数据库实现文档

18373792 魏鑫龙

18373163 刘育含

18373803 毛健欢

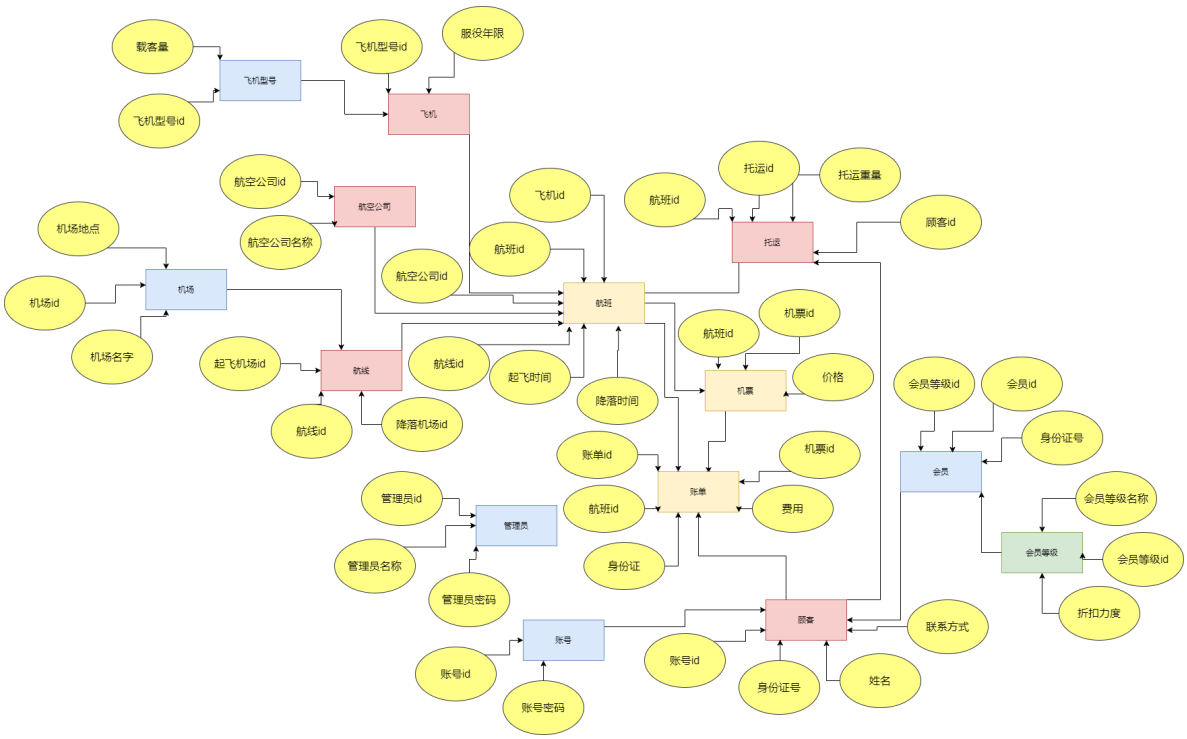
1 实现环境

前端:`vue`+`iview`组件库.

