设计期末工程作业，选定某个信息系统应用背景

1.调研某一应用领域，给出该应用领域的详细需求描述；

火车票售票的管理系统，具有售票系统、系统管理、综合查询等功能。该系统分为三大板块，分别为售票系统、系统管理和营收结算。

售票管理：本系统主要功能包括：售当日票、预售票、退票、废票、选座位、站点班次查询、售票统计等。设计了多种售票模式，售票员可输入自定义站点编码或站点拼音代码，即可显示经过该站点的所有可售班次，班次车辆的座位状态实时、直观地显示并完成售票操作。对于售票员来说，可同时售数张相同或不同站点，相同或不同票种（全票、半票、优惠票）的车票，可以实现累加本次售票款，直至下次新售票开始。同时提供用户端，乘客也可以通过互联网购票。

系统管理：用户管理、线路设置、站点设置、退票参数、预售票参数、票价设置等。该系统主要分为四大部分，分别为用户、车票、车次和路线板块，每个板块可以进行查询与修改操作。

营收管理：售票员结算、汇总报表。对营收缴款进行管理，可以实时掌握车站营收情况，方便财务监督。能够快速、准确地生成各种营收结算报表，显著地提高了营收结算的工作质量和效率。

需求分析

用户登录注册：用户包括工作人员和乘客，有管理员、售票员、乘客 3 类用户；普通人可以通过注册功能成为新用户，用户通过登录可以使用系统提供的相应功能，如添加乘客，购买车票，查询订单等等。

系统需要提供查询列车详细信息的功能：用户根据始发站和终点站，查询可以满足自己行程要求并且正常运行的列车，并且可以进一步查看开车时间，到达时间以及列车剩余座位的数量和票价;用户可以搜索具体的某一趟车次,可以得到该车次的具体路线信息以及发车时间。

系统需要提供售票功能:售票员输入站点即可显示经过该站点的所有可售班次以及班次车辆的座位状态；一个售票员可同时售数张相同或不同站点，相同或不同票种（一等、二等、站票）的车票，可以实现累加本次售票款，直至下次新售票开始；还应实现辅助提示应找款以及退票的功能，售票员可以查询当天售票帐单，乘客也可以通过互联网购票。

系统需要提供管理功能：包括用户管理、线路设置、站点设置、票价设置等。该系统主要分为四大部分，分别为用户、车票、车次和路线板块，每个板块可以进行查询与修改操作，管理员可以添加、修改和删除用户信息、线路信息、车票信息以及车次信息。

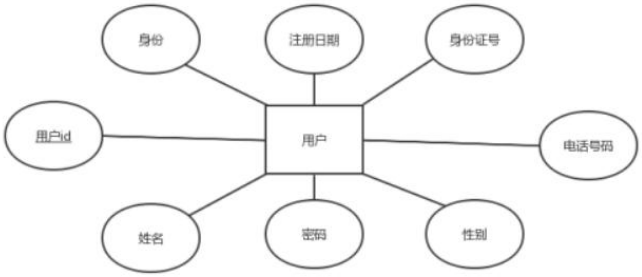
系统需要提供营收结算功能：包括售票员结算、汇总报表。对营收缴款进行管理，可以实时掌握车站营收情况，方便财务监督。能够快速、准确地生成各种营收结算报表。汇总报表主要包括：根据用户选择或者输入的各种条件（包括：时间，售票员，票号等）汇总车站运营数据，包括：售票汇总，退票汇总，预售票汇总，废票汇总，综合汇总，财务汇总，售票员结算单等等。

系统需要提供综合查询功能：包括：汇总线路、站点、班线、班次等各种基础数据；统计营运数据，生成日报表、月报表和任意时段报表；以汇总表形式，显示出售票张数、售票金额以及售票情况。

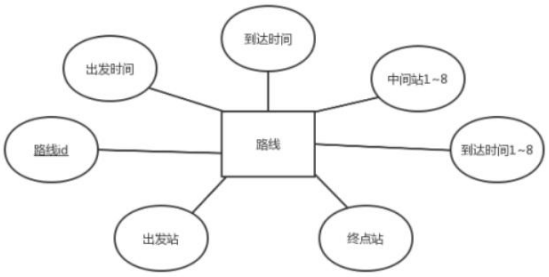
2. 采用教材中介绍的方法实现如下设计：

a) 画出该领域的概念模型 ER 图（至少有五个以上的实体，含有子类的形式，注意一定标明每个实体的主键）

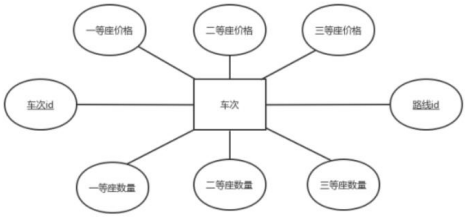
1.admin 管理员、user 用户、sellers 售票员实体：在该实体中，用户 id 为主键，用来作为每一个用户的唯一标识，同时也作为登陆系统的用户名使用；身份属性用于区分用户的类型，1 为普通用户，2 为管理员，3 为售票员；每个用户都有唯一的身份证号和电话号码。



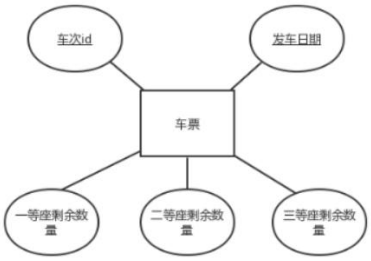
route 路线实体：在该实体中，路线 id 为主键，用来作为每一条路线的唯一标识，另外还记录了该路线途径的每一站以及到达该站的时刻。



