数据库原理课程设计[实验六]

综合实验--民航订票管理系统

一、实验准备

1.业务逻辑简述

民航订票系统主要分为机场、航空公司和客户三方的服务。航空公司提供航线和飞机的资料，机场则对在本机场起飞和降落的航班和机票进行管理，而客户能得到的服务应该有查询航班路线和剩余票数，以及网上订票等功能。客户又可以分为两类：一类是普通客户，对于普通客户只有普通的查询和订票功能，没有相应的机票优惠；另一类是经常旅客，需要办理注册手续，但增加了里程积分功能和积分优惠政策。机场还要有紧急应对措施，在航班出现延误时，要发送相应的信息。

2.实验软件环境

(1)数据库系统采用 SQLServer V2014

(2)E/R图 图像绘制软件 采用 亿图图示

(3)E/R逻辑模型与关系模型建立 使用ERStdudio V8.0

二、实验内容

1.绘制E/R图

1.1 需求分析

1.1.1 航空公司

- 航空公司：用来陈列出不同的航空公司，每一家航空公司应该拥有一个唯一的编号，且应列出航空公司的名称，额外的信息还可有表示公司规模的数值，例如公司所拥有的飞机数量与航线数量。
- 飞机：每家航空公司拥有着若干飞机，而对每一架飞机都应该单独享有唯一的客机编号、客机中的座位容量(用来决定售票时的票总数)。此外，还应标识该飞机属于哪一家航空公司。
- 航线 ：每家航空公司应该拥有若干航线，而每条航线应该有一个唯一的航线编号，航线需要有三项基本信息：航线起点城市、航线的终点城市、航线的里程(航线公里数)，此外还应标识航线所属的航空公司。哪怕同起始和终止的航线若分属不同公司也是算作不同的航线(参考 国际民航航线管理办法)。

1.1.2 机场

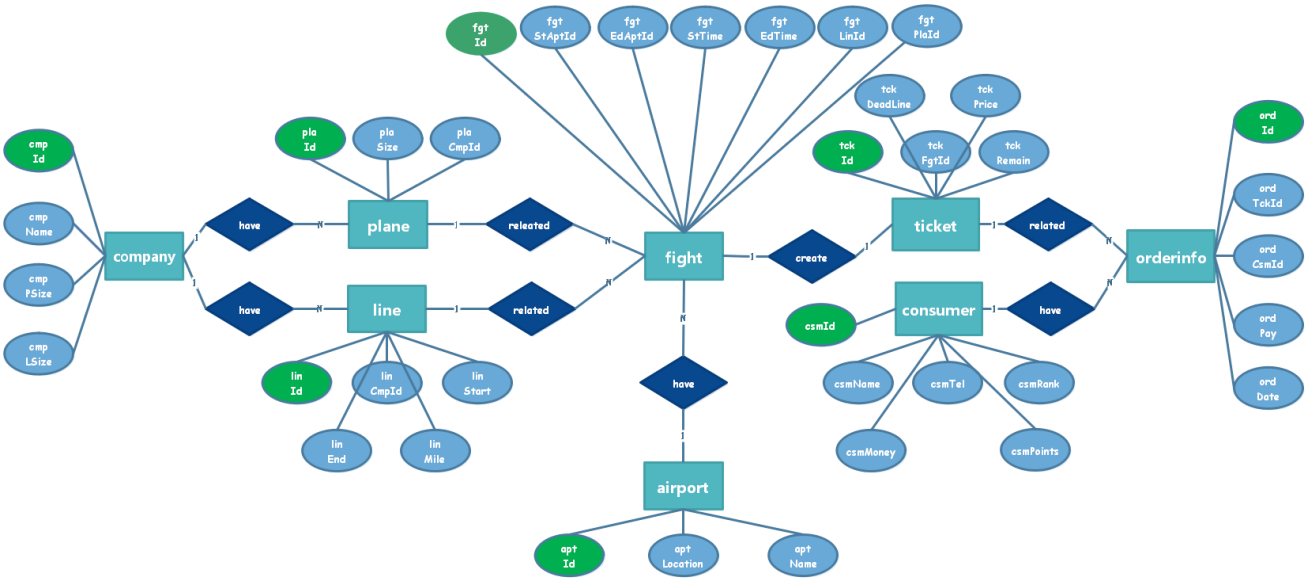
- 机场：我们需要能陈列出不同的机场，且每一家机场都应该有一个唯一标识的编号，同时应该列出机场名，以及机场容易被忽略的机场地理位置(或说机场所处的城市)。
- 航班：首先航班需要有一个唯一标识的编号，而航班即某架飞机按照某条航线飞行的一条预定计划，因此航班需要具备你对应航线和对应飞机的编号。也应该列出航班的起飞时间、到达时间(由受限SQLServer类型，此处用datetime表示)。最重要的，航班的始发机场要组织管理售票、航班的重点机场要进行检票与行李分发。但本失利中并未考虑检票相关操作，终点机场仅仅作为一个航班信息的普通字段。
- 订票管理：此处我们认为航班和售票应该分为两个独立的关系，因为航班信息更应该作为一种静态展示的资源，每轮售票依据一条航班信息可以动态生成一条售票信息来表示当前该航班售票的动态情况。该售票信息应包含主体的若干航班信息，本期的余票数量(为0时售罄 不能再购票)、本期的单张票价(每一期都能对同一航班设置不同的价格)，此外还应符合实际售票情况(发机前1个小时停止售票)，故而应包含售票的截止日期。最后，应给每一期放票信息设置一个唯一的标识号(类似于订单号一样的存在)。为保证一定的并发度，我们规定其生成规则为16位数字（14位年月日时分秒缩写+2位随机数）