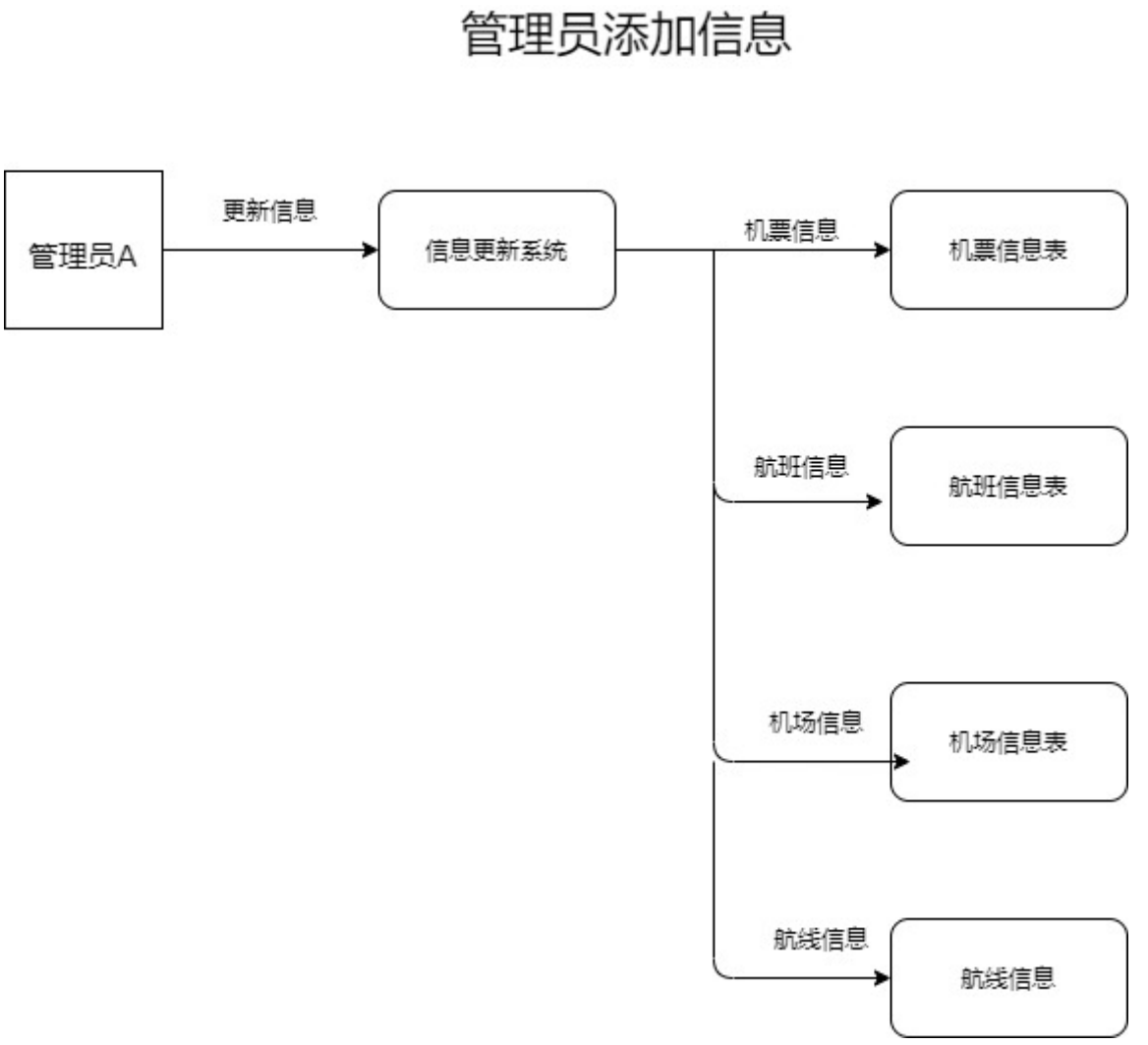
后端:`python3.7`+`django3.1.3`

2 系统功能结构图

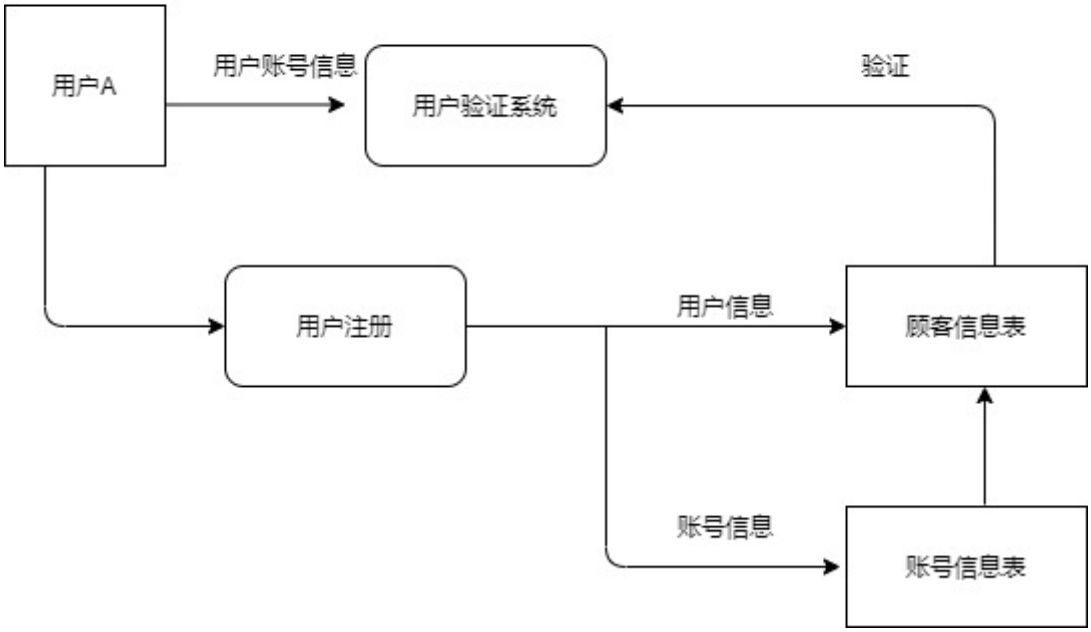
2.1 E-R图


载客量
飞机型号
飞机型号id
飞机型号id
服役年限
飞机
航空公司id
航空公司
航空公司名称
航空公司id
飞机id
航班id
航班
机场地点
机场
机场id
机场名字
起飞机场id
航线
航线id
航线id
降落机场id
起飞时间
降落时间
托运id
托运重量
航班id
托运
顾客id
机票id
航班id
机票
价格
账单id
机票id
账单
航班id
费用
身份证
管理员id
管理员
管理员名称
管理员密码
账号
账号id
账号密码
账号id
身份证号
姓名
顾客
联系方式
会员等级id
会员id
身份证号
会员
会员等级名称
会员等级
会员等级id
折扣力度

