《数据库原理与应用》实验报告

实验名称	C/S结构的数据库编程	实验序号	2
实验日期	Tue. Nov. 15 2022	实验人	苏靖博

1、 实验目的

学会通过 React Native 组件访问数据库, 熟悉使用 React Native 进行 C/S 结构的数据库应用程序的设计，通过 React Native 接口对数据库进行操作.

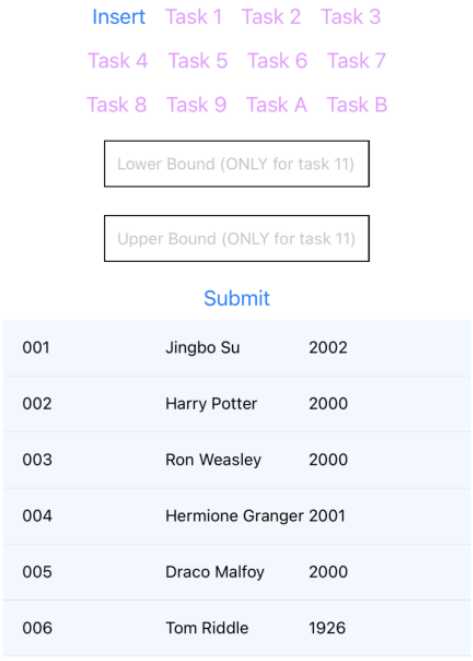
二、实验任务

通过编程工具（VS2005或者以上版本）编写访问数据库的应用程序，编程序设计良好的一个人机交互界面, 列出实验一中的查询，将SQL语句嵌入VS2005中完成“实验一”中的11个语句的结果显示。也可选择其他编程工具开发.

项目链接🔗https://github.com/Sue217/NCUT/tree/main/DataBase/labs/0x03

3、 关键代码与执行结果截图

（1） 求全体学生的学号、姓名和出生年份。



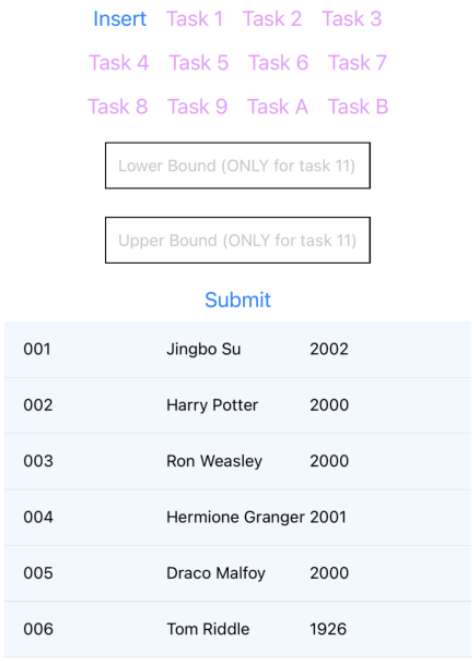
- **client**

```
<Button title="Task 1" color="#f194ff" onPress={() =>
{setButton(1);}}/>
case 1:
  return <Table1 key={val.sname} val={val} />;
```

- **server**

```
app.get("/api/get/task1", (req, res) => {
  const sqlSelect = tasks(1);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

（2） 求每个系的学生总人数并按降序排列。



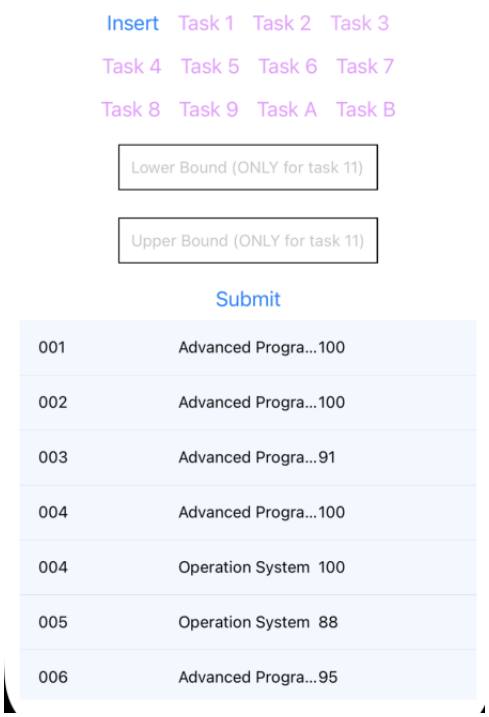
- **client**

```
<Button title="Task 2" color="#f194ff" onPress={() =>
{setButton(2);}}/>
case 2:
  return <Table2 key={val.sdept} val={val} />;
```

- **server**

```
app.get("/api/get/task2", (req, res) => {
  const sqlSelect = tasks(2);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

