数据库操作

接口自动化测试课程

黑马程序员
www.itheima.com
传智教育旗下
高端IT教育品牌

学习目标

Learning Objectives

1. 知道数据库操作的应用场景
2. 掌握使用PyMySQL对数据库的增、删、改、查
3. 数据库工具类封装

目录

Contents

01 数据库操作应用场景

学习目标

1. 知道数据库操作的应用场景

数据库操作应用场景

接口自动化测试数据库主要应用场景?

接口自动化测试

1. 测试数据校验
2. 测试数据构造

接口自动化测试

与**数据库交互场景**有哪些?

数据库操作应用场景

数据库操作的应用场景有哪些?

目录

Contents

数据库操作实现

02

- 安装PyMySQL
- 操作数据库的基本流程
- 数据库操作实现

学习目标

1. 掌握操作数据库的基本流程
2. 掌握使用PyMySQL对数据库的增、删、改、查

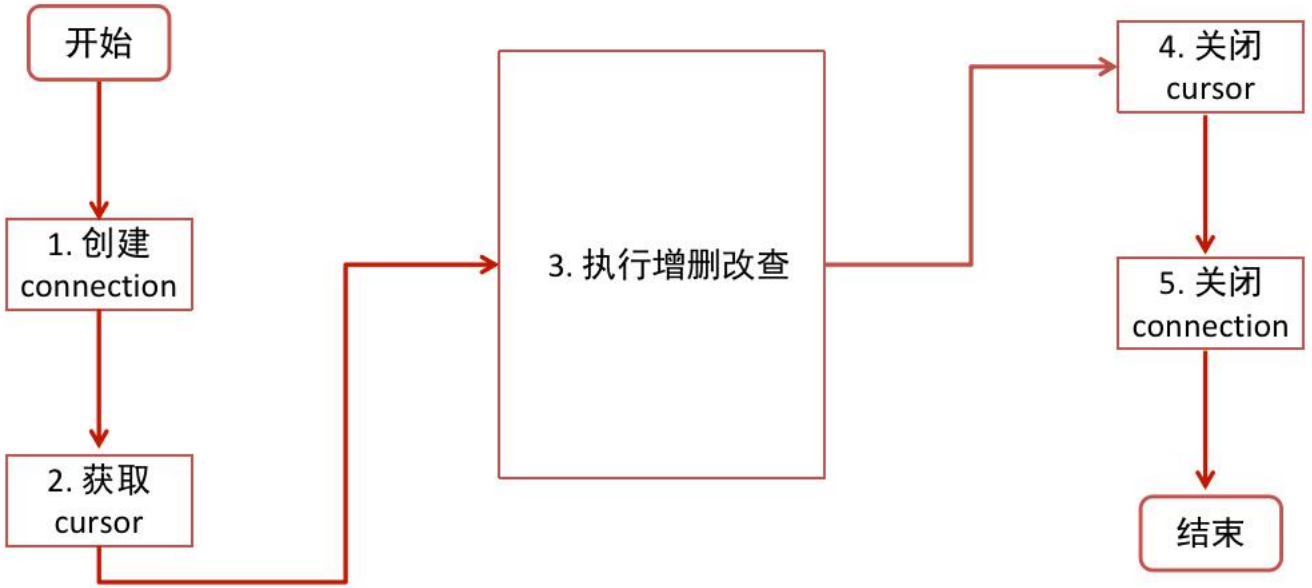
数据库操作实现

PyMySQL：Python3.x 版本中连接 MySQL 服务器的一个库

安装PyMySQL：**pip install PyMySQL**

数据库操作实现

操作数据库的基本流程



代码实现步骤：

1. 导包
2. 创建数据库连接
3. 获取游标对象
4. 执行操作
5. 关闭游标对象
6. 关闭数据库连接

数据库操作实现

测试数据库连接信息

- **host**: 211.103.136.244
- **port**: 7061
- **user**: student
- **password**: iHRM_student_2021
- **database**: test_db

数据库操作–连接数据库

```
conn = pymysql.connect(host=None, user=None, password="",
database=None, port=0, charset="")
```