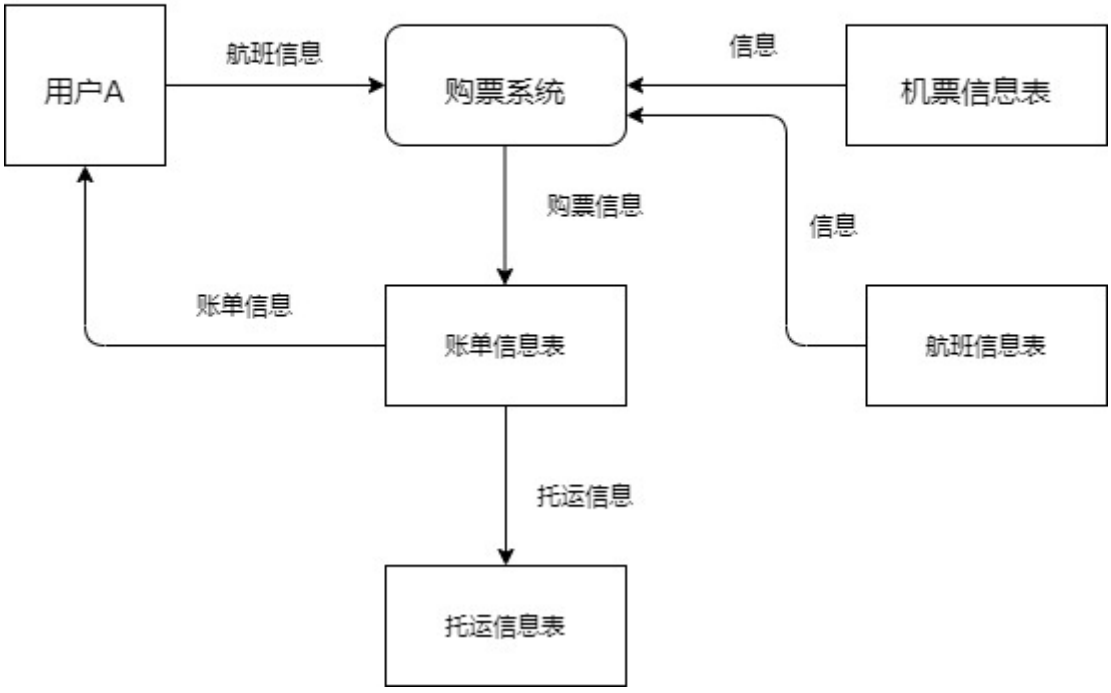
2.2 数据流图


管理员添加信息
管理员A
更新信息
信息更新系统
机票信息
机票信息表
航班信息
航班信息表
机场信息
机场信息表
航线信息
航线信息

用户登录



用户购票查询



3 表的构造

3.1 基本表及主外键约束定义

表名	属性1	属性2	属性3	属性4	属性5	属性6	属性7
顾客信息	身份证号(主码)	账号id(外码)	姓名	联系方式			
账号信息	账号id(主码)	密码					
会员信息	会员id(主码)	身份证号(外码)	会员等级(外码)				
会员等级信息	会员等级id(主码)	会员等级名称	折扣力度				
管理员信息	管理员id(主码)	管理员密码	管理员名称				
机票信息	机票id(主码)	航班id(外码)	标注价格	状态			
账单信息	账单id(主码)	身份证号(外码)	航班id(外码)	机票id(外码)	费用		
航班信息	航班id(主码)	航线id(外码)	飞机id(外码)	航空公司id(外码)	起飞时间	降落时间	剩余票数
飞机信息	飞机id(主码)	型号id(外码)	服役年限				
型号信息	型号id(主码)	载客量	最大航程				
机场信息	机场id(主码)	机场名字	机场地点				
航线信息	航线id(主码)	起飞机场id(外码)	降落机场id(外码)				
航空公司信息	航空公司id(主码)	航空公司名称					
托运信息	托运id(主码)	顾客id(外码)	航班id(外码)	重量			

3.2 索引的定义

3.2.1 唯一索引

唯一索引是不允许其中任何两行具有相同索引值的索引。当现有数据中存在重复的键值时，数据库不允许将新创建的唯一索引与表一起保存。数据库还可能防止添加将在表中创建重复键值的新数据。例如，如果在employee表中职员的姓上创建了唯一索引，则任何两个员工都不能同姓。

3.2.2 主键索引

数据库表经常有一列或多列组合，其值唯一标识表中的每一行。该列称为表的主键。在数据库关系图中为表定义主键将自动创建主键索引，主键索引是唯一索引的特定类型。该索引要求主键中的每个值都唯一。当在查询中使用主键索引时，它还允许对数据的快速访问。

4 系统安全性设计

数据库的安全性是指保护数据库以防止不合法使用所造成的数据泄露、更改或破坏 。系统安全保护措施是否有效是数据库系统主要的性能指标之一。

我们的数据库总共设计了两种角色——管理员和普通用户，并采用了静态口令鉴别的方式验证使用者的权限。管理员可以向数据库中添加新的数据，例如添加新的飞机、新的航线、新的航班等；而普通用户只能进行查询数据库和订票操作。同时在外模式方面，管理员可以看到有关机票和航线的统计信息，例如每天的航班数量、最近一段时间全国各地航班的情况概览等；而普通用户只能看到订票相关的信息。