1.1.3客户

- 客户：我们需要记录不同的用户，且每一位用户应该具有一个唯一标识编号，基本的用户信息应包括：用户姓名、用户的手机号码(用来在飞机延误时给用户发动通知消息)、用户等级(普通用户、经常旅客)，以及用户的里程积分。此处我们规定每飞行10公里累积1个积分，每10积分可以在购票时抵1元，而抵现之后本次将不再积累里程，客户信息还应包括其账户金额。
- 订单：用户在购票以后，理应有一条订单将用户的支付行为记录下来以备用户和机场方面查验，订单我们做一些简化后，应包含：唯一标识一条订单的订单号、订单对应的机票信息、发起订单支付的具体用户、订单的实际支付金额、订单的支付具体日期。(此处为了简化，我们没再考虑应付款)。考虑到一定的并发度避免订单号冲突，订单号的生成规则我们规定为：17位数字(14位年月日时分秒缩写+3位随机数）

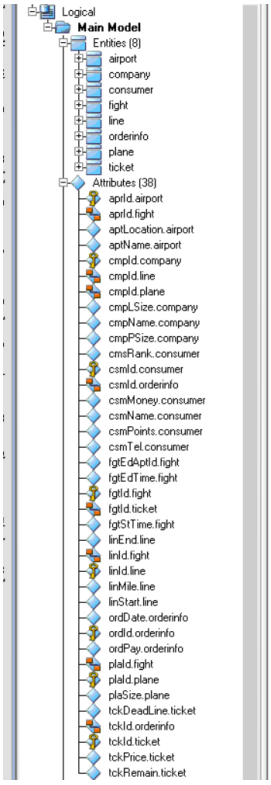
  为了简化实验中的金额操作 我们将全部金额、里程、积分全用int类型表示 里程数也全部录为10的整数倍

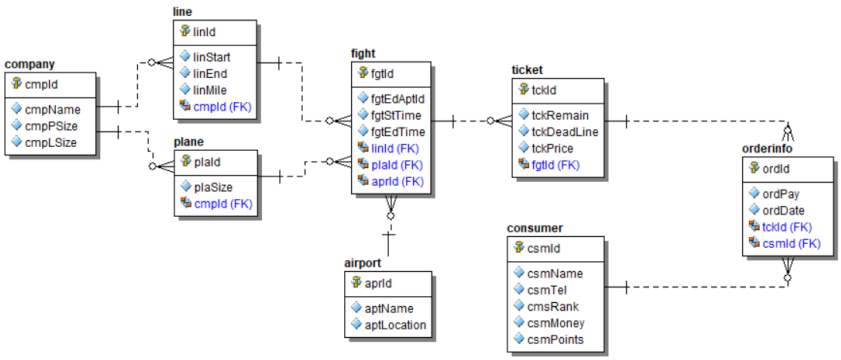
1.2 E/R图



2.模型建立

2.1 E/R逻辑模型





2.2关系模型

(为了后期管理和组织方便 我们对外键重新赋名，采用本表前缀+外键对应主表主键全称的方式)

(主键采用加粗下划线显示，外键采用加粗斜体显示)

company(**cmpId**,cmpName,cmpPSize,cmpLSize);

line(**linId**,linStart,linEnd,linMile,***linCmpId***);

plane(**plaId**,plaSize,***plaCmpId***);

fight(**fgtId**,fgtEdAptId,fgtStTime,fgtEdTime,***fgtStAptId***,***fgtLinId***,***fgtAptId***)

airport(**aptId**,aptName,aptLocation)

ticket(**tckId**,tckRemain,tckPrice,***tckFgtId***)

consumer(**csmId**,csmName,csmTel,csmRank,csmMoney,csmPoints)

orderinfo(**ordId**,ordPay,ordDate,***ordTckId***,***ordCsmId***)

3.数据库实例化

3.0 物理模型

表3-1 航空公司(company)

字段	说明	类型	可否为空
cmpId	公司编号 是公司的唯一标识,以数字1开头	char(4)	否

字段	说明	类型	可否为空
cmpName	公司名称	varchar(40)	否
cmpPSize	公司的飞机数量	smallint	
cmpLSize	公司的航线数量	smallint	

表3-2 机场(airport)

字段	说明	类型	可否为空
aptId	机场唯一标识 由2开头的四位数字组成	char(4)	否
aptName	机场名称	varchar(40)	否
aptLocation	机场所在城市	varchar(40)	

表3-3 客户(consumer)

字段	说明	类型	可否为空
csmId	用户的唯一标识 以3开头 暂定为4位数字	char(4)	否
csmName	用户的姓名	varchar(40)	否
csmTel	用户的手机号码(粗记12位) 用于紧急状况下通知	varchar(12)	否
csmRank	用户身份等级 0是普通用户 1是经常旅客	char(1)	否
csmPoints	用户的里程积分 每10公里累积1积分 每10积分可抵1元(抵现后 本次不累积里程)	int	

表3-4 航线(line)

字段	说明	类型	可否为空
linId	航线编号 以JK-xxx组成(x为数字)	char(6)	否
linCmpId	航线所属航空公司编号	char(4)	否
linStart	航线的起点城市	varchar(40)	否

字段	说明	类型	可否为空
linEnd	航线的终点城市	varchar(40)	否
linMile	航空里程 航线的公里数	int	否

表3-5 飞机(plane)

字段	说明	类型	可否为空
plaId	客机的注册编号 中国民航客机均为B-xxxx	char(6)	否
plaSize	客机的载客容量 与类型有关 最多不超400	smallint	否
plaCmpId	飞机所属公司的Id	char(4)	否

表3-6 航班(flight)

字段	说明	类型	可否为空
fgtId	航班编号 由航线所属公司的2位字母缩写同4位数字组成	char(6)	否
fgtStAptId	航班的始发站机场编号 也是航班的售票机场	char(4)	否
fgtEdAptId	航班的终点站机场编号	char(4)	
fgtStTime	航班的起飞时间	datetime	否
fgtEdIime	航班的到达时间	datetime	否
fgtLinId	航班对应的航线编号	char(6)	否
fgtPlaId	航班对应的飞机编号	char(6)	否

