

# 实验8 数据库设计与应用开发实验

---

学号	19335162	姓名	潘思晗
----	----------	----	-----

## 一、实验目的

掌握数据库设计的基本方法（ER图），掌握数据库开发的能力。

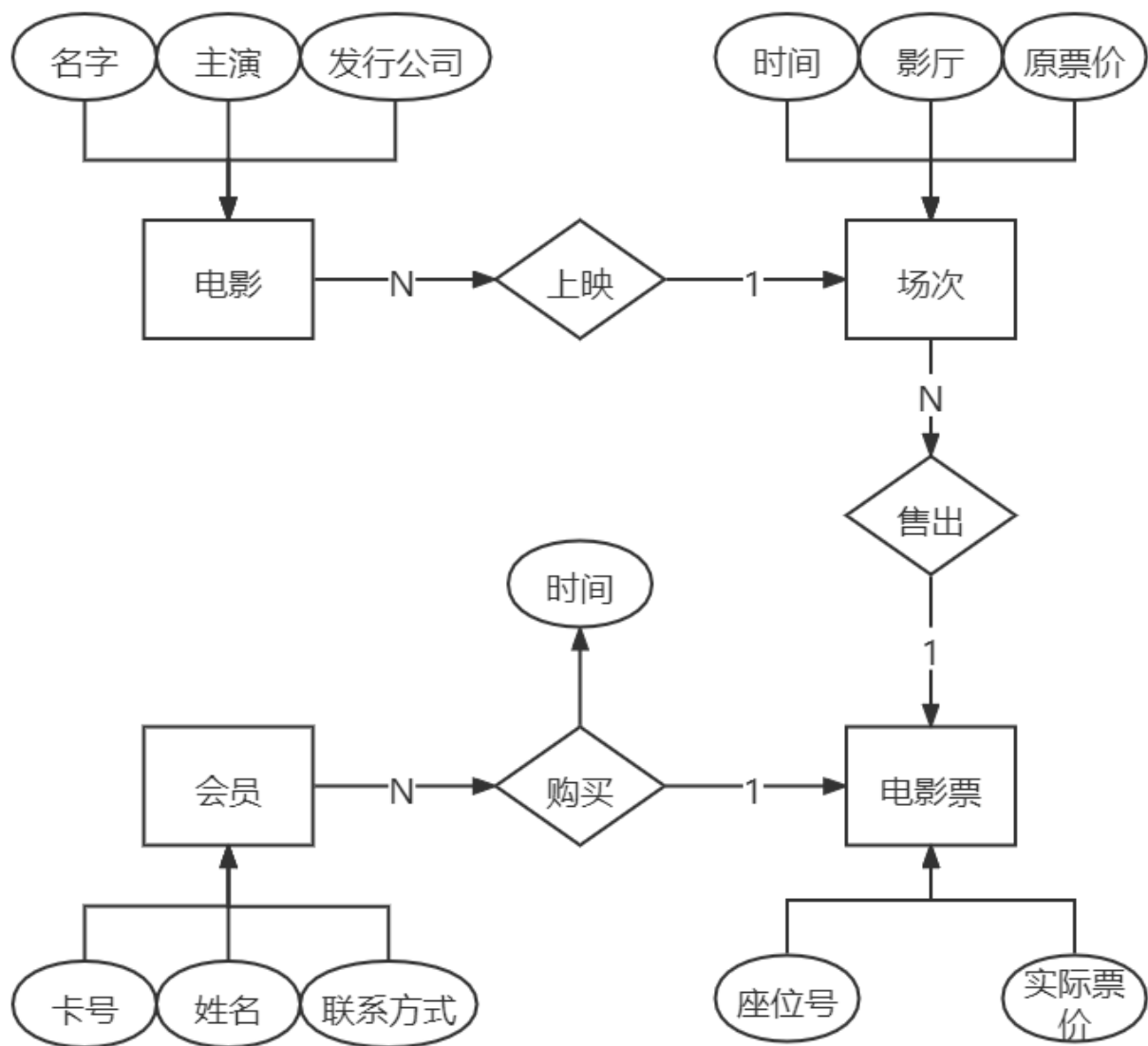
## 二、实验内容及结果

### 1.数据库设计

为某电影院设计数据库。每部电影需记录名字、主演、发行公司等信息。一部电影上映多个场次，场次要记录播放时间，影厅，和原票价。每个场次售出多张电影票，电影票要记录座位号和实际票价。影院采用会员制，一个会员可购买多张电影票，购买要记录时间，会员要记录会员卡号，姓名，联系方式。