5 触发器和函数说明

5.1 触发器

5.1.1.检查余额是否足够

在插入账单时判断用户余额是否足够,如果余额不够则拒绝该插入请求并返回"余额不足"

```
    def addBill(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            balance =
models.Customer.objects.get(id=body["customerid"]).balance
            viplevel =
models.Vip.objects.get(customer_id_id=body["customerid"]).level_id.le
vel_id
            price =
models.Flight.objects.get(id=body["flightid"]).price
            ticketNum =
models.Flight.objects.get(id=body["flightid"]).ticketNum
            if (viplevel == 1):
                price = price*0.7
            elif viplevel == 2:
                price = price*0.8
                if (balance < price):
                    return HttpResponse("余额不足", status=400)
```

5.1.2 检查密码是否符合规范

在创建账户时,检查密码是否符合要求.如果密码为空则拒绝该插入请求,并返回"密码不能为空！"

```
    def addAccount(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            if models.Account.objects.filter(id=body['accountid']):
                return HttpResponse("id已存在！",status=400)
            else:
                if body['secret'] == "":
                    return HttpResponse("密码不能为空！", status=400)
```

5.1.3.检查航班时间是否符合规范

在添加航班时,检查航班的出发时间与到达时间,如果出发时间大于到达时间,则拒绝该插入请求,并返回"时间错误!"

```
def addFlight(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            if body["time1"] > body["time2"]:
                return HttpResponse("时间不符合要求!", status=400)
```

5.2 函数说明

5.2.1 GET请求函数

1.queryAllCompay/

查询所有航空公司

输出:

```
[
    {
        "id": 1,
        "name": "中国国际航空公司"
    },
    {
        "id": 2,
        "name": "中国南方航空公司"
    },
    {
        "id": 3,
        "name": "东方航空公司"
    }
]
```

2.queryAccount/

查询账户信息及用户信息

输出:

```
[
    {
        "accountid": "aaa222", #账号
        "secret": "0000000", #密码
        "customerid": 123456, #身份证号
        "name": "xxx", #用户姓名
        "telnum": 123456, #电话号码
        "balance": 0, #余额
        "levelid": 1, #等级id
        "levelname": "铂金卡", #等级名
        "discount": 7 #打折
    }
]
```

3.queryAdmin/

查询管理员账户.

输出:

```
[
    {
        "id": "admin",
        "name": "管理员",
        "secret": "admin"
    }
]
```

4.queryAllFlight/

查询目前所有航班有关的信息

输出:

```
[
    {
        "id": 1,
        "startPortName": "双流国际机场", #起飞机场名字
        "startPorPlace": "成都",  #起飞机场位置
        "arrivePortName": "首都国际机场", #到达机场名字
        "arrivePortPlace": "北京",    #到达机场位置
        "companyName": "中国国际航空公司", #所属航空公司名字
        "price": 800,   #售价
        "startTime": "2020-10-01 14:00:00", #起飞时间
        "arriveTime": "2020-10-01 16:00:00", #到达时间
        "Typename": "波音737", #机型
        "Capacity": 150, #载客量
        "MaxVoyage": 3000, #飞机最大航程
        "leftTicket": 150 #剩余票数
    }
]
```

5.queryAllAirport/

查询所有机场信息

输出:

```
[
    {
        "id": 1,
        "name": "双流国际机场",
        "place": "成都"
    },
    {
        "id": 2,
        "name": "首都国际机场",
        "place": "北京"
    },
```

```
    {
        "id": 3,
        "name": "浦东国际机场",
        "place": "上海"
    },
    {
        "id": 4,
        "name": "白云国际机场",
        "place": "广州"
    }
]
```

6.queryAirport/

查询指定某个机场信息

输入:

```
id:1
```

输出:

```
{
    "name": "双流国际机场",
    "place": "成都"
}
```

7.queryAllPlaneType/

查询所有飞机型号

输出:

```
[
    {
        "id": 1,
        "name": "波音737",
        "capacity": 150,
        "voyage": 3000
    },
    {
        "id": 2,
        "name": "空客A320",
        "capacity": 80,
        "voyage": 6000
    }
]
```

8.queryPlaneType/

查询指定飞机型号:

输入:

```
id:1
```

输出:

```
{
    "name": "波音737",
    "capacity": 150,
    "voyage": 3000
}
```

9.queryAllRoute/

查询所有航线

输出:

```
[
    {
        "routeid":1, #该航线的id
        "id1": 1, #起飞机场id
        "name1": "双流国际机场", #起飞机场名字
        "id2": 2, #到达机场id
        "name2": "首都国际机场" #到达机场名字
    }
]
```

10.queryAllPlane/

查询所有飞机信息

输出:

```
[
    {
        "id": 1, #飞机编号
        "company_id": 1, #所属公司编号
        "company_name": "中国国际航空公司", #所属公司名字
        "type_id": 1, #所属类型编号
        "type_name": "波音737", #所属类型名字
        "work_time": 2 #已服役年限
    },
    {
        "id": 2,
        "company_id": 2,
        "company_name": "中国南方航空公司",
```

```
        "type_id": 2,
        "type_name": "空客A320",
        "work_time": 4
    }
]
```