表3-7 放票(ticket)

字段	说明	类型	可否为空
tckId	本期放票编号 14位数字(年月日时分秒缩写)+2位随机数	char(16)	否
tckFgtId	本期放票所属的航班号	char(6)	否

字段	说明	类型	可否为空
tckRemain	本期放剩余容量(为0时售完)	smallint	否
tckDeadLine	本期放票截止日期(多为发机前一小时)	datetime	否
tckPrice	本期票价(单张机票票价在10wRMB以下)	int	否

表3-8 订单(orderinfo)

字段	说明	类型	可否为空
odrId	订单号 单笔订单的唯一标识 14位数字(年月日时分秒缩写)+3位随机数	char(17)	否
odrTckId	订单对应的放票编号	char(16)	否
ordCsmId	发起订单的用户	char(4)	否
odrPay	实际支付金额	int	否
ordDate	订单日期	datetime	否

3.1 数据库与表建立

建立数据库如下：

```
--创建民航数据库
--!!!!需要提前在D盘下创建文件夹CVA!!!!!
create database CVA on
(
name=civilAviationData,
filename='D:\CVA\civilAviationData.mdf',
size=10,
maxsize=50,
filegrowth=5%
)
log on
(
name=civilAviationLog,
filename='D:\CVA\civilAviationLog.ldf',
size=2,
maxsize=5,
filegrowth=1);
```

建立数据表如下:

```
-- 航空公司角色:
```

```
--航空公司表
create table company(
cmpId char(4) PRIMARY KEY NOT NULL,--'航空公司编号' 以1开头 4位数
cmpName varchar(40) NOT NULL ,-- '航空公司名称'
cmpPSize smallint ,-- '公司飞机数量'
cmpLSize smallint -- '公司航线数量'
);
--航线表
create table line(
linId char(6) PRIMARY KEY NOT NULL,--航线编号 以JK-xxx组成
linCmpId char(4) NOT NULL,--航线所属航空公司编号
linStart varchar(40) NOT NULL,--航线起点城市
linEnd varchar(40) NOT NULL,--航线终点城市
linMile int  NOT NULL,--航线的公里数
FOREIGN KEY(linCmpId) REFERENCES company(cmpId)
);
--客机表
create table plane(
plaId char(6) PRIMARY KEY NOT NULL,--客机注册编号 B-xxx
plaSize smallint NOT NULL,--客机的座位数量
plaCmpId char(4) NOT NULL,--客机所属公司的Id
FOREIGN KEY(plaCmpId) REFERENCES company(cmpId)
);

-- 机场角色：
--机场表
create table airport(
aptId char(4) PRIMARY KEY NOT NULL,--机场编号 以2开头 4位数
aptName varchar(40) NOT NULL,--机场名称
aptLocation varchar(40)--机场所在城市
)
--航班表
create table fight(
fgtId char(6) PRIMARY KEY NOT NULL,--航班编号 由航线所属公司的2位字母缩写同4位数字组成
fgtStAptId char(4) NOT NULL,--航班始发机场(售票机场)编号
fgtEdAptId char(4) ,--航班的终点机场编号(本例中并未使用到)
fgtStTime datetime NOT NULL,--航班的起飞时间
fgtEdTime datetime NOT NULL,--航班的到达时间
fgtLinId char(6) NOT NULL,--航班对应航线编号
fgtPlaId char(6) NOT NULL,--航班对应的飞机编号
FOREIGN KEY(fgtLinId) REFERENCES line(linId),
FOREIGN KEY(fgtPlaId) REFERENCES plane(plaId),
FOREIGN KEY(fgtStAptId) REFERENCES airport(aptId)
);
--机票表
create table ticket(
tckId char(16) PRIMARY KEY NOT NULL,--本期放票编号 14位日期缩写 2位随机数
tckFgtId char(6) NOT NULL,--本期放票所属的航班号
tckRemain smallint NOT NULL,--本期放票的剩余容量
tckDeadLine datetime NOT NULL,--本期放票截止日期
tckPrice int NOT NULL,--本期票价
FOREIGN KEY(tckFgtId) REFERENCES fight(fgtId)
);
```

```
-- 客户角色：
--客户表
create table consumer(
csmId char(4) PRIMARY KEY NOT NULL,--用户唯一标识 以3开头 4位数
csmName varchar(40) NOT NULL,--用户姓名
csmTel varchar(11),--用户手机号 用于紧急状况下通知
csmRank char(1),--用户身份等级(0是普通用户 1是经常旅客)
csmMoney int,--用户账上金额
csmPoints int --用户里程积分
);
--订单
create table orderinfo(
ordId char(17) PRIMARY KEY NOT NULL,--订单号 14位日期缩写 3位随机数
ordTckId char(16) NOT NULL,-- 订单对应的放票编号
ordCsmId char(4) NOT NULL,--发起订单的用户
ordPay int NOT NULL,--实际支付金额
ordDate datetime NOT NULL,--订单日期
FOREIGN KEY(ordTckId) REFERENCES ticket(tckId),
FOREIGN KEY(ordCsmId) REFERENCES consumer(csmId)
);
```