参数说明：

- host: 数据库服务器地址
- user: 登录用户名
- password: 密码
- database: 要连接的数据库名称
- port: 数据库连接端口(默认值: 3306)
- charset: 字符集 (设置为: utf8)

数据库操作–获取游标对象

```
cursor = conn.cursor()
```

说明：

- 调用数据库连接对象的cursor()方法获取游标对象

数据库操作–执行SQL语句

```
cursor.execute(sql)
```

说明:

- 调用游标对象的execute()方法来执行SQL语句
- sql: 要执行的sql语句

数据库操作–释放资源

```
# 关闭游标对象
cursor.close()

# 关闭数据库连接
conn.close()
```

说明：

- 调用数据库连接对象、游标对象的close()方法来释放资源

案例

入门案例：查询数据库服务器版本信息

需求：

- 连接到数据库（host:211.103.136.244 port：7061 user:student password:iHRM_student_2021 database:test_db）
- 获取数据库服务器版本信息

分析：

- 如何获取数据库的连接?
- 获取数据库服务器版本信息的SQL语句是什么?
  - sql语句：select version()
- 如何执行SQL语句?
  - **cursor.execute("select version()")**
  - **获取查询结果：cursor.fetchone()**
- 如何释放资源?

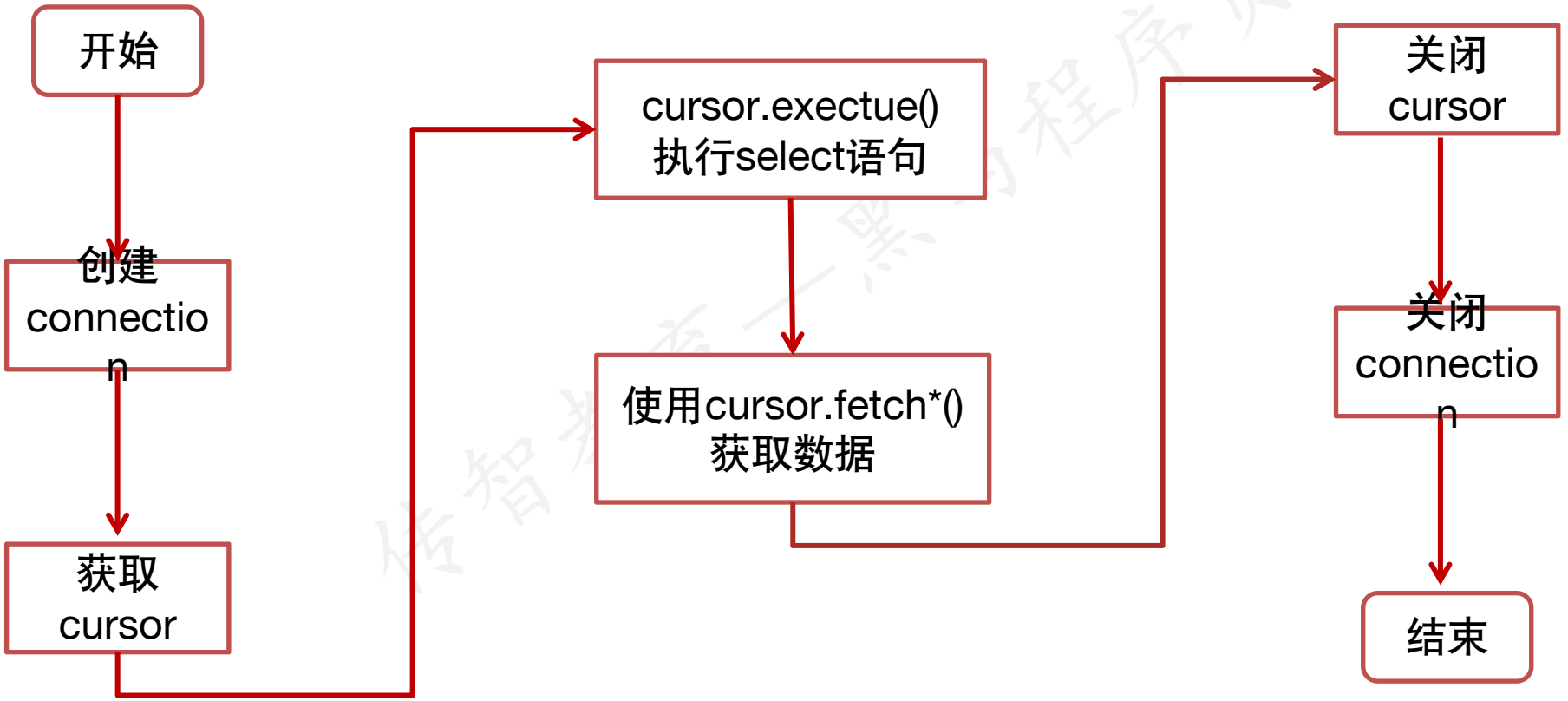
案例

入门案例：查询数据库服务器版本信息

```
# 导包
import pymysql
# 创建数据库连接
conn = pymysql.connect(host="localhost", user="root", password="root", database="books")
# 创建游标对象
cursor = conn.cursor()
# 执行操作：查询数据库版本信息
cursor.execute("select version()")
# 获取查询结果
result = cursor.fetchone()
print("result=", result)
# 关闭游标对象
cursor.close()
# 关闭数据库连接
conn.close()
```

