MySQL 存储

一、MySQL 的安装的使用教程

2023 年 MySQL 8.0 安装配置 最简易（保姆级） _mysql安装-CSDN博客

二、数据库的操作

1、连接数据库：

```
import pymysql

db = pymysql.connect(host='localhost', user='root', password='190901sjnh',
port=3306)
cursor = db.cursor()                                    # 创建游标对象 cursor，用于
执行 SQL 查询和操作。
cursor.execute("CREATE DATABASE students DEFAULT CHARACTER SET utf8mb4")
 # 创建一个名为 "students" 的数据库，并使用 utf-8 编码
db.close()                                              # 关闭数据库链接
```

2、创建表：

```
import pymysql

db = pymysql.connect(host='localhost', user='root', password='190901sjnh',
port=3306, db='spiders')
cursor = db.cursor()

# 创建表 students ：包含三个字段，id、name、age
sql = 'CREATE TABLE IF NOT EXISTS students (id VARCHAR(255) NOT NULL, name
VARCHAR(255) NOT NULL, age INT NOT NULL, PRIMARY KEY (id))'
cursor.execute(sql)                     # 执行 SQL 语句
db.close()
```

创建表的时候指定了3个字段：

字段名	含义	类型
id	学号	varchar，长为255字节
name	姓名	varchar，长为255字节
age	年龄	int

3、插入数据（静态插入）：

我们将已经获取的数据存入数据库，其步骤如下：

- 连接数据库：调用 `connect` 方法连接数据库；
- 创建 `cursor` 对象；
- 构造 `sql` 语句；
- 调用 `execute` 方法执行 `sql` 语句，并**调用 commit 方法**将**事务**提交至数据库；
- 若发生异常，则调用 `rollback` 执行**数据回滚**。

```
import pymysql

id = '20120001'
user = 'Bob'
age = 20

db = pymysql.connect(host='localhost', user='root', password='190901sjnh',
port=3306, db='spiders')
cursor = db.cursor()

# 这种写法可以有效避免字符串拼接的麻烦和引号冲突的问题
sql = 'INSERT INTO students(id, name, age) values(%s, %s, %s)'

try:
    cursor.execute(sql, (id, user, age))     # 通过元组的方式进行数据传递
    db.commit()                              # 在执行写操作（如 INSERT、
UPDATE、DELETE）后，都需要调用 commit() 方法将事务提交，才能使修改生效
    print('Insert successfully')
except:
    db.rollback()                            # 若执行失败，则调用 rollback 执
行数据回滚
db.close()
```

4、插入数据（动态插入）：

在很多情况下，我们希望插入方法无需改动，将其作为通用的方法，只需改变传入的作为参数的字典即可。我们实现通用的插入方法如下：

```
import pymysql

data = {
    'id': '20120002',
    'name': 'John',
    'age': 20
}

table = 'students'
keys = ', '.join(data.keys())                # 将字典 data 中的键(即数据库表的
列名)以逗号分隔的形式拼接成一个字符串。
```

```
values = ', '.join(['%s'] * len(data))      # 生成一个包含多个 %s 占位符的字符串，用于构建 SQL 语句中的值部分

db = pymysql.connect(host='localhost', user='root', password='190901sjnh', port=3306, db='spiders')
cursor = db.cursor()
sql = 'INSERT INTO {table}({keys}) VALUES ({values})'.format(table=table, keys=keys, values=values)

try:
    if cursor.execute(sql, tuple(data.values())):       # tuple(data.values()) 的作用是将字典 data 中的值转换为元组。
        print('Successful')
        db.commit()
except Exception as e:
    print('Failed', e)
    db.rollback()
db.close()
```

5、更新数据：

更新数据的原则：

- 若数据已经在数据库中，则插入数据；
- 否则，更新数据库即可，无需执行插入操作。

```
import pymysql

data = {
    'id': '20120003',
    'name': 'Jack',
    'age': 21
}

table = 'students'
keys = ', '.join(data.keys())
values = ', '.join(['%s'] * len(data))
update = ','.join(["{key} = %s".format(key=key) for key in data])

db = pymysql.connect(host='localhost', user='root', password='190901sjnh', port=3306, db='spiders')
cursor = db.cursor()

# ON DUPLICATE KEY UPDATE: 若主键已经存在，就执行更新操作
# 完整的 SQL 语句为:
#     INSERT INTO students(id, name, age) VALUES (%s, %s, %s) ON DUPLICATE KEY UPDATE id = %s, name = %s, age = %s
sql = 'INSERT INTO {table}({keys}) VALUES ({values}) ON DUPLICATE KEY UPDATE '.format(table=table, keys=keys, values=values)
sql += update
```

```
try:
    if cursor.execute(sql, tuple(data.values())*2):      # 在完整的 SQL 语句
中有6个 %s 语句，因此 execute 方法中第二个参数就需要 *2
        print('Successful')
        db.commit()
except Exception as e:
    print('Failed', e)
    db.rollback()
db.close()
```