(3) 求选修了课程号为**002**或**003**的学生的学号、课程名和成绩。



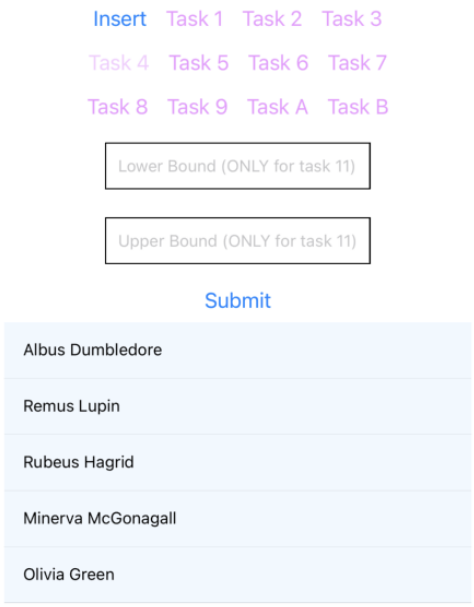
- **client**

```
<Button title="Task 3" color="#f194ff" onPress={() =>
{setButton(3);}}/>
case 3:
  return <Table3 key={[val.cname][val.sno]} val={val} />;
```

- **server**

```
app.get("/api/get/task3", (req, res) => {
  const sqlSelect = tasks(3);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

(4) 检索选修某课程的学生人数多于**3**人的教师姓名。



- **client**

```
<Button title="Task 4" color="#f194ff" onPress={() =>
{setButton(4);}}/>
case 4:
  return <Table4 key={val.tname} val={val} />;
```

- **server**

```
app.get("/api/get/task4", (req, res) => {
  const sqlSelect = tasks(4);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

(5) 查询所有未选课程的学生姓名和所在系。

— NULL —

- **client**

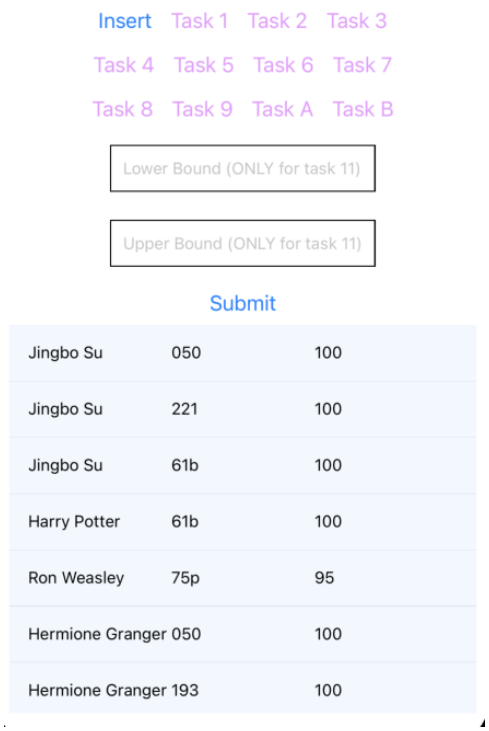
```
<Button title="Task 5" color="#f194ff" onPress={() =>
{setButton(5);}}/>
case 5:
  return <Table5 key={val.sname} val={val} />;
```

- **server**

```
app.get("/api/get/task5", (req, res) => {
  const sqlSelect = tasks(5);
  db.query(sqlSelect, (err, result) => {
```

```
        res.send(result);
      });
    });
```

(6) 求每个同学的课程成绩的最高分，查询结果项包括：学生姓名、课程号及最高分。



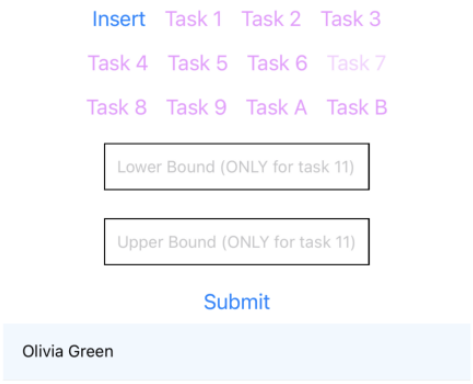
- **client**

```
<Button title="Task 6" color="#f194ff" onPress={() =>
{setButton(6);}}/>
case 6:
  return <Table6 key={[val.cno][val.sname]} val={val} />;
```

- **server**

```
app.get("/api/get/task6", (req, res) => {
  const sqlSelect = tasks(6);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

(7) 求所有讲授数据结构课程的教师姓名



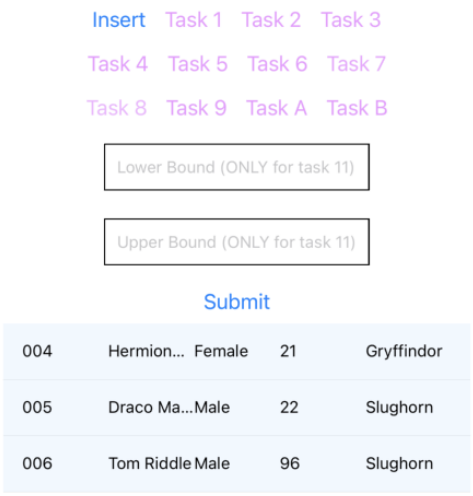
- **client**