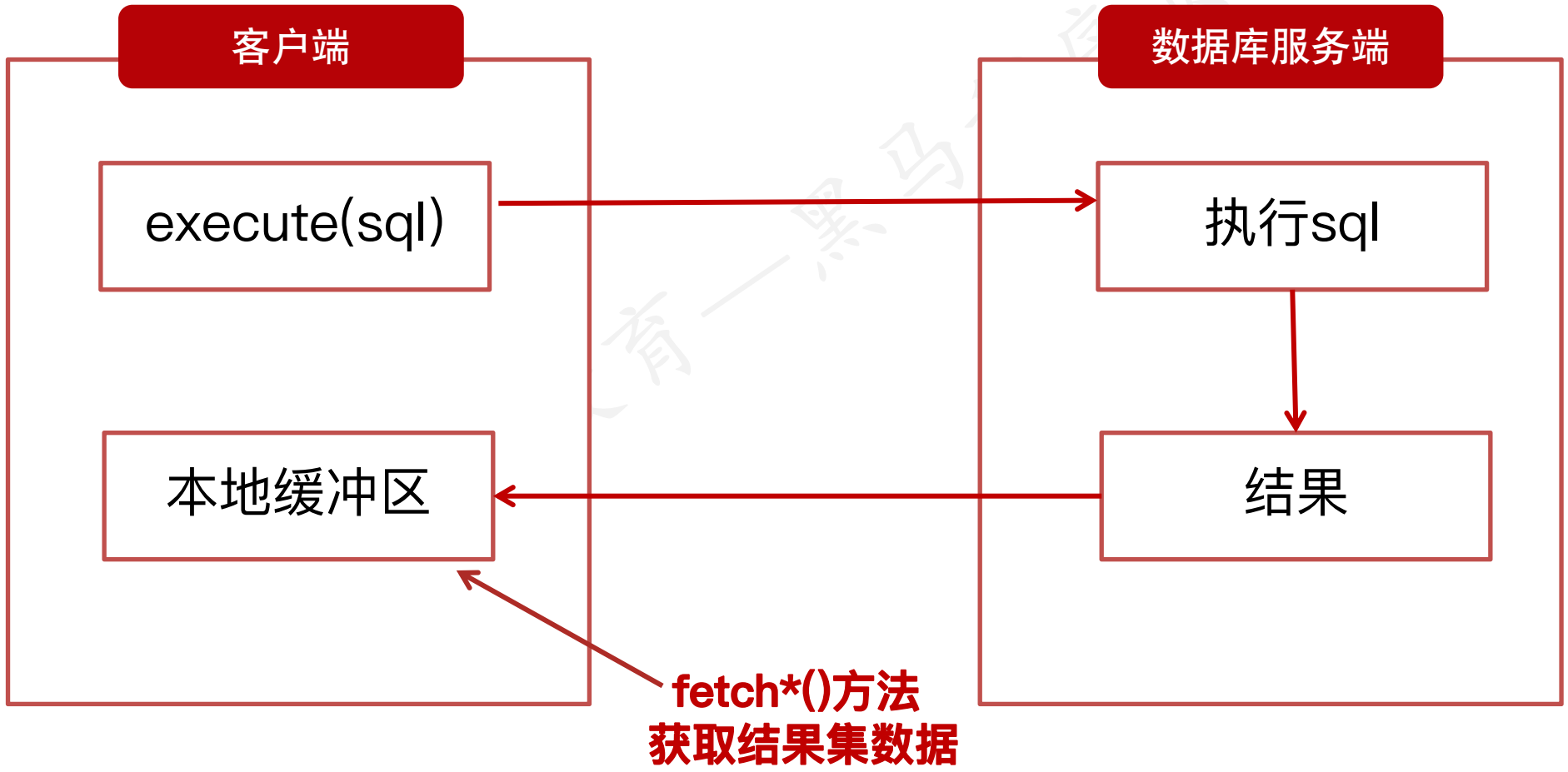
数据库操作实现

数据库查询—select



数据库操作

cursor对象的execute方法：执行sql并返回结果到客户端



获取查询结果集

cusor对象的fetch*()方法

常见方法

- **fetchone()**

  获取结果集中的下一行数据 → 返回值：元组数据

- **fetchmany(size)**

  获取结果集中的指定size行数据 → 返回值：((),(),...)

- **fetchall()**

  获取结果集中所有行数据 → 返回值：((),(),...)

