数据管理技术大作业测试样例

建表和注入数据测试

执行脚本 `insert.sh` ，能够正确运行代码，在数据库中成功建表，并且能够从csv文件中读取数据并插入到数据库表中，同学们可以连接数据库查询或者用DBeacer等工具可视化查看验证。

Restful接口测试

首先执行脚本 `start.sh` ，程序正常运行

```
('8.0.28-231002',)
 * Serving Flask app 'RESTful'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5555
 * Running on http://192.168.1.252:5555
Press CTRL+C to quit
```

插入接口测试

以departments表为例，插入一个新部门

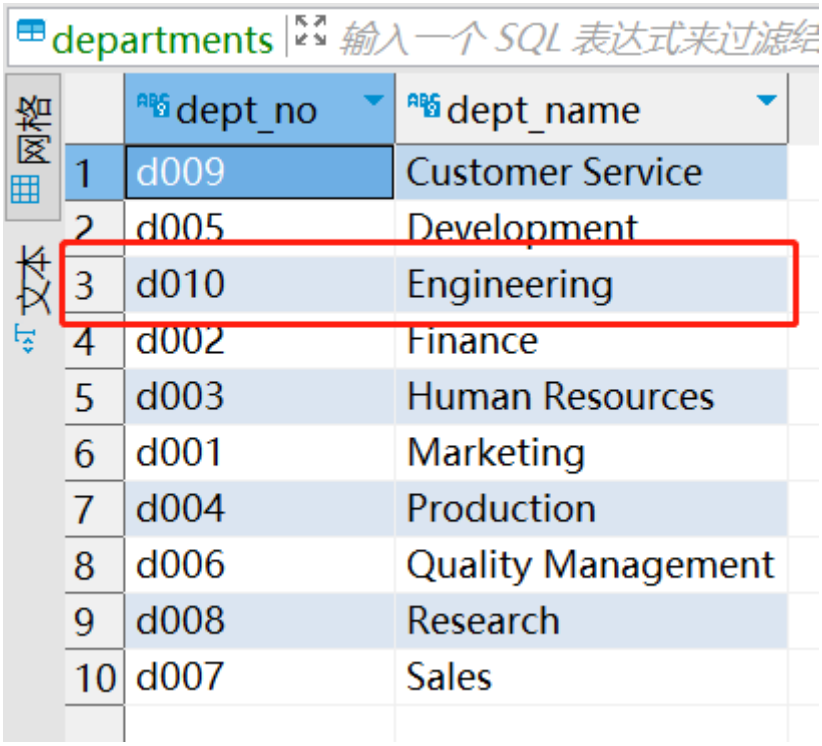
- 接口地址：**http://127.0.0.1:5555/api/v1/departments**
- 接口方法：**POST**
- body：JSON格式，包含想要插入的部门信息

```
{
    "rows": [
        {"dept_no": "d010", "dept_name": "Engineering"}
    ]
}
```

- 预期结果

服务器返回一个响应确认数据已成功插入，并且在数据库的 `departments` 表中实际存在了新添加的部门记录。

```
{
    "message": "rows inserted successfully"
}
```



更新接口测试

以departments表为例，将刚刚插入的 `dept_no` 为 `d010` 的 `dept_name` 改为 `Operations`

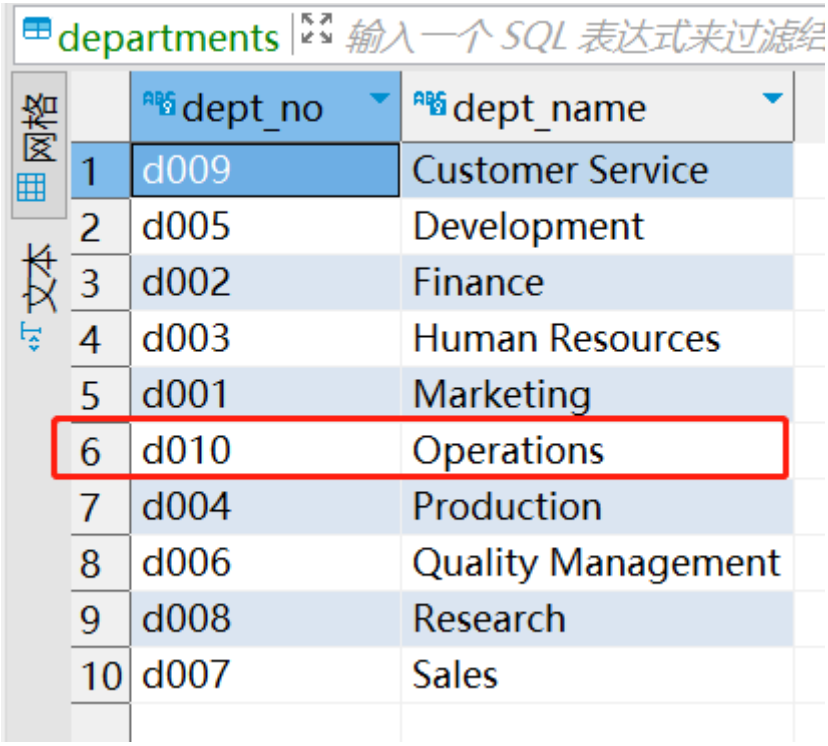
- 接口地址：**http://127.0.0.1:5555/api/v1/departments**
- 接口方法：**PUT**
- body：JSON格式，包含想要更新的部门信息，**保证表的主键字段都在请求体字典的最前面**

```
{
    "dept_no": "d010",
    "dept_name": "Operations"
}
```