3.2 基础信息录入

```
--航空公司
insert into company values('1001','CA中国国际航空公司',3,3);
insert into company values('1002','MU东方航空公司(上海)',2,2);
insert into company values('1003','HU中国海南航空公司',2,2);
--航线
insert into line values('JK-101','1001','北京','上海',1080);
insert into line values('JK-102','1001','北京','南京',900);
insert into line values('JK-103','1001','北京','广州',1900);

insert into line values('JK-201','1002','北京','上海',1080);
insert into line values('JK-202','1002','上海','广州',800);

insert into line values('JK-301','1003','北京','上海',1080);
insert into line values('JK-302','1003','广州','厦门',500);
--飞机
insert into plane values('B-101',400,'1001');
insert into plane values('B-102',300,'1001');
insert into plane values('B-103',400,'1001');

insert into plane values('B-201',300,'1002');
insert into plane values('B-202',300,'1002');

insert into plane values('B-301',300,'1003');
insert into plane values('B-302',200,'1003');
--机场
insert into airport values('2001','北京首都国际机场','北京');
insert into airport values('2002','上海浦东国际机场','上海');
insert into airport values('2003','南京禄口国际机场','南京');
insert into airport values('2004','广州白云国际机场','广州');
insert into airport values('2005','厦门高崎国际机场','厦门');
--航班(为保证之后查询可以找到 航班日期取到了2021年)
```

```
insert into fight values('CA7001','2001','2002','2021-12-24 08:20:00','2021-
12-24 10:40:00','JK-101','B-101');
insert into fight values('CA7002','2001','2003','2021-12-24 06:40:00','2021-
12-24 08:40:00','JK-102','B-102');
insert into fight values('CA7003','2001','2004','2021-12-24 12:00:00','2021-
12-24 15:20:00','JK-103','B-103');
insert into fight values('MU7001','2001','2002','2021-12-24 14:30:00','2021-
12-24 16:50:00','JK-201','B-201');
insert into fight values('MU7002','2002','2004','2021-12-24 20:15:00','2021-
12-24 23:05:00','JK-202','B-202');
insert into fight values('HU7001','2001','2002','2021-12-24 18:20:00','2021-
12-24 20:50:00','JK-301','B-301');
insert into fight values('HU7002','2004','2005','2021-12-24 21:10:00','2021-
12-24 22:30:00','JK-302','B-302');

--客户
insert into consumer values('3001','刘一','18035067832','0',1000,0);
insert into consumer values('3002','李二','13233445561','1',3000,500);
insert into consumer values('3003','张三','18295899599','1',4000,1000);
```

3.3 存储过程模拟

由于时间关系和个人精力的问题 此处我们仅选取了订票相关的三项核心业务(机场放票、客户查票、客户订票)通过存储过程来模拟其较为复杂的综合性的业务逻辑功能，并给出使用示例。

3.3.1 机场放票 存储过程(select&insert类型)

```
--机场根据航班信息 来完成一条放票信息的生成
--输入参数 航班编号fgtId 放票日期tckDate datetime
create proc makeTickets(@fgtId char(6),@tckPrice int,@tckDate datetime)
as
if (@tckDate is null) set @tckDate=getdate();

--0.声明写入数据需要的变量
declare @tckId char(16),@tckRemain smallint,@tckDeadLine datetime

--1.生成放票编号
declare
--生成两位随机数
@randnum varchar(2) =cast(floor(rand()*81)+10 as varchar(2)),
--提取日期中的年月日 并补0
@ymd varchar(8) =convert(varchar(8),@tckDate,112),
--提取日期中的时分秒 并补0
@hms varchar(6)
=right('00'+datename(hh,@tckDate),2)+right('00'+datename(mi,@tckDate),2)+right
('00'+datename(ss,@tckDate),2)
--拼接上述的三个字段得到放票编号
set @tckId=@ymd+@hms+@randnum;

--2.查找航班获取放票截止日期(起飞前1小时)和放票的总票数(飞机容量)
select @tckDeadLine=dateadd(HH,-1,fgtStTime),@tckRemain=plaSize from
fight,plane
```

```
where fgtId=@fgtId and fgtPlaId=plaId;

--3.插入一条放票信息
insert into ticket values(@tckId,@fgtId,@tckRemain,@tckDeadLine,@tckPrice);

--4.查看当前插入的放票信息
select * from ticket where tckId=@tckId;
go
```

```
--机场根据航班信息 来完成一条放票信息的生成
--输入参数 航班编号fgtId 放票日期tckDate datetime
create proc makeTickets(@fgtId char(6),@tckPrice int,@tckDate datetime)
as
if (@tckDate is null) set @tckDate=getdate();

--0.声明写入数据需要的变量
declare @tckId char(16),@tckRemain smallint,@tckDeadLine datetime

--1.生成放票编号
declare
--生成两位随机数
@randnum varchar(2) =cast(floor(rand()*81)+10 as varchar(2)),
--提取日期中的年月日 并补0
@ymd varchar(8) =convert(varchar(8),@tckDate,112),
--提取日期中的时分秒 并补0
@hms varchar(6) =right('00'+datename(hh,@tckDate),2)+right('00'+datename(mi,@tckDate),2)+right('00'+datename(ss,@tckDate),2)
--拼接上述的三个字段得到放票编号
set @tckId=@ymd+@hms+@randnum;

--2.查找航班获取放票截止日期(起飞前1小时)和放票的总票数(飞机容量)
select @tckDeadLine=dateadd(HH,-1,fgtStTime),@tckRemain=plaSize from fight,plane
where fgtId=@fgtId and fgtPlaId=plaId;

--3.插入一条放票信息
insert into ticket values(@tckId,@fgtId,@tckRemain,@tckDeadLine,@tckPrice);

--4.查看当前插入的放票信息
select * from ticket where tckId=@tckId;
go
```

```
00 %
消息
命令已成功完成。
```

放票存储过程

注!用户订单号和放票编号有随机数影响 运行本pdf中代码得到编号相同结果可能性较低

执行放票，如下所示:

```
--为若干航班放票
exec makeTickets 'CA7001',300,null;
exec makeTickets 'CA7002',400,null;
exec makeTickets 'CA7003',800,'2020-12-04 07:35:20';
exec makeTickets 'MU7001',450,null;
exec makeTickets 'MU7002',600,'2020-12-13 09:41:32';
exec makeTickets 'HU7001',750,'2020-12-23 10:30:20';
exec makeTickets 'Hu7002',200,null;
```

```
--为若干航班放票
exec makeTickets 'CA7001',300,null;
exec makeTickets 'CA7002',400,null;
exec makeTickets 'CA7003',800,'2020-12-04 07:35:20';
exec makeTickets 'MU7001',450,null;
exec makeTickets 'MU7002',600,'2020-12-13 09:41:32';
exec makeTickets 'HU7001',750,'2020-12-23 10:30:20';
exec makeTickets 'Hu7002',200,null;
```