获取查询结果集案例一

案例 查询books表的指定信息

数据库地址：

host: 211.103.136.244, port: 7061,

user: student, password: iHRM_student_2021, database: test_db

要求：

①：查询t_book表第一条数据

②：获取t_book前两条数据

③：查询t_book表的全部数据

分析：

1）cursor.fetchone()

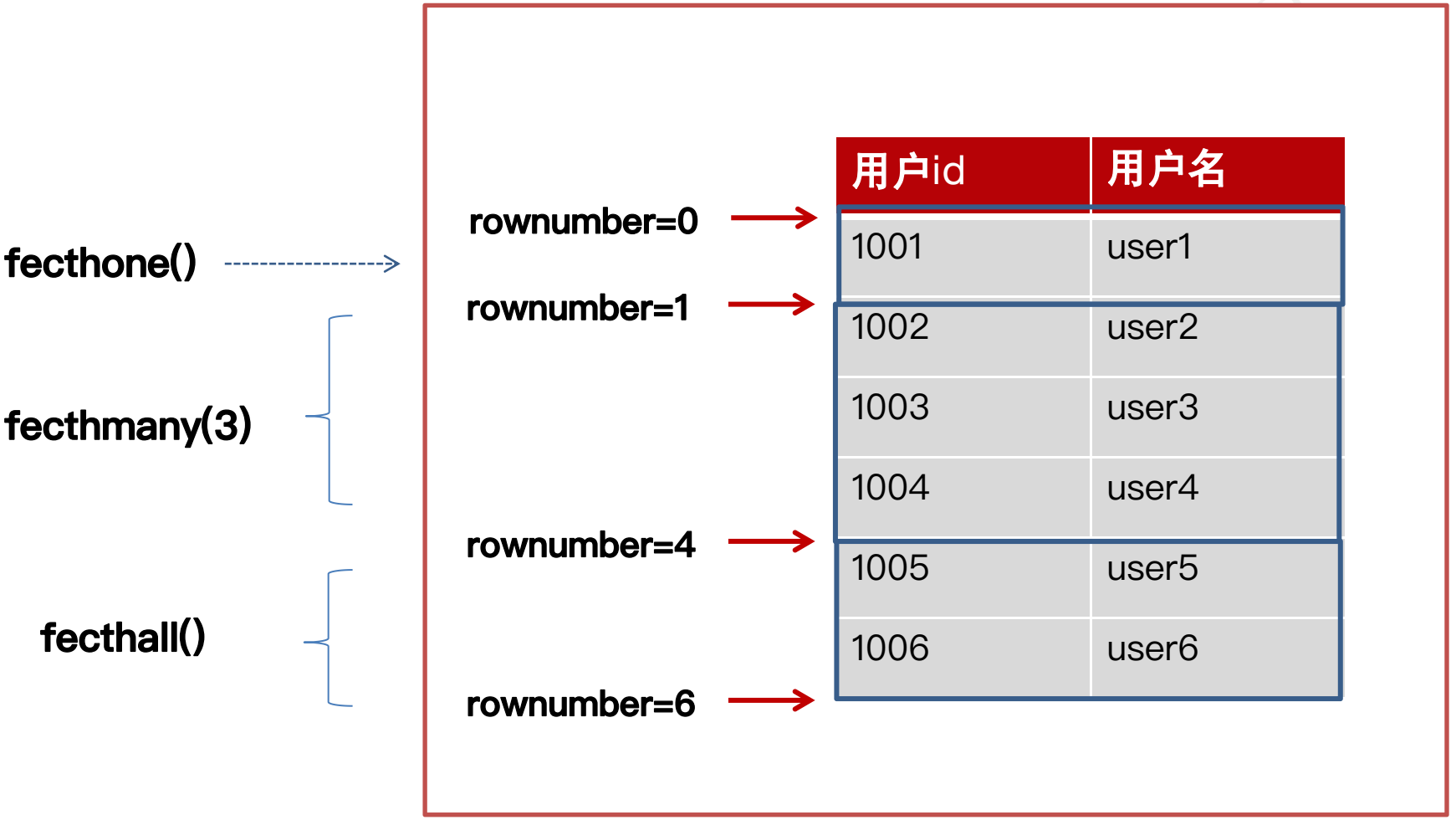
2）cursor.fetchmany(2)