- 预期结果

服务器返回一个响应确认数据已成功更新，并且在数据库的 `departments` 表中实际更新了相应的部门信息。

```
{
    "message": " row updated successfully"
}
```



	dept_no	dept_name
1	d009	Customer Service
2	d005	Development
3	d002	Finance
4	d003	Human Resources
5	d001	Marketing
6	d010	Operations
7	d004	Production
8	d006	Quality Management
9	d008	Research
10	d007	Sales

删除接口测试

以dept_emp表为例，删除 `emp_no` 为 `10001` 且 `dept_no` 为 `d005` 的数据

- 接口地址：**http://127.0.0.1:5555/api/v1/dept_emp/10001/d005**
- 接口方法：**DELETE**
- body：**None**
- 预期结果：

服务器返回一个响应确认数据已成功删除，并且在数据库的 `dept_emp` 表中实际删除了相应的信息。

```
{
    "message": "row deleted successfully"
}
```

Flask路由定义 `@app.route('/api/v1/<table_name>/<path:args>', methods=['DELETE'])` 中，`<path:args>` 是一个URL参数模式。它允许你捕获URL中的一部分作为路由参数。

这里的 `args` 是一个特殊的参数类型，它使用了 `path` 转换器（`PathConverter`），这意味着它会尝试匹配URL中的任何路径组成部分，并将其作为一个Python字符串返回。例如，如果你的URL是 `/api/v1/dept_emp/10001/d005`，那么 `args` 将捕获 "10001/d005" 并将其赋值给变量 `args`。

在删除数据的操作中，这个 `args` 参数用于确定要删除的特定记录的标识。根据不同表格的例子：

- 对于 `departments` 表， `args` 将被解析为部门号 `dept_no`。
- 对于 `employees` 表， `args` 将被解析为员工号 `emp_no`。
- 对于 `dept_emp` 表， `args` 将捕获两个部分：第一个作为员工号 `emp_no`，第二个作为部门号 `dept_no`。
- 对于 `dept_manager` 表，同样捕获两个部分：员工号 `emp_no` 和部门号 `dept_no`。
- 对于 `titles` 表，将捕获三个部分：员工号 `emp_no`，职位 `title`，以及从日期 `from_date`。

举个例子，如果要删除departments表中 `dept_no` 为 `d010` 的数据，则相应的接口地址为 **http://127.0.0.1:5555/api/v1/departments/d010**，

查询接口测试

查询接口要求实现两种方式，第一种和DELETE相似，以titles表为例

- 接口地址：**http://127.0.0.1:5555/api/v1/titles/10002/Staff/1996-08-03**
- 接口方法：**GET**
- body：**None**
- 预期结果：

服务器返回正确的查询结果，JSON格式

```
[
    [
        10002,
        "Staff",
        "Sat, 03 Aug 1996 00:00:00 GMT",
        "Fri, 01 Jan 9999 00:00:00 GMT"
    ]
]
```

第二种情况为指定一种过滤条件，对应于SQL中的`where a = b` 。**Hint**：用`request.args`获取`RequestArgumentDict`对象，它代表了HTTP请求中的查询参数，再调用`.to_dict()`方法，你将`RequestArgumentDict`转换为一个标准的Python字典，然后可以在SQL语句中使用这些值来过滤数据库中的记录。

- 接口地址：**http://127.0.0.1:5555/api/v1/titles?to_date=1995-12-01**

对于Apifox来说，Query参数在请求参数中填写，如下图所示

- 接口方法：**GET**
- body：**None**
- 预期结果：

服务器返回正确的查询结果，JSON格式

```
[
    [
        10004,
        "Engineer",
        "Mon, 01 Dec 1986 00:00:00 GMT",
        "Fri, 01 Dec 1995 00:00:00 GMT"
    ],
    [
        19397,
        "Staff",
        "Tue, 01 Dec 1987 00:00:00 GMT",
        "Fri, 01 Dec 1995 00:00:00 GMT"
    ],
    ...
```

```
]
```

触发器测试

如果触发器设置正确的话，在插入数据之后 `dept_manager_title` 表中应该有相应的数据了，同学们可以从 `dept_manager` 表中删除一条记录，验证 `dept_manager_title` 表中是否会删除对应的记录；