（1）根据上述语义画出**E-R**图

E-R图绘制如下图所示：



## (2) 将该E-R模型转换为关系模型

转化后的关系模型如下：

电影（名字，主演，发行公司，场次）

场次（时间，影厅，原票价，电影票号）

电影票（座位号，实际票价）

会员（卡号，姓名，联系方式，电影票号）

## 2.数据库应用开发

构建一个 *web* 程序，实现对图书借阅数据库系统 *JY* 的图书表的增删改查。

实验环境：Windows10

### 1. JavaScript连接SQL Server数据库

建立服务器 *server.js* 来实现数据库的连接和内容的获取。

首先添加需要用到的依赖，定义 *express* 服务器并配置相关的中间件：

```
1 var express = require('express');
2 var sql = require('mssql');
3 let bodyParser=require('body-parser');
4 var app = express();
5
6 app.use(bodyParser.json());
7 app.use(bodyParser.urlencoded({ extended: true }));
8
9 app.use("/*", function(request, response, next) {
10     response.header("Access-Control-Allow-Origin", "*");
11     response.header("Access-Control-Allow-Headers", "Content-
12     Type, Authorization, X-Requested-With");
13     response.header("Access-Control-Allow-Methods",
14     "PUT,POST,GET,DELETE,OPTIONS");
15     response.header("Access-Control-Allow-Credentials", true);
16     //可以带cookies
17     response.header("X-Powered-By", 'Express');
18     if (request.method == 'OPTIONS') {
19         response.sendStatus(200);
20     } else {
21         next();
22     }
23 });
```

设置数据库配置的相关内容，此部分用于连接SQL Server数据库：

```

1  var dbConfig = {
2      user: 'sa',
3      password: 'sa1234',
4      server: 'localhost',
5      database: 'JY',
6      port: 1433,
7      options:{
8          trustServerCertificate:true
9      }
10 };
11
12 var conn = new sql.ConnectionPool(dbConfig);

```

为了实现图书表book的增删改查，设置了4个API接口，GET请求接收图书名用于查询图书信息（实现“查”），POST请求接收新的图书信息添加到数据库（实现“增”），PUT请求接收图书id和新的图书信息更新数据库（实现“改”），DELETE请求接收图书id按id在数据库中删除相关图书信息（实现“删”）。

查询所有书籍信息：

```

1  app.get('/book/list', function (req, res) {
2      var req = new sql.Request(conn);
3      conn.connect(function (err) {
4          if (err) {
5              console.log(err);
6              return;
7          }
8          // 查询 book 表
9          req.query(`SELECT * FROM book`, function (err,
10 recordset) {
11              if (err) {
12                  console.log(err);
13                  return;
14              }
15              else {
16                  console.log(recordset);
17                  res.send(recordset);
18              }
19              conn.close();
20          });
21      });
22  });

```

```
22 // res.send('Hello world');
23 })
```

依据GET请求中的图书名称查询书籍信息：

```
1 app.get('/book/search', function (request, response) {
2     var req = new sql.Request(conn);
3     let name1 = request.query.name;
4     console.log(name1);
5     conn.connect(function (err) {
6         if (err) {
7             console.log(err);
8             return;
9         }
10        // 查询 book 表
11        req.query(`SELECT * FROM book where book_name =
12        '${name1}'`, function (err, recordset) {
13            if (err) {
14                console.log(err);
15                return;
16            }
17            else {
18                console.log(recordset);
19                response.send(recordset);
20            }
21            conn.close();
22        });
23    });
24 })
```

`name1`接收书籍的名称信息`request.query.name`，按照这个书名在数据库中使用`select`语句查询信息，返回的参数`recordset`即为查询的全部结果，可以解析出来并返回给用户。

根据POST请求增加书籍信息：

```
1 app.post('/book/upload', function (request, response) {
2     console.log('收到post请求')
3     var req = new sql.Request(conn);
4     let newbook = request.body;
5     let _id = newbook.book_id;
6     let _name = newbook.book_name;
```

```

7     let _isbn = newbook.book_isbn;
8     let _author = newbook.book_author;
9     let _publisher = newbook.book_publisher;
10    let _price = newbook.book_price;
11    let _interviews = newbook.interview_times;
12
13    conn.connect(function (err) {
14        if (err) {
15            console.log(err);
16            return;
17        }
18        // 上传图书信息到 book 表
19        req.query(`insert into book
values('${_id}','${_name}','${_isbn}','${_author}','${_publisher}
','${_price}','${_interviews}')` , function (err, recordset) {
20            if (err) {
21                console.log(err);
22                return;
23            }
24            else {
25                let res = {};
26                res['state'] = true;
27                res['message'] = "Upload Succeed!";
28                console.log('添加图书成功');
29                response.send(res);
30            }
31            conn.close();
32        });
33    });
34 })