3）cursor.fetall()

fetch*()方法原理

rownumber属性：**标识游标的当前位置**（默认初始位置从0开始）

fetch*()方法是基于游标当前位置**逐行向下**获取数据



获取查询结果集案例二

案例 查询books表的指定信息

要求：

①：查询t_book表的全部字段数据

②：获取查询结果集的第一条数据

③：获取查询结果集的第3条和第4条数据

④：获取全部的查询结果集及总记录数

分析：

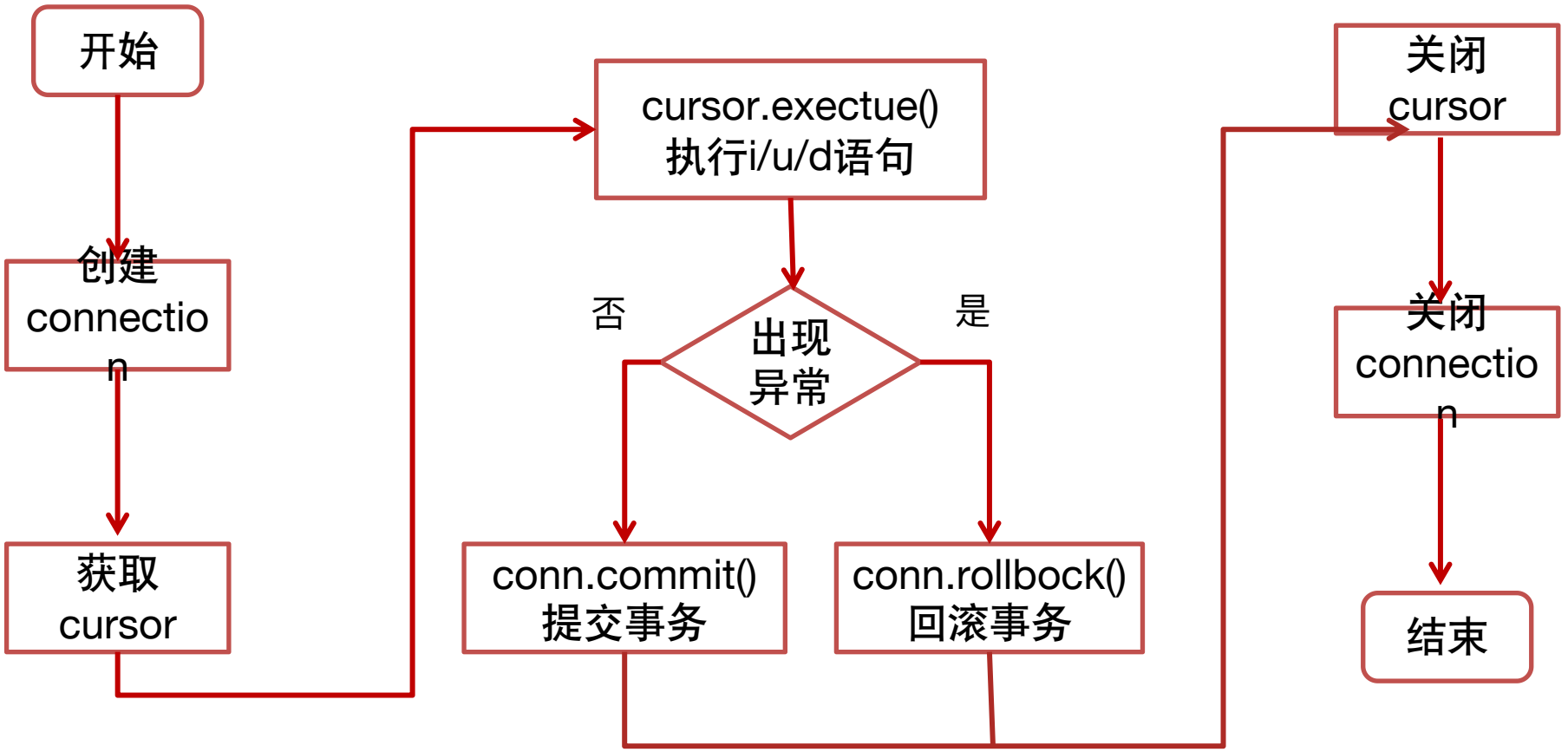
1) select * from t_book

2) cursor.rownumber= 0　　cursor.fetchone()

3) cursor.rownumber= 2　　cursor.fetchmany(2)

4) cursor.rownumber= 0　　cursor.fetchall()

数据库更新—insert/update/delete演示



数据库查询案例

案例 更新t_book表的指定信息

数据库地址：

host: 211.103.136.244, port: 7061,

user: student, password: iHRM_student_2021, database: test_db

要求单独实现：

①：新增一条图书数据（id:5 title:西游记 pub_date:1986-01-01 ）

②：把图书名称为‘西游记’的阅读量加一

③：删除名称为‘西游记’的图书

数据库操作实现

案例

```
import pymysql
import traceback
conn = pymysql.connect(host="211.103.136.244", port=7061, user="student",
                       password="iHRM_student_2021", database="test_db", charset="utf8")
cursor = conn.cursor()
try:
    # sql_i = "insert into t_book(id, title, pub_date) values(5, '西游记', '1986-01-01')"
    # sql_u = "update t_book set `read` = `read` + 1 where title='西游记'"
    sql_d = "delete from t_book where title='西游记'"
    cursor.execute(sql_d)
    conn.commit()
except Exception as e:
    conn.rollback()
    raise e
finally:
    cursor.close()
    conn.close()
```

数据库工具类封装

学习目标

1. 掌握数据库工具类封装

数据库工具类封装

封装实现

- 类方法：get_conn()　// 获取数据库连接
- 类方法：close_conn()　// 关闭数据库连接
- 类方法：get_one(sql)　// 查询一条数据
- 类方法：uid_db(sql)　// 执行数据库增删改

要求：定义一个DBUtil工具类，对外提供以上方法

1. 操作数据库的应用场景有哪些?
2. 能否使用PyMySQL对数据库的增、删、改、查?
3. 数据库工具类封装

传智教育旗下高端IT教育品牌