11.queryAllBill/

查询所有账单信息

输出:

```
[
    {
        "id": 1, #账单id
        "price": 350.0, #价格
        "starttime": "2020-12-23 00:00:00", #航班起飞时间
        "arrivetime": "2020-12-24 00:00:00", #航班到达时间
        "weight": 5.0, #携带重量
        "customerid": 111111, #用户身份证号
        "customername": "xxx", #用户姓名
        "customertel": 123456, #用户电话号
        "flightid": 1, #航班号
        "planetype": "空客A320", #飞机类型
        "fromport": "双流国际机场", #起飞机场
        "fromplace": "成都", #起飞地点
        "toport": "首都国际机场", #到达机场
        "toplace": "北京" #到达地点
    }
]
```

12.estimateBill/

统计每日流水:

输出:

```
[
    {
        "time": "2020-12-23", #时间
        "price": 1050.0 #当日交易额
    }
]
```

5.2.1 POST请求函数

1.addCompany/

添加航空公司

输入:

```
{
    "name":"四川航空公司"
}
```

输出:

```
数据添加成功!
```

2.addAirport/

添加机场

输入:

```
{
    "name":"北京大兴国际机场",
    "place":"北京"
}
```

输出:

```
数据添加成功!
```

3.addPlaneType/

添加飞机型号

输入:

```
{
    "name": "波音777",
    "capacity": 300,
    "voyage": 10000
}
```

输出:

```
数据添加成功
```

4.addRoute/

添加航线信息

输入:

```
{
    "id1": 3, #起飞机场id
    "id2": 4 #到达机场id
}
```

输出:

```
数据添加成功!
```

5.addFlight/

添加航班

输入:

```
{
    "route_id": 1, #航线id
    "plane_id":2, #飞机id
    "price":1000, #价格
    "time1":"2020-12-01 14:00:00", #起飞时间
    "time2":"2020-12-01 16:00:00", #降落时间
    "num":300 #初始飞机票数
}
```

输出:

```
数据添加成功
```

6.addAccount/

添加用户账户

输入:

```
{
    "accountid":"abc123", #用户名
    "secret":"aaa111...", #密码
    "customerid":123456, #身份证号
    "name":"xxx", #姓名
    "telnum":"123456", #电话号码
    "balance":0, #余额
    "levelid":1, #vip等级id
}
```

输出:

```
数据添加成功!
```

7.addPlane/

添加飞机

输入;

```
{
    "type_id":1, #所属类型的id
    "company_id":1, #所属航空公司的id
    "work_time":5 #服役年限
}
```

输出:

```
数据添加成功!
```

8.1.addBill/

添加账单

输入:

```
{
    "customerid":123456, #顾客的身份证号
    "flightid":1, #航线的id
    "weight":5 #携带重量
}
```

5.2.3 DELETE请求函数

1.deleteCompany/

删除航空公司

输入

```
{
    "id":4
}
```

输出

```
数据删除成功
```

2.deleteAirport/

删除机场

输入:

```
{
    "id":"5"
}
```

输出:

```
数据删除成功
```

3.deletePlaneType/

删除飞机型号

输入:

```
{
    "id": 4
}
```

输出:

```
数据删除成功!
```

4.deleteRoute/

删除航线信息

输入:

```
{
    "id": 2
}
```

输出:

```
数据删除成功!
```

5.deleteFlight/

输入:

```
{
    "id": 2
}
```

输出:

```
数据删除成功！
```

6.deletePlane/

输入:

```
{
    "id":1
}
```

输出:

```
数据删除成功！
```

6 实现过程主要技术论述

TODO:

7 系统功能运行实例

TODO:

8 源程序说明

8.1 后端

8.1.2 航空公司

POST : 添加航空公司　　{name}

DELETE : 删除航空公司　{id}

GET : 查询航空公司列表　　{} -> {id, name}

method	url	effect
GET	queryAllCompay/	查询所有航空公司
POST	addCompany/	添加航空公司
DELETE	deleteCompany/	删除航空公司

```
# 航空公司
class Company:
    # 查询操作
    def queryAllCompany(request):
        if request.method == "GET":
```

```
            ret = models.Company.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
                json_dict["id"] = i.id
                json_dict["name"] = i.name

                json_list.append(json_dict)
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,content_type="application/json")
    # 添加操作
    def addCompany(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            if models.Company.objects.filter(name=body['name']):
                return HttpResponse("name已存在！",status=400)
            else:
                models.Company.objects.create(name=body['name'])
                return HttpResponse("<p>数据添加成功！</p>")

    def deleteCompany(request):
        if request.method == "DELETE":
            body = json.loads(request.body.decode('utf-8'))
            models.Company.objects.filter(id = body['id']).delete()
            return HttpResponse("<p>数据删除成功！</p>")
```