```

`newbook` 接收书籍的主体信息 `request.body`，在 `body` 中解析出各个对应的数据，并使用 `insert` 语句在数据库中增加信息，返回的参数 `res` 表示结果状态变量，如果处理成功则返回 `true` 和相应的 `message`。

根据 `PUT` 请求的图书 `id`，修改图书表中的对应信息：

```

1  app.put('/book/change', function (request, response) {
2      console.log('收到put请求')
3      var req = new sql.Request(conn);
4
5      let target_id = request.query.book_id;

```

```
6 console.log(target_id);
7
8 let newbook = request.body;
9 let _name = newbook.book_name;
10 let _isbn = newbook.book_isbn;
11 let _author = newbook.book_author;
12 let _publisher = newbook.book_publisher;
13 let _price = newbook.book_price;
14 let _interviews = newbook.interview_times;
15
16 conn.connect(function (err) {
17     if (err) {
18         console.log(err);
19         return;
20     }
21     // 更新图书信息到 book 表
22     req.query(`update book set book_name='${_name}',
23               book_isbn='${_isbn}',
24               book_author='${_author}',
25               book_publisher='${_publisher}',
26               book_price='${_price}',
27               interview_times='${_interviews}'
28               where book_id='${target_id}'`,
29     function (err, recordset) {
30         if (err) {
31             console.log(err);
32             return;
33         }
34         else {
35             let res = {};
36             res['state'] = true;
37             res['message'] = "Update Succeed!";
38             console.log('更新成功');
39             response.send(res);
40         }
41         conn.close();
42     });
43 });
44 })
```

`target_id`接收书籍的ID信息 `request.query.book_id`，按照这个ID在数据库中使用 `update` 语句更新信息，更新的具体内容需要接收 `body` 的参数并解析（8-14行），如果处理成功则 `res` 返回 `true` 和相应的 `message`（35-39行）。

依据 `DELETE` 请求的图书 `id` 在数据库中删除对应的记录：

```
1  app.delete('/book/target', function (request, response) {
2      console.log("收到delete请求")
3      var req = new sql.Request(conn);
4      let target = request.query;
5      console.log(target);
6      let _id = target.book_id;
7      console.log(_id);
8
9      conn.connect(function (err) {
10         if (err) {
11             console.log(err);
12             return;
13         }
14         // 依据 book_id 删除 book 表相关图书信息
15         req.query(`delete FROM book where book_id = '${_id}'`,
16         function (err, recordset) {
17             if (err) {
18                 console.log(err);
19                 return;
20             }
21             else {
22                 let res = {};
23                 res['state'] = true;
24                 res['message'] = "Delete Succeed!";
25                 console.log('删除图书信息成功');
26                 response.send(res);
27             }
28             conn.close();
29         });
30     });
31 }
```

`_ID`接收书籍的ID信息 `request.query.book_id`（3-7行），按照这个ID在数据库中使用 `delete` 语句删除对应的书籍信息（15-18行），如果处理成功则 `res` 返回 `true` 和相应的 `message`。



## 2. Web界面设计和测试结果

选择使用HTML表单进行简单的网页设计。

表单信息的处理过程：当点击表单的提交按钮时，输入在表单中的信息就会上传到服务器server中，然后由服务器的有关应用程序进行处理，处理后或将用户提交的信息储存在服务器端的数据库中，或者将有关的信息返回到客户端浏览器上。

基本的模式如下：

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
7          scale=1.0">
8      <title>Document</title>
9  </head>
10 <body>
11     <h1>书籍管理页面</h1>
12     ...
13 </body>
```

篇幅原因省略一部分代码，最终得到的基本网页规划设计如下所示：

# 书籍管理页面

## 书籍查询

Book Name:

Book ID:

查询

结果

## 上传图书

Book ID:

Book Name:

Book isbn:

Author:

Publisher:

Price:

Interview:

查询

## 更新图书信息

Book ID:

Book Name:

Book isbn:

Author:

Publisher:

Price:

Interview:

更新

结果

## 删除图书信息

Book ID:

确认删除

结果

不同板块可以分别实现数据库JY的图书表增删改查的功能。

下面进行测试和验证。

### （1）查询书籍信息

通过输入书名来查询书籍的相关信息。

例如输入书名《中国通史》，可以看到下方打印出与该书相关的信息。

# 书籍管理页面

## 书籍查询

Book Name:

Book ID:

查询

结果

ID: b0003  
书名: 中国通史  
作者: 于海娣  
索书号: 978-7-5388-53155  
出版社: 黑龙江科学技术出版社  
价格: 68.00  
借阅次数: 25

我们使用接口测试工具APIFOX来验证一下server服务器在后端接收query参数并返回查询信息的全过程。

开启端口后，APIFOX发送请求参数query.name='中国通史'