100 %

结果 消息

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020122417241875	CA7001	400	2021-12-24 07:20:00.000	300

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020122417241884	CA7002	300	2021-12-24 05:40:00.000	400

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020120407352030	CA7003	400	2021-12-24 11:00:00.000	800

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020122417241882	MU7001	300	2021-12-24 13:30:00.000	450

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020121309413263	MU7002	300	2021-12-24 19:15:00.000	600

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020122310302012	HU7001	300	2021-12-24 17:20:00.000	750

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020122417241828	Hu7002	200	2021-12-24 20:10:00.000	200

机场发起放票

3.3.2 客户查询 存储过程(复杂select类型)

```
--机票查询服务 用户需给出出发、到达的城市、出发日期(年-月-日)
--三个参数中 日期为null则显示全部日期 到达为null则显示指定出发点全部到达点(出发为null亦然) 全null则select *
--查询的信息为:航空公司-售票编号-航班编号-出发-到达-始发机场-出发时间-到达时间-机票价格-当前余票-售票截至日期-飞机座位数-航行客机
create proc showTickets(@myStart varchar(40),@myEnd varchar(40),@myDate datetime)
as
--将出发和到达的null 处理为全匹配的通配符
if (@myStart is null) set @myStart='%';
if (@myEnd is null) set @myEnd='%';
--先按照出发和到达进行限制 并筛选字段 将结果存到临时表tempTck中
select cmpName'航空公司',tckId'售票编号',fgtId'航班编号',linStart'出发',
       linEnd'到达',fgtStTime StartTime,fgtEdTime EndTime,
       aptName'始发机场',tckPrice'机票单价',tckRemain'当前余票',
       tckDeadLine'出票截至',plaSize'飞机座位数',plaId '航机编号'
into #tempTck
from company,line,plane,airport,fight,ticket
```

```
where tckFgtId=fgtId and fgtLinId=linId and fgtPlaId=plaId
and fgtStAptId=aptId and plaCmpId=cmpId and linCmpId=cmpId
and linStart like @myStart and linEnd like @myEnd;
--根据日期是否为null 将结果进一步筛选后列出
if(@myDate is null)
select * from #tempTck;
else
select * from #tempTck
where DATEDIFF(day,StartTime,@myDate)=0;
go
```

```
--机票查询服务 用户需给出出发、到达的城市、出发日期(年-月-日)
--三个参数中 日期为null则显示全部日期 到达为null则显示指定出发点全部到达点(出发为null亦然) 全null则select *
--查询的信息为:航空公司-售票编号-航班编号-出发-到达-始发机场-出发时间-到达时间-机票价格-当前余票-售票截至日期-飞机座位数-航行客机
create proc showTickets(@myStart varchar(40),@myEnd varchar(40),@myDate datetime)
as
--将出发和到达的null 处理为全匹配的通配符
if (@myStart is null) set @myStart='%';
if (@myEnd is null) set @myEnd='%';
--先按照出发和到达进行限制 并筛选字段 将结果存到临时表tempTck中
select cmpName'航空公司',tckId'售票编号',fgtId'航班编号',linStart'出发',
       linEnd'到达',fgtStTime StartTime,fgtEdTime EndTime,
       aptName'始发机场',tckPrice'机票单价',tckRemain'当前余票',
       tckDeadLine'出票截至',plaSize'飞机座位数',plaId '航机编号'
into #tempTck
from company,line,plane,airport,fight,ticket
where tckFgtId=fgtId and fgtLinId=linId and fgtPlaId=plaId
and fgtStAptId=aptId and plaCmpId=cmpId and linCmpId=cmpId
and linStart like @myStart and linEnd like @myEnd;
--根据日期是否为null 将结果进一步筛选后列出
if(@myDate is null)
select * from #tempTck;
else
select * from #tempTck
where DATEDIFF(day,StartTime,@myDate)=0;
go
```

110 %

消息

命令已成功完成。

用户搜索机票示例

```
--客户查询2021年12月24日 从北京飞往上海的航班售票信息
exec showTickets '北京','上海','2021-12-24';
--客户查询2021年12月24日 全部到达广州的航班售票信息
exec showTickets null,'广州','2021-12-24';
--客户查询2021年12月24日 全部北京出发航班售票信息
exec showTickets '北京',null,'2021-12-24';
--客户查询全部航班售票信息
exec showTickets null,null,null;
```

```
--客户查询2021年12月24日 从北京飞往上海的航班售票信息
exec showTickets '北京','上海','2021-12-24';
--客户查询2021年12月24日 全部到达广州的航班售票信息
exec showTickets null,'广州','2021-12-24';
--客户查询2021年12月24日 全部北京出发航班售票信息
exec showTickets '北京',null,'2021-12-24';
--客户查询全部航班售票信息
exec showTickets null,null,null;
```

100 %

结果 消息

	航空公司	售票编号	航班编号	出发	到达	StartTime	EndTime	始发机场	机票单价	当前余票	出票截至	飞机座位数	航机编号
1	HU中国海南航空公司	2020122310302012	HU7001	北京	上海	2021-12-24 18:20:00.000	2021-12-24 20:50:00.000	北京首都国际机场	750	300	2021-12-24 17:20:00.000	300	B-301
2	CA中国国际航空公司	2020122417241875	CA7001	北京	上海	2021-12-24 08:20:00.000	2021-12-24 10:40:00.000	北京首都国际机场	300	400	2021-12-24 07:20:00.000	400	B-101
3	MU东方航空公司(上海)	2020122417241882	MU7001	北京	上海	2021-12-24 14:30:00.000	2021-12-24 16:50:00.000	北京首都国际机场	450	300	2021-12-24 13:30:00.000	300	B-201