8.1.2 机场

POST : 添加机场　　{name, city}

DELETE: 删除机场　　{id}

GET: 查询所有机场　　{} -> {id, name, city}

GET: 查询指定机场　　{id} -> {name, city}

method	url	effect
GET	queryAllAirport/	查询所有机场信息
GET	queryAirport/	查询指定某个机场信息
POST	addAirport/	添加机场
DELETE	deleteAirport/	删除机场

```
# 机场
class Airport:
    # 查询操作
    def queryAllAirport(request):
```

```
        if request.method == "GET":
            ret = models.Airport.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
                json_dict["id"] = i.id
                json_dict["name"] = i.name
                json_dict["place"] = i.place
                json_list.append(json_dict)
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")

    def queryAirport(request):
        if request.method == "GET":
            json_dict = {}
            json_dict["name"] = models.Airport.objects.get(id =
request.GET.get("id")).name
            json_dict["place"] = models.Airport.objects.get(id =
request.GET.get("id")).place
            ret1 = json.dumps(json_dict)
            return HttpResponse(ret1,
content_type="application/json")

    def addAirport(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            if models.Airport.objects.filter(name=body['name']):
                return HttpResponse("name已存在！",status=400)
            else:

 models.Airport.objects.create(name=body['name'],place=body['place'])
                return HttpResponse("<p>数据添加成功！</p>")

    def deleteAirport(request):
        if request.method == "DELETE":
            body = json.loads(request.body.decode('utf-8'))
            models.Airport.objects.filter(id = body['id']).delete()
            return HttpResponse("<p>数据删除成功！</p>")
```

8.1.3 飞机型号

POST: 添加型号　{name, capacity, voyage}

DELETE: 删除型号　{id}

GET:查询所有型号　{} -> {id, name, capacity, voyage}

GET:查询指定型号的信息　{id} -> {name, capacity, voyage}

method	url	effect
GET	queryAllPlaneType/	查询所有飞机型号
GET	queryPlaneType/	查询指定飞机型号
POST	addPlaneType/	添加飞机型号
DELETE	deletePlaneType/	删除飞机型号

```
# 飞机型号
class PlaneType:
    def queryAllPlaneType(request):
        if request.method == "GET":
            ret = models.PlaneType.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
                json_dict["id"] = i.id
                json_dict["name"] = i.name
                json_dict["capacity"] = i.capacity
                json_dict["voyage"] = i.voyage
                json_list.append(json_dict)
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")

    def queryPlaneType(request):
        if request.method == "GET":
            json_dict = {}
            json_dict["name"] = models.PlaneType.objects.get(id =
request.GET.get("id")).name
            json_dict["capacity"] = models.PlaneType.objects.get(id =
request.GET.get("id")).capacity
            json_dict["voyage"] = models.PlaneType.objects.get(id =
request.GET.get("id")).voyage
            ret1 = json.dumps(json_dict)
            return HttpResponse(ret1,
content_type="application/json")

    def addPlaneType(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            if models.PlaneType.objects.filter(name=body['name']):
                return HttpResponse("name已存在！",status=400)
            else:
                models.PlaneType.objects.create(name=body['name'],
capacity=body['capacity'],voyage = body['voyage'])
                return HttpResponse("<p>数据添加成功！</p>")
```

```
    def deletePlaneType(request):
        if request.method == "DELETE":
            body = json.loads(request.body.decode('utf-8'))
            models.PlaneType.objects.filter(id=body['id']).delete()
            return HttpResponse("<p>数据删除成功！</p>")
```

8.1.4 飞机信息

POST: 添加飞机　{type, company, work_time}

DELETE: 删除飞机　{id}

GET: 查询所有飞机　　{} -> {id,type, company, work_time}

method	url	effect
GET	queryAllPlane/	查询所有飞机
POST	addPlane/	添加飞机
DELETE	deletePlane/	删除飞机

```
class Plane:
    def queryAllPlane(request):
        if request.method == "GET":
            ret = models.Plane.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
                json_dict["id"] = i.id
                json_dict["company_id"] = i.company_id.id
                json_dict["company_name"] = i.company_id.name
                json_dict["type_id"] = i.type_id.id
                json_dict["type_name"] = i.type_id.name
                json_dict["work_time"] = i.work_time
                json_list.append(json_dict)
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")

    def addPlane(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))

 models.Plane.objects.create(type_id=models.PlaneType.objects.get(id=
body['type_id']),

 company_id=models.Company.objects.get(id=body['company_id']),
                                       work_time=body['work_time'])
```

```
            return HttpResponse("<p>数据添加成功！</p>")

    def deletePlane(request):
        if request.method == "DELETE":
            body = json.loads(request.body.decode('utf-8'))
            models.Plane.objects.filter(id=body['id']).delete()
            return HttpResponse("<p>数据删除成功！</p>")
```