```
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js test.js`
应用实例，访问地址为localhost:8081
```

Query 参数

启用	参数名	参数值	说明
<input checked="" type="checkbox"/>	name	中国通史	

发送请求后，可以看到成功返回书籍的查询信息如下：

```
[nodemon] starting `node server.js test.js`
应用实例，访问地址为localhost:8081
中国通史
{
  recordsets: [ [ [Object] ] ],
  recordset: [
    {
      book_id: 'b0003 ',
      book_name: '中国通史',
      book_isbn: '978-7-5388-53155 ',
      book_author: '于海娣',
      book_publisher: '黑龙江科学技术出版社',
      interview_times: 25,
      book_price: 68
    }
  ],
  output: {},
  rowsAffected: [ 1 ]
}
```

校验 Response 

成功 (200)

返回 Body

返回 Cookie

返回 Header (7)

控制台

实际请求 •

状态码: 200 OK 耗时: 454 ms 大小: 453 B

✓ Response 数据结构校验通过!

Pretty

Raw

Preview

utf8 ▾

⬇️ 📄 🔍 ...

```
2  "recordsets": [
3    [
4      {
5        "book_id": "b0003 ",
6        "book_name": "中国通史",
7        "book_isbn": "978-7-5388-53155 ",
8        "book_author": "于海娣",
9        "book_publisher": "黑龙江科学技术出版社",
10       "interview_times": 25,
```

验证了查询功能的正确性。

## (2) 增加书籍信息

上传书籍相关的各类信息之后点击'上传'即可增加book表中的记录，演示如下：

# 上传图书

Book ID:

Book Name:

Book isbn:

Author:

Publisher:

Price:

Interview:

## 结果

上传成功!

后端测试如下，首先POST接收body的参数并解析，之后response参数返回状态变量state和message

终端返回信息如下图所示：

收到post请求  
添加图书成功

状态变量如下图所示：



查询数据库，可以发现已经成功增加新的图书数据：

结果	消息						
	book_id	book_name	book_isbn	book_author	book_publisher	interview_times	book_price
1	b0001	SQL Server 2012宝典	978-7-121-22013-5	廖梦怡	电子工业出版社	18	80.00
2	b0002	职称英语专用教材	978-7-121-14800-2	孙若红	电子工业出版社	36	45.00
3	b0003	中国通史	978-7-5388-53155	于海娣	黑龙江科学技术出版社	25	68.00
4	b0004	丰子恺儿童文学选集	978-7-5007-8972-7	丰子恺	中国少年儿童出版社	40	22.50
5	b0005	英语同义词辨析词典	978-7-5135-2294-6	赵同水	外语教学与研究出版社	6	55.00
6	b0006	数据库基础与应用	978-7-304-06430-3	徐孝凯	中央广播电视大学出版社	5	35.00
7	b0007	微积分初步	978-7-304-03742-0	赵坚	中央广播电视大学出版社	4	17.00
8	b0008	ASP.NET从入门到精通	978-7-302-28753-7	明日科技	清华大学出版社	27	89.80
9	b0009	雪国	978-7-121-22092-8	川端康成	南海出版公司	39	16.00

因此充分验证了增加图书功能的正确性。

### （3）修改书籍信息

依据目标书籍的id来修改数据库中对应书籍的信息。

## 更新图书信息

Book ID:

Book Name:

Book isbn:

Author:

Publisher:

Price:

Interview:

### 结果

更新成功！

PUT的请求参数query发送目标书籍的id以及修改内容给服务器，服务器处理后返回参数，终端得到的log讯息如下：