	航空公司	售票编号	航班编号	出发	到达	StartTime	EndTime	始发机场	机票单价	当前余票	出票截至	飞机座位数	航机编号
1	CA中国国际航空公司	2020120407352030	CA7003	北京	广州	2021-12-24 12:00:00.000	2021-12-24 15:20:00.000	北京首都国际机场	800	400	2021-12-24 11:00:00.000	400	B-103
2	MU东方航空公司(上海)	2020121309413263	MU7002	上海	广州	2021-12-24 20:15:00.000	2021-12-24 23:05:00.000	上海浦东国际机场	600	300	2021-12-24 19:15:00.000	300	B-202

	航空公司	售票编号	航班编号	出发	到达	StartTime	EndTime	始发机场	机票单价	当前余票	出票截至	飞机座位数	航机编号
1	CA中国国际航空公司	2020120407352030	CA7003	北京	广州	2021-12-24 12:00:00.000	2021-12-24 15:20:00.000	北京首都国际机场	800	400	2021-12-24 11:00:00.000	400	B-103
2	HU中国海南航空公司	2020122310302012	HU7001	北京	上海	2021-12-24 18:20:00.000	2021-12-24 20:50:00.000	北京首都国际机场	750	300	2021-12-24 17:20:00.000	300	B-301
3	CA中国国际航空公司	2020122417241875	CA7001	北京	上海	2021-12-24 08:20:00.000	2021-12-24 10:40:00.000	北京首都国际机场	300	400	2021-12-24 07:20:00.000	400	B-101
4	MU东方航空公司(上海)	2020122417241882	MU7001	北京	上海	2021-12-24 14:30:00.000	2021-12-24 16:50:00.000	北京首都国际机场	450	300	2021-12-24 13:30:00.000	300	B-201
5	CA中国国际航空公司	2020122417241884	CA7002	北京	南京	2021-12-24 06:40:00.000	2021-12-24 08:40:00.000	北京首都国际机场	400	300	2021-12-24 05:40:00.000	300	B-102

	航空公司	售票编号	航班编号	出发	到达	StartTime	EndTime	始发机场	机票单价	当前余票	出票截至	飞机座位数	航机编号
1	CA中国国际航空公司	2020120407352030	CA7003	北京	广州	2021-12-24 12:00:00.000	2021-12-24 15:20:00.000	北京首都国际机场	800	400	2021-12-24 11:00:00.000	400	B-103
2	MU东方航空公司(上海)	2020121309413263	MU7002	上海	广州	2021-12-24 20:15:00.000	2021-12-24 23:05:00.000	上海浦东国际机场	600	300	2021-12-24 19:15:00.000	300	B-202
3	HU中国海南航空公司	2020122310302012	HU7001	北京	上海	2021-12-24 18:20:00.000	2021-12-24 20:50:00.000	北京首都国际机场	750	300	2021-12-24 17:20:00.000	300	B-301
4	HU中国海南航空公司	2020122417241828	HU7002	广州	厦门	2021-12-24 21:10:00.000	2021-12-24 22:30:00.000	广州白云国际机场	200	200	2021-12-24 20:10:00.000	200	B-302
5	CA中国国际航空公司	2020122417241875	CA7001	北京	上海	2021-12-24 08:20:00.000	2021-12-24 10:40:00.000	北京首都国际机场	300	400	2021-12-24 07:20:00.000	400	B-101
6	MU东方航空公司(上海)	2020122417241882	MU7001	北京	上海	2021-12-24 14:30:00.000	2021-12-24 16:50:00.000	北京首都国际机场	450	300	2021-12-24 13:30:00.000	300	B-201
7	CA中国国际航空公司	2020122417241884	CA7002	北京	南京	2021-12-24 06:40:00.000	2021-12-24 08:40:00.000	北京首都国际机场	400	300	2021-12-24 05:40:00.000	300	B-102