8.1.5 航线信息

POST:添加航线

DELETE:删除航线

GET:查询所有航线信息

method	url	effect
GET	queryAllRoute/	查询所有航线信息
POST	addRoute/	添加航线信息
DELETE	deleteRoute/	删除航线信息

```
class Route:
    def queryAllRoute(request):
        if request.method == "GET":
            ret = models.Route.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
                json_dict["routeid"] = i.id
                json_dict["id1"] = i.startPort_id.id
                json_dict["name1"] = i.startPort_id.name
                json_dict["id2"] = i.arrivePort_id.id
                json_dict["name2"] = i.arrivePort_id.name
                json_list.append(json_dict)
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")


    def addRoute(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))

 models.Route.objects.create(startPort_id=models.Airport.objects.get(
id=body["id1"]),
arrivePort_id=models.Airport.objects.get(id=body["id2"]))
            return HttpResponse("<p>数据添加成功！</p>")
```

```
    def deleteRoute(request):
        if request.method == "DELETE":
            body = json.loads(request.body.decode('utf-8'))
            models.Route.objects.filter(id=body['id']).delete()
            return HttpResponse("<p>数据删除成功！</p>")
```

8.1.6 航班信息

POST: 添加航班　{route_id, plane_id, price, time1, time2, num}

POST:卖出一票　{account_id, flight_id} //暂时不用写

DELETE: 删除航班

method	url	effect
GET	queryAllFlight/	查询目前所有航班有关的信息
POST	addFlight/	添加航班
DELETE	deleteFlight/	删除航班

```
# 航班
class Flight:
    def queryAllFlight(request):
        if request.method == "GET":
            ret = models.Flight.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
                json_dict["id"] = i.id
                json_dict["startPortName"] =
i.route_id.startPort_id.name;
                json_dict["startPorPlace"] =
i.route_id.startPort_id.place;
                json_dict["arrivePortName"] =
i.route_id.arrivePort_id.name;
                json_dict["arrivePortPlace"] =
i.route_id.arrivePort_id.place;
                json_dict["companyName"] =
i.plane_id.company_id.name;
                json_dict["price"] = i.price
                json_dict["startTime"] = i.start_time.strftime("%Y-
%m-%d %H:%M:%S");
                json_dict["arriveTime"] = i.arrive_time.strftime("%Y-
%m-%d %H:%M:%S");
                json_dict["Typename"] = i.plane_id.type_id.name;
                json_dict["Capacity"] = i.plane_id.type_id.capacity;
                json_dict["MaxVoyage"] = i.plane_id.type_id.voyage;
```

```
                json_dict["leftTicket"] = i.ticketNum;
                json_list.append(json_dict)
            json_list.sort(key=lambda x:x["startTime"])
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")

    def addFlight(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            if body["time1"] > body["time2"]:
                return HttpResponse("时间不符合要求!", status=400)

 models.Flight.objects.create(route_id=models.Route.objects.get(id=bo
dy["route_id"]),

 plane_id=models.Plane.objects.get(id=body["plane_id"]),
                                          price=body["price"],
                                          start_time=body["time1"],
                                          arrive_time=body["time2"],
                                          ticketNum=body["num"])
            return HttpResponse("<p>数据添加成功! </p>")

    def deleteFlight(request):
        if request.method == "DELETE":
            body = json.loads(request.body.decode('utf-8'))
            models.Flight.objects.filter(id=body['id']).delete()
            return HttpResponse("<p>数据删除成功! </p>")
```

8.1.7 账户信息

创建账户:{身份证号,姓名,联系方式,密码}(顾客和账户一起创建)

method	url	effect
GET	queryAccount/	查询用户账户
GET	queryAdmin/	查询管理员账户
POST	addAccount/	创建账户

```
# 账户
class Account:
    def queryAccount(request):
        if request.method == "GET":
            ret = models.Customer.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
```

```
                json_dict["accountid"] = i.acount_id.id
                json_dict["secret"] = i.acount_id.secret
                json_dict["customerid"] = i.id
                json_dict["name"] = i.name
                json_dict["telnum"] = i.tel_num
                json_dict["balance"] = i.balance
                json_dict["levelid"] =
models.Vip.objects.get(customer_id=i).level_id.level_id
                json_dict["levelname"] =
models.Vip.objects.get(customer_id=i).level_id.name
                json_dict["discount"] =
models.Vip.objects.get(customer_id=i).level_id.discount

                json_list.append(json_dict)
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")

    def queryAdmin(request):
        if request.method == "GET":
            ret = models.Admin.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
                json_dict["id"] = i.id
                json_dict["name"] = i.name
                json_dict["secret"] = i.secret
                json_list.append(json_dict)
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")

    def addAccount(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            if models.Account.objects.filter(id=body['accountid']):
                return HttpResponse("id已存在！",status=400)
            else:
                if body['secret'] == "":
                    return HttpResponse("密码不能为空！", status=400)
                models.Account.objects.create(id=body['accountid'],
secret=body['secret'])

 models.Customer.objects.create(id=body["customerid"],name=body["name
"],tel_num=body["telnum"],acount_id=models.Account.objects.get(id=bod
y['accountid']),balance=body["balance"])
```

```
 models.Vip.objects.create(customer_id=models.Customer.objects.get(id
=body["customerid"]),level_id=models.VipLevel.objects.get(level_id=bo
dy["levelid"]))

                return HttpResponse("<p>数据添加成功！</p>")
```

