

cursor = cnx.cursor()

tomorrow = datetime.now().date() + timedelta(days=1)

add\_employee = ("INSERT INTO employees "

"(first\_name, last\_name, hire\_date, gender, birth\_date) "

"VALUES (%s, %s, %s, %s, %s)")

add\_salary = ("INSERT INTO salaries "

"(emp\_no, salary, from\_date, to\_date) "

"VALUES (%(emp\_no)s, %(salary)s, %(from\_date)s, %(to\_date)s)")

data\_employee = ('Geert', 'Vanderkelen', tomorrow, 'M', date(1977, 6, 14))

# Insert new employee

cursor.execute(add\_employee, data\_employee)

emp\_no = cursor.lastrowid

# Insert salary information

data\_salary = {

'emp\_no': emp\_no,

'salary': 50000,

'from\_date': tomorrow,

'to\_date': date(9999, 1, 1),

}

cursor.execute(add\_salary, data\_salary)

# Make sure data is committed to the database

cnx.commit()

cursor.close()

cnx.close()

## Querying Data Using Connector/Python

The following example shows how to [query](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_query) data using a cursor created using the connection's [cursor()](https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqlconnection-cursor.html) method. The data returned is formatted and printed on the console.

The task is to select all employees hired in the year 1999 and print their names and hire dates to the console.

import datetime

import mysql.connector

cnx = mysql.connector.connect(user='scott', database='employees')

cursor = cnx.cursor()

query = ("SELECT first\_name, last\_name, hire\_date FROM employees "

"WHERE hire\_date BETWEEN %s AND %s")

hire\_start = datetime.date(1999, 1, 1)

hire\_end = datetime.date(1999, 12, 31)

cursor.execute(query, (hire\_start, hire\_end))

for (first\_name, last\_name, hire\_date) in cursor:

print("{}, {} was hired on {:%d %b %Y}".format(

last\_name, first\_name, hire\_date))

cursor.close()

cnx.close()