3.3.3 客户订票 存储过程(select&update&insert类型)

```
--用户发起订票操作
--输入参数 放票编号@tckId 订票用户@csmId
--       订票用户欲使用积分数@points(普通用户该项写0即可) 订票日期(null取当前)
create proc orderTickets(@tckId char(16),@csmId char(4),@points int,@ordDate
datetime)
as
--引入未提交读隔离等级 以防出现"修改丢失"导致"一票多售"的情况。
set tran isolation level read uncommitted;
begin tran --开始事务
if (@ordDate is null) set @ordDate=getdate();
else set @ordDate=convert(datetime,@ordDate);
--0.声明写入或更新数据需要的变量
declare @ordId char(17),@ordPay int,@addPoints int;

--1.生成订单号
declare
--生成三位随机数(100-900)
@randnum varchar(3) =cast(floor(rand()*801)+100 as varchar(3)),
--提取日期中的年月日 并补0
@ymd varchar(8) =convert(varchar(8),@ordDate,112),
--提取日期中的时分秒 并补0
@hms varchar(6)
=right('00'+datename(hh,@ordDate),2)+right('00'+datename(mi,@ordDate),2)+right
('00'+datename(ss,@ordDate),2)
--拼接上述的三个字段得到订单号
set @ordId=@ymd+@hms+@randnum;

--2.获取放票信息（剩余票 售票截至期 票价 飞行航程）
declare @tckRemain int,@tckDeadLine datetime,@tckPrice int,@tckMile int
select
@tckRemain=tckRemain,@tckDeadLine=tckDeadLine,@tckPrice=tckPrice,@tckMile=linM
ile from ticket,fight,line
where tckId=@tckId and tckFgtId=fgtId and fgtLinId=linId;
--检验是否已经逾期(在秒级别检验)
if(DATEDIFF(ss,@ordDate,@tckDeadLine)<=0)
```

```
begin
    print'您所订机票已停止网上售卖,请至柜台处购买!';
    return;
end
--检验是否还有余票
if(@tckRemain<=0)
begin
    print'您所订航班已无余票,请等待他人退票或更换航班!';
    return;
end

--3.获取用户信息(用户等级 用户账户金额 用户里程积分)
declare @csmRank char(1),@csmMoney int,@csmPoints int
select @csmRank=csmRank,@csmMoney=csmMoney,@csmPoints=csmPoints from consumer
where csmId=@csmId;
--检验用户积分是否够
if(@csmRank='1' and @csmPoints<@points)
begin
    print'您的积分不够!';
    return;
end
--检验普通用户当前余额是否够用
if(@csmRank='0' and @csmMoney<@tckPrice)
begin
    print'您的账户金额不足购票!';
    return;
end
--检验经常旅客当前余额(附加用户期望使用的积分)是否够用
if(@csmRank='1' and @csmMoney<(@tckPrice-floor(@points/10)))
begin
    print'您的账户金额与积分不足购票!';
    return;
end

--4.更新ticket信息、用户信息并录入订单
--计算实付和积分变动
if( @csmRank='0')
begin
    set @ordPay=@tckPrice;
    set @addPoints=0;
end
else
begin
    set @points=10*floor(@points/10); --积分以10的倍数使用
    set @ordPay=@tckPrice-@points/10; --10积分抵1元
    if(@points>0) set @addPoints=-@points;  --使用积分则本次不加积分反而扣积分
    else set @addPoints=(@tckMile/10); --不使用积分 10里程加1积分
end
--更新售票和用户信息
update ticket set tckRemain=tckRemain-1 where tckId=@tckId;
update consumer set csmMoney=csmMoney-@ordPay,csmPoints=csmPoints+@addPoints
where csmId=@csmId;
--插入新的订单信息
insert into orderinfo values(@ordId,@tckId,@csmId,@ordPay,@ordDate);
commit tran --事务提交
```

```
print @csmId;
print @tckId;
print @ordId;
--5.查看当前更新和插入的信息
select * from ticket where tckId=@tckId;
select * from consumer where csmId=@csmId;
select * from orderinfo where ordId=@ordId;
go
```

```
--用户发起订票操作
--输入参数 放票编号@tckId 订票用户@csmId
--       订票用户欲使用积分数@points(普通用户该项写0即可) 订票日期(null取当前)
create proc orderTickets(@tckId char(16),@csmId char(4),@points int,@ordDate datetime)
as
if (@ordDate is null) set @ordDate=getdate();
else set @ordDate=convert(datetime,@ordDate);
--0.声明写入或更新数据需要的变量
declare @ordId char(17),@ordPay int,@addPoints int;

--1.生成订单号
declare
--生成三位随机数(100-900)
@randnum varchar(3) =cast(floor(rand()*801)+100 as varchar(3)),
--提取日期中的年月日 并补0
@ymd varchar(8) =convert(varchar(8),@ordDate,112),
--提取日期中的时分秒 并补0
@hms varchar(6) =right('00'+datename(hh,@ordDate),2)+right('00'+datename(mi,@ordDate),2)+right('00'+datename(ss,@ordDate),2)
--拼接上述的三个字段得到订单号
set @ordId=@ymd+@hms+@randnum;

--2.获取放票信息(剩余票 售票截至期 票价 飞行航程)
declare @tckRemain int,@tckDeadLine datetime,@tckPrice int,@tckMile int
select @tckRemain=tckRemain,@tckDeadLine=tckDeadLine,@tckPrice=tckPrice,@tckMile=linMile from ticket,fight,line
where tckId=@tckId and tckFgtId=fgtId and fgtLinId=linId;
--检验是否已经逾期(在秒级别检验)
if(DATEDIFF(ss,@ordDate,@tckDeadLine)<=0)
begin
    print'您所订机票已停止网上售卖,请至柜台处购买!';
    return;
```

100 %

消息

命令已成功完成。

订票模拟

```
--普通用户订票(CA中国国际航空公司 北京飞往广州的航班)
---查询之前的信息(便于对比订票)
select * from ticket where tckId='2020120407352030';
select * from consumer where csmId='3001';
exec orderTickets '2020120407352030','3001',0,null;
```

```
--普通用户订票(CA中国国际航空公司 北京飞往广州的航班)
---查询之前的信息(便于对比订票)
select * from ticket where tckId='2020120407352030';
select * from consumer where csmId='3001';
exec orderTickets '2020120407352030','3001',0,null;
```

100 %

结果 消息

订票前 放票信息与用户个人信息

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020120407352030	CA7003	400	2021-12-24 11:00:00.000	800

	csmId	csmName	csmTel	csmRank	csmMoney	csmPoints
1	3001	刘一	18035067832	0	1000	0

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020120407352030	CA7003	399	2021-12-24 11:00:00.000	800

	csmId	csmName	csmTel	csmRank	csmMoney	csmPoints
1	3001	刘一	18035067832	0	200	0

订票后信息

	ordId	ordTckId	ordCsmId	ordPay	ordDate
1	20201224172909212	2020120407352030	3001	800	2020-12-24 17:29:09 807

订单信息

```
--常驻旅客3002订票(广州飞厦门的航班)
exec showTickets '广州','厦门',null;
---查询之前的信息(便于对比订票)
select * from ticket where tckId='2020122417241828';
select * from consumer where csmId='3002';
--订票 并使用500积分抵现
exec orderTickets '2020122417241828','3002',500,null;
```

```
--常驻旅客3002订票(广州飞厦门的航班)
exec showTickets '广州','厦门',null;
---查询之前的信息(便于对比订票)
select * from ticket where tckId='2020122417241828';
select * from consumer where csmId='3002';
--订票 并使用500积分抵现
exec orderTickets '2020122417241828','3002',500,null;
```