```
[nodemon] restarting due to changes...
[nodemon] starting 'node test.js'
应用实例，访问地址为localhost:8081
收到put请求
b0009
更新成功
```

response返回的状态变量显示也表明可以正确处理请求：

PUT

http://127.0.0.1:8081/book/change

发送

保存为用例

校验 Response

成功 (200)

返回 Body

返回 Cookie

返回 Header (7)

控制台

实际请求

状态码: 200 OK 耗时: 192 ms 大小: 34 B

Response 数据结构校验通过!

Pretty

Raw

Preview

utf8

1

2

3

4

{

"state": true,

"message": "Suceed!"

}

数据库查询验证结果如下：

更新前

结果	消息						
	book_id	book_name	book_isbn	book_author	book_publisher	interview_times	book_price
1	b0001	SQL Server 2012宝典	978-7-121-22013-5	廖梦怡	电子工业出版社	18	80.00
2	b0002	职称英语专用教材	978-7-121-14800-2	孙若红	电子工业出版社	36	45.00
3	b0003	中国通史	978-7-5388-53155	于海娣	黑龙江科学技术出版社	25	68.00
4	b0004	丰子恺儿童文学选集	978-7-5007-8972-7	丰子恺	中国少年儿童出版社	40	22.50
5	b0005	英语同义词辨析词典	978-7-5135-2294-6	赵同水	外语教学与研究出版社	6	55.00
6	b0006	数据库基础与应用	978-7-304-06430-3	徐孝凯	中央广播电视大学出版社	5	35.00
7	b0007	微积分初步	978-7-304-03742-0	赵坚	中央广播电视大学出版社	4	17.00
8	b0008	ASP.NET从入门到精通	978-7-302-28753-7	明日科技	清华大学出版社	27	89.80
9	b0009	雪国	978-7-121-22092-8	川端康成	南海出版公司	39	16.00

更新后

结果	消息						
	book_id	book_name	book_isbn	book_author	book_publisher	interview_times	book_price
1	b0001	SQL Server 2012宝典	978-7-121-22013-5	廖梦怡	电子工业出版社	18	80.00
2	b0002	职称英语专用教材	978-7-121-14800-2	孙若红	电子工业出版社	36	45.00
3	b0003	中国通史	978-7-5388-53155	于海娣	黑龙江科学技术出版社	25	68.00
4	b0004	丰子恺儿童文学选集	978-7-5007-8972-7	丰子恺	中国少年儿童出版社	40	22.50
5	b0005	英语同义词辨析词典	978-7-5135-2294-6	赵同水	外语教学与研究出版社	6	55.00
6	b0006	数据库基础与应用	978-7-304-06430-3	徐孝凯	中央广播电视大学出版社	5	35.00
7	b0007	微积分初步	978-7-304-03742-0	赵坚	中央广播电视大学出版社	4	17.00
8	b0008	ASP.NET从入门到精通	978-7-302-28753-7	明日科技	清华大学出版社	27	89.80
9	b0009	活着	978-7-302-28753-7	余华	清华大学出版社	27	89.80

充分验证了修改功能的正确性。

#### （4）删除书籍数据

通过需要删除的书籍id来删除数据库内该书的相关数据。

## 删除图书信息

Book ID:

结果

删除成功！

DELETE的请求参数query发送目标书籍的id给服务器，服务器处理后返回参数。

server服务器终端的log讯息如下：

```
收到delete请求
{ book_id: 'b0009' }
b0009
删除图书信息成功
```

response返回的状态变量如下：





查询数据库book表的结果如下：

	book_id	book_name	book_isbn	book_author	book_publisher	interview_times	book_price
1	b0001	SQL Server 2012宝典	978-7-121-22013-5	廖梦怡	电子工业出版社	18	80.00
2	b0002	职称英语专用教材	978-7-121-14800-2	孙若红	电子工业出版社	36	45.00
3	b0003	中国通史	978-7-5388-53155	于海娣	黑龙江科学技术出版社	25	68.00
4	b0004	丰子恺儿童文学选集	978-7-5007-8972-7	丰子恺	中国少年儿童出版社	40	22.50
5	b0005	英语同义词辨析词典	978-7-5135-2294-6	赵同水	外语教学与研究出版社	6	55.00
6	b0006	数据库基础与应用	978-7-304-06430-3	徐孝凯	中央广播电视大学出版社	5	35.00
7	b0007	微积分初步	978-7-304-03742-0	赵坚	中央广播电视大学出版社	4	17.00
8	b0008	ASP.NET从入门到精通	978-7-302-28753-7	明日科技	清华大学出版社	27	89.80

可以看到第9条记录被成功删除，验证了删除功能的正确性。

综上所述，完整实现了对图书借阅数据库系统JY的图书表的增删改查。

### 三、实验总结

本次实验第一次接触到web设计和数据库的开发。通过这次实验，我学习到了有关web设计和开发的很多新知识，从一开始完全没有头绪到实现完整的功能，对很有挑战性，总体而言收获巨大。