```
<Button title="Task 7" color="#f194ff" onPress={() =>
{setButton(7);}}/>
case 7:
  return <Table7 key={val.tname} val={val} />;
```

- **server**

```
app.get("/api/get/task7", (req, res) => {
  const sqlSelect = tasks(7);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

(8) 查询所有选修了 **Severus Snape** 老师的课程的学生信息



- **client**

```
<Button title="Task 8" color="#f194ff" onPress={() =>
```

```
{setButton(8);}}/>
case 8:
  return <Table8 key={[val.sname][val.ssex]} val={val} />;
```

- **server**

```
app.get("/api/get/task8", (req, res) => {
  const sqlSelect = tasks(8);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

(9) 新建一个关系表**S1(SNO,SNAME,SD,SA)**, 其字段类型定义与**Student**表中的相应字段**(SNO,SNAME,SDEPT,SAGE)**的数据类型定义相同。将表**Student**中在计算机系**'CS'**的学生数据插入到表**S1**中。

- **client**

```
<Button title="Task 9" color="#f194ff" onPress={() =>
{setButton(9);}}/>
case 9:
  return <Table9 key={[val.sd][val.sno]} val={val} />;
```

- **server**

```
app.get("/api/get/task9", (req, res) => {
  const sqlSelect = tasks(9);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

(10) 建立计算机系的学生的视图**STUDENT_CS**。利用视图**STUDENT_CS**，将学生的

001号课程的成绩提高**5**分。

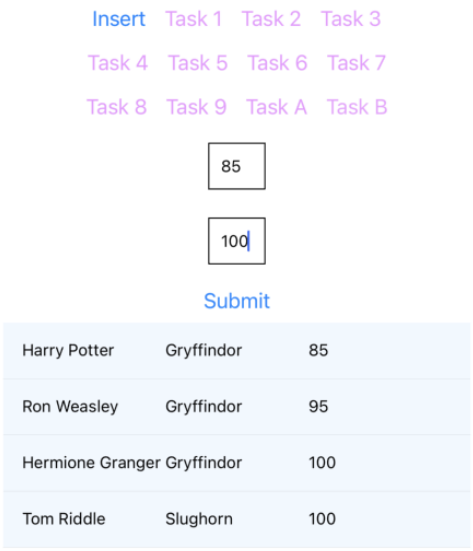
- **client**

```
<Button title="Task A" color="#f194ff" onPress={() =>
{setButton(10);}}/>
case 10:
  return <Table1 key={val.sname} val={val} />;
```

- **server**

```
app.get("/api/get/task10", (req, res) => {
  const sqlSelect = tasks(10);
  db.query(sqlSelect, (err, result) => {
    res.send(result);
  });
});
```

(11) 创建一个带参数的存储过程，将数据结构课程的成绩在**Low**与**Hign**分数段的学生信息提取出来，查询后的结果包括：姓名、所在的系、成绩。



- **client**

```
<Button title="Task B" color="#f194ff" onPress={() =>
{setButton(11);}}/>
case 11:
  return <Table1 key={val.sname} val={val} />;
```

- **server**

```
var lower = 0;
var higher = 0;

app.post("/api/fetch/values", (req, res) => {
  lower = parseInt(req.body.min);
  higher = parseInt(req.body.max);
  // console.log(lower, higher);
});

app.get("/api/fetch/result", (req, res) => {
  // console.log(lower, higher, typeof lower, typeof higher);
  const sqlCall = "CALL __PROCEDURE_TASK__(?,?)";
  db.query(sqlCall, [lower, higher], (err, result) => {
    res.send(result);
  });
});
```

4、 实验中的问题及解决方法

在使用原生 React 框架制作网页后, 学习其衍生技术变得简单许多. 相较于原生 React , React Native 作为移动端开发有跨平台的优势, 明显的感觉是可以让 JavaScript 开发者在不会 IOS 或 Android 开发语言或不具备相应开发环境条件下, 仅仅通过对 React App 的移植就能够实现网站跨平台的移动开发. 本次试验主要尝试在 IOS 端使用系统自带模拟器进行开发和调试. 问题在实验报告3中提出过, 在开发过程中发现原生 JS 代码出现无穷 re-render 的情况, 解决方案删除了等待时间的 button 控件函数; 另外, 由于开发使用的是 IOS 模拟器, 在使用物理机进行测试时并没有像模拟器中一样顺利, 反之报了一些网络连接方面的问题. 经搜索, 将主要问题归结为开发时使用了 http 协议, 这样的协议在本机上的安全性是不能够被保障的, 但是精力有限最终没有对代码进行很大程度的改动.

5、 实验的总结与收获

1. 掌握使用 React Native 框架实现跨平台移动端窗体开发及连接数据库编程

2. 掌握了 B/S 架构应用的实现方式

3. 掌握了在 IOS 端使用模拟器调试程序的相关操作

4. 精进了 JavaScript 及 ES6 相关语法