100 %

结果 消息

用户查询机票

	航空公司	售票编号	航班编号	出发	到达	StartTime	EndTime	始发机场	机
1	HU中国海南航空公司	2020122417241828	HU7002	广州	厦门	2021-12-24 21:10:00.000	2021-12-24 22:30:00.000	广州白云国际机场	2

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020122417241828	Hu7002	199	2021-12-24 20:10:00.000	200

	csmId	csmName	csmTel	csmRank	csmMoney	csmPoints
1	3002	李二	13233445561	1	2850	500

订票前信息

	tckId	tckFgtId	tckRemain	tckDeadLine	tckPrice
1	2020122417241828	Hu7002	198	2021-12-24 20:10:00.000	200

	csmId	csmName	csmTel	csmRank	csmMoney	csmPoints
1	3002	李二	13233445561	1	2700	0

订票后信息
500积分 抵现50RMB

	ordId	ordTckId	ordCsmId	ordPay	ordDate
1	20201224173953718	2020122417241828	3002	150	2020-12-24 17:39:53.193

订单信息

3.4 事务分析

在考虑到订票系统的事务情况下，有这样一种“修改丢失”的情况：

即在高并发的情况下，当两个用户发起购票时:用户A查询得到余票为400，用户B滞后A一些查到余票为400，在用户B查询结束后用户A进行了购票并生成了订单将余票写成了399，而B用户在之后也完成了购票生成了订单，并将余票写作399。这样便造成了"一票两卖"，也就类似于两个用户同时订购了一张机票(同一个座位)。

仅仅使用事务-提交的方式虽然可以保持一系列update、insert操作的统一连贯性，但并未能处理好上述的修改丢失(或称修改覆盖)情况。针对这一情况，我们需要在存储过程的投标修改隔离级别为未读提交:

```
set tran isolation level read uncommitted;
```

三、实验总结

本次数据库综合实验的破题关键在于如何对需求进行合理的抽象和简化。过分追求让建模符合实际情况，往往会致使设计过于复杂而无法下手。就比如:体现在 对于"公司-航线"二者关系上，不同航空公司可以拥有同一条航线(这才是符合实际情况的)，这样造成了公司和航线出现多对多关系，这样不仅使得航线和公司建立关系变得复杂化，难以通过航线去追溯公司，还会使得后期的航班设计变得臃肿不堪。故而在后期，我们通过合理抽象的方式将相同”起点“-"终点"但公司不同的航线处理为”编号不同“的航线，使公司和航线变为1对多关系， 从而在不影响功能实现和模拟的情况下，简化了设计。此外，对于里程的处理上，我们统一将所有航空公司的里程数和飞行公里数进行了一比一折算，在实际中不同公司的里程数折算是不统一的。这样的简化处理也是为了降低在无关性设计中投入的精力。

在抽象中适当地扩建表段可以提供系统应用的灵活度。在最初的设计中，我们将机票售价和余票等信息一起归入了”航班“实体中，每次订票都要对航班信息表进行更新，而且航班信息往往要比售票信息确定地更容易、也更早，因为后者往往要受到市场因素的影响，而航班可以在年初就排布到全年的航班情况，但机票价格确实每若干月才会更新一批。航班作为一个相对静态的资源，而机票售卖信息作为需要高频率动态更新的资源，将二者糅合到一张表中，显然是不太合适的。此外，这样设计的弊端还在于机票售卖的形式较为单一，例如机票售卖有一项非常常用的策略：全部机票按照比例给予不同的价格以刺激购买。例如总票300张，这其中可能有200张是400元，而有100张是300元。因此，我们将航班信息和售票信息进行有机抽离。航班信息列为一张表，而针对该航班的放票情况单独列表。基于此，管理人员可以优先将很长时间的航班信息全部录入，而此后的售票则由放票相关的库表去管理调节(哪怕是同一航班的300张票分成若干条放票信息每次售不同数量不同价格)，这样可以更灵活地控制售票进程，对售票按照分段分价策略管理，而航班信息只作为静态资源被调用，与售票这一动态过程割裂开来。

起初着实为设计数据库时需要进行的E/R图绘制、逻辑模型与关系模型建立感到不忿。在此之前，我往往是直接列表、列字段、列主外键进行物理建表，从未觉得有必要去设计E/R概念图等。但经过本次实验中，我充分认识到了概念图、模型的意义所在，尤其是在较为复杂的业务逻辑中，从E/R图到各种模型再到最后的物理模型，极其类似于计算机网络体系中的”分层“概念。逐层递进不断地帮助设计者将复杂的逻辑从简单的关系概念中开始构建，逐步复杂化完整化，有助于梳理设计思路。严格遵循分析实体关系到给出逻辑模型、关系模型的思考和设计流程，也将能更加全面地考虑问题、提高设计的容错性。

在系统功能的模拟中，由于考虑到一些业务逻辑往往无法由单条语句独立完成、或是由几条简单语句来组合实现。还涉及到了一些逻辑关系、条件选择、不同查询和操作间数据的共享传递等，因而我选择了使用存储过程来对一个逻辑功能实例来进行封装，并再其中穿插了各自条件检测、数据检查等操作，得以较安全、便捷且形象地模拟系统功能的操作过程。由于涉及到组合操作，我们需要通过数据库事务来保证操作的完整连贯性，通过提交和回滚保证了在一项功能内数据操作的统一和一致。在对于高并发情况下存在的事务bug进行分析后，订票系统易出现一票多购的情况，因而要改变事务处理的隔离级别，改用未读提交的方式，虽然会在设计上会引入脏读，但在本系统中，脏读无论是从触发概率还是带来的影响上，都比修改丢失要小得多，因而折衷考虑后，我们还是认为应该采用未读提交的隔离级别来处理。

##