6、删除数据：

删除数据的操作，**直接使用 DELETE 语句**即可，只是需要指定要删除的目标表名和删除条件。

```
import pymysql

table = 'students'
condition = 'age > 20'          # 删除条件

db = pymysql.connect(host='localhost', user='root', password='190901sjnh',
port=3306, db='spiders')
cursor = db.cursor()
sql = 'DELETE FROM  {table} WHERE {condition}'.format(table=table,
condition=condition)
try:
    cursor.execute(sql)
    db.commit()
except:
    db.rollback()

db.close()
```

7、查询数据：

```
import pymysql

sql = 'SELECT * FROM students WHERE age >= 20'          # 查询条件

db = pymysql.connect(host='localhost', user='root', password='190901sjnh',
port=3306, db='spiders')
cursor = db.cursor()
try:
    cursor.execute(sql)                                 # 执行查询语句
    print('Count:', cursor.rowcount)                    # 调用 rowcount 属
性，获取查询结果的条数

    one = cursor.fetchone()                             # 调用 fetchone 方
法，获取结果的第一条数据，并以元组形式呈现
    print('One:', one)
```

```
    results = cursor.fetchall()                                    # 调用 fetchall 方法，获取所有结果，并以二重元组形式呈现
    print('Results:', results)
    print('Results Type:', type(results))

    for row in results:
        print(row)
except:
    print('Error')
```

这里需要注意，当执行 `results = cursor.fetchall()` 的时候，获取的不是所有数据，而是**从偏移指针开始之后的所有数据。**由于在这一语句之前，我们执行了 `one = cursor.fetchone()` 使得偏移指针指向了第二条数据，因此，上述程序中，执行 `results = cursor.fetchall()` 的结果将有 `count-1` 条数据。

由于 fetchall 方法将结果以元组的形式返回，如果数据量很大，那么占用的开销就会很高。因此，我们通常使用 `while 循环+fetchone 方法` 的组合来实现所有数据的获取：

```
import pymysql

sql = 'SELECT * FROM students WHERE age >= 20'
db = pymysql.connect(host='localhost', user='root', password='190901sjnh', port=3306, db='spiders')
cursor = db.cursor()

try:
    cursor.execute(sql)
    print('Count:', cursor.rowcount)
    row = cursor.fetchone()                    # 通过 while 循环 + fetchone 方法的组合来获取所有数据
    while row:
        print('Row:', row)
        row = cursor.fetchone()
except:
    print('Error')
```

三、MySQL 操作的相关概念

1、sql 语句构造的技巧：

在上述的例子中，我们构造 sql 语句时，通常使用了 格式化符%s 的思想，而非字符串拼接的方法。

- 格式化符%s：

  ```
  sql = 'INSERT INTO students(id, name, age) values(%s, %s, %s)'
  ```

- 字符串拼接：

```
sql = 'INSERT INTO students(id, name, age) values(' + id + ', ' + name
+ ', ' + age + ')'
```

由于使用了格式化符的方法，**有几个 value 就写几个 %s**，我们只需要在 execute 方法的第一个参数传入该 sql 语句，value 值用统一的元组传过来即可。这样既可以避免字符串拼接的麻烦，也可以避免引号冲突的问题。

```
sql = 'INSERT INTO {table}({keys}) VALUES ({values})'.format(table=table,
keys=keys, values=values)
cursor.execute(sql, tuple(data.values()))
```

2、commit 方法：

commit 方法才是真正将语句提交到数据库执行的方法，对于数据的插入、删除、更新操作，都需要在执行 execute 方法之后调用 commit 方法，事务才能生效。

3、事务：

事务机制能够保证**数据的一致性**，即：若一件事要么已经发生完整了，要么完全没有发生。例如：插入一条数据，不会存在插入一半的情况，要么完全插入，要么都不插入，这就是**事务的原子性。**事务还有其他3个属性——**一致性、隔离性、持久性**，这4个属性通常称为ACID 特性。

属性	解释
原子性（atomicity）	事务是一个**不可分割**的工作单位，事务中的操作要么都做、要么不做
一致性（consistency）	事务必须使数据库从一个一致性状态变到另一个一致性状态，一致性和原子性是密切相关的
隔离性（isolation）	**一个事务的执行不能被其他事务干扰**，即：一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能干扰
持久性（durability）	持续性，又称永久性，指**一个事务一旦提交，它对数据库中数据做的改变就应该是永久性的。**接下来的其他操作或故障不应该对数据有任何影响

4、数据回滚：

数据回滚是数据库事务处理中的一种重要机制，它用于撤销之前已经执行但尚未提交的事务操作，**将数据库恢复到之前的状态。**