8.1.8 账单信息

method	url	effect
GET	queryAllBill/	查询所有账单信息
POST	addBill/	添加账单
POST	estimateBill/	统计每日流水额

```
class Bill:
    def addBill(request):
        if request.method == "POST":
            body = json.loads(request.body.decode('utf-8'))
            balance =
models.Customer.objects.get(id=body["customerid"]).balance
            viplevel =
models.Vip.objects.get(customer_id_id=body["customerid"]).level_id.le
vel_id
            price =
models.Flight.objects.get(id=body["flightid"]).price
            ticketNum =
models.Flight.objects.get(id=body["flightid"]).ticketNum
            if (viplevel == 1):
                price = price*0.7
            elif viplevel == 2:
                price = price*0.8
            if (balance < price):
                return HttpResponse("余额不足", status=400)

 models.Customer.objects.filter(id=body["customerid"]).update(balance
= balance - price)

 models.Flight.objects.filter(id=body["flightid"]).update(ticketNum=t
icketNum-1)

 models.Consign.objects.create(weight=body["weight"],customer_id=mode
ls.Customer.objects.get(id=body["customerid"]),fligth_id=models.Fligh
t.objects.get(id=body["flightid"]))
```

```
            t =
models.Consign.objects.filter(weight=body["weight"],customer_id=model
s.Customer.objects.get(id=body["customerid"]),fligth_id=models.Flight
.objects.get(id=body["flightid"]))[0]

 models.Bill.objects.create(price=price,customer_id=models.Customer.o
bjects.get(id=body["customerid"]),consign_id=t)
            return HttpResponse("<p>数据添加成功！</p>")

    def queryBill(request):
        if request.method == "GET":
            ret = models.Bill.objects.all()
            json_list = []
            for i in ret:
                json_dict = {}
                json_dict["id"] = i.id
                json_dict["price"] = i.price
                json_dict["starttime"] =
i.consign_id.fligth_id.start_time.strftime("%Y-%m-%d %H:%M:%S");
                json_dict["arrivetime"] =
i.consign_id.fligth_id.arrive_time.strftime("%Y-%m-%d %H:%M:%S");
                json_dict["weight"] = i.consign_id.weight
                json_dict["customerid"] = i.customer_id.id
                json_dict["customername"] = i.customer_id.name
                json_dict["customertel"] = i.customer_id.tel_num
                json_dict["flightid"] = i.consign_id.fligth_id.id
                json_dict["planetype"] =
i.consign_id.fligth_id.plane_id.type_id.name
                json_dict["fromport"] =
i.consign_id.fligth_id.route_id.startPort_id.name
                json_dict["fromplace"] =
i.consign_id.fligth_id.route_id.startPort_id.place
                json_dict["toport"] =
i.consign_id.fligth_id.route_id.arrivePort_id.name
                json_dict["toplace"] =
i.consign_id.fligth_id.route_id.arrivePort_id.place


                json_list.append(json_dict)
            json_list.sort(key=lambda x:x["starttime"])
            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")

    def estimateBill(request):
        if request.method == "GET":
            ret = models.Bill.objects.all()
            temp_list = []
            json_list = []
```

```
            for i in ret:
                json_dict = {}
                json_dict["price"] = i.price
                json_dict["time"] =
i.consign_id.fligth_id.start_time.strftime("%Y-%m-%d");

                temp_list.append(json_dict)
            temp_list.sort(key=lambda x:x["time"])
            temp_dict = {"time":0,"price":0}
            cnt = 0
            for i in temp_list:
                if temp_dict.get("time") and temp_dict["time"] ==
i["time"]:
                    temp_dict["price"] = temp_dict["price"]  +
i["price"]
                else:
                    if (cnt != 0):
                        json_list.append(temp_dict)
                    cnt = cnt + 1
                    temp_dict = {"time":0,"price":0}
                    temp_dict["time"] = i["time"]
                    temp_dict["price"] = i["price"]
            json_list.append(temp_dict)

            ret1 = json.dumps(json_list)
            return HttpResponse(ret1,
content_type="application/json")
```

9 收获和体会

在这次项目设计中,我们小组所选择的是设计一个机票预订系统,这对我们来说是一次尝试与创新的过程,也可以说是一个挑战的过程。虽然学了数据库这么久了,但是我们还是缺少经验。现在我们利用自己学到的知识设计并制作一个图书管理系统,这本身就是一个知识转化为生产力的过程,所以大家都很兴奋,都不同程度的投入了很高的热情与努力。
在具体的设计与实施中,我们看到并感受到了一个管理系统从无到有的过程,对具体的设计步骤、思路、方法、技巧都有了进一步的了解,并感受深刻。这次课程设计加深了我们对数据库系统设计相关知识以及 数据库相关功能的理解。比如在建立基本的表、视图、索引、存储过程、触发器等,都比以前更加熟悉了,并在解决各种问题的过程中学到了很多新的知识。在设计中我们基本能按照规范的方法和步骤进行,首先对现有的系统进行调查,并查阅有关资料,最后确定设计方案,然后设计并制作,实施过程中我们深刻的认识到认真执行管理系统软件标准的重要性,由于我们对管理系统软件相关的标准和规范不太了解,缺少行为操作准则,所以在设计中手法比较生硬,主与次也没能很好把握住,这些方面我们需要继续加强与改正。