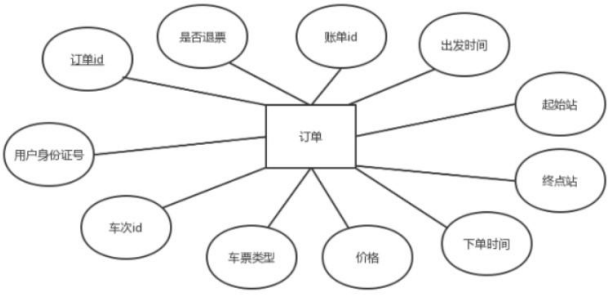
train 车次实体：在该实体中，车次 id 为主键，用来作为每一个车次的唯一标识；不同车次的列车拥有自己的一等座数量、二等座数量和三等座数量；路线 id 作为该实体的外键与路线表相联系，该车次的一等座、二等座以及三等座的价格也通过对应路线的历时和价格参数计算出来。



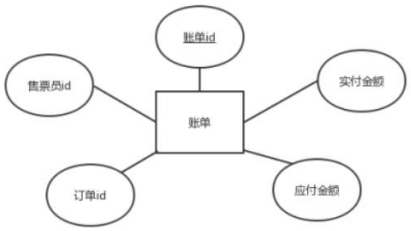
ticket 车票实体：在该实体中，车次 id 和发车日期作为联合主键，车次 id 作为外键与车次表相关联；一等座、二等座以及三等座剩余数量分别记录了某一车次某一日期当天的不同车票类型的剩余数量。

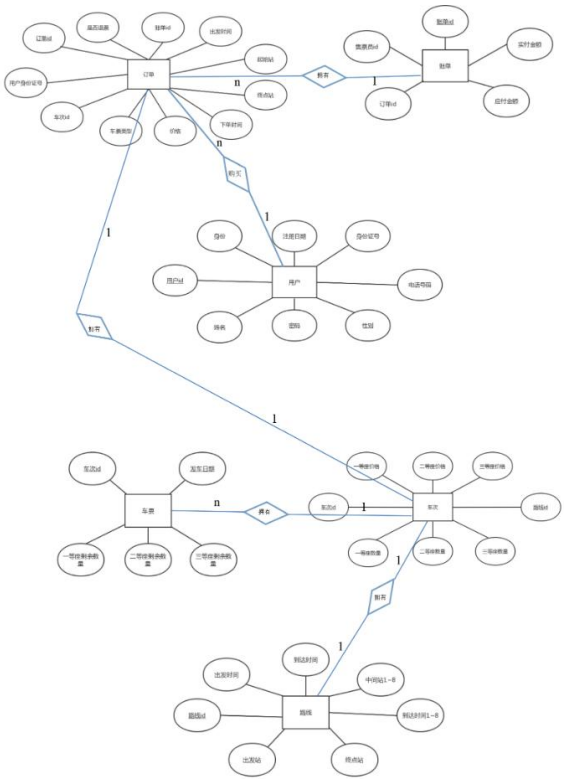


orders 用户订单实体：在该实体中，订单 id 作为主键，用来作为每一份订单的唯一标识；对于每一份独立的订单，记录着用户购买某一张车票的所有信息：包括用户身份证号、购买车次 id、车票类型（一等座、二等座和三等座）、该车票的价格、下单时间、起始站、终点站、出发时间、账单 id 以及是否发生退票行为（1 为发生了退票 2 为未发生退票）；其中车次 id 作为该实体类型的外键与车次表相关联，可以对应到相关车次的具体信息。



bill 用户账单实体：在该实体中，账单 id 作为主键，用来作为每一份账单的唯一标识；售票员 id 作为外键与用户表相关联，且该用户身份属性的值为 3，代表着该账单对应的某一个售票员；订单 id 作为外键与订单表相关联，用于记录该订单属于某一账单，一份账单可以对应到多份订单；此外该实体还包括应付金额和实付金额这两个属性，通过这两个属性可以计算出找零金额。





b）请按课堂上讲授的 ER 图转换成关系模式的方法，将上述 ER 图转换成关系模式，并标明每个关系的主键属性和外键属性；

实体转化为关系模式

用户信息（用户 id，电话号码，密码，身份证号，真实姓名，用户类型，性别，注册日期）

路线信息（路线 id，出发时间，到达时间，出发站，目的站，中间站 1，中间站 2，中间站 3，中间站 4，中间站 5，中间站 6，中间站 7，中间站 8，到达时间 1，到达时间 2，到达时间 3，到达时间 4，到达时间 5，到达时间 6，到达时间 7，到达时间 8）

车次信息（车次 id，路线 id，一等座数量，二等座数量，三等座数量，一等票价格，二等票价格，三等票价格）

车票信息（车次 id，发车日期，一等座剩余数量，二等座剩余数量，三等座剩余数量）

订单信息（订单 id，用户身份证号，账单 id，车次 id，车票类型，价格，下单时间，发车时间，出发地，目的地，是否退票）

账单信息（账单 id，订单 id，售票员 id，实付金额，应付金额）

联系转化为关系模式

用户拥有订单（用户身份证号，订单编号）

订单拥有车次（订单编号，车次编号）

列车拥有路线信息（车次编号，路线编号）

账单拥有订单（账单编号，订单编号）

账单对应售票员（账单编号，售票员编号）

车票拥有车次（车次编号，发车日期）

c）用 SQL 语句创建上述关系模式

```
create schema “Ticket” authorization Jiang
create table User (
u_id int primary key,
u_name varchar(45) not null,
u_password varchar(45) not null,
u_phone varchar(45) not null,
u_idcard varchar(45) not null,
date varchar(45) not null,
u_authority varchar(45) not null,
u_sex varchar(45)not null
);
```

```
create table Route(
route_id char(6) primary key,
start_station varchar(20) not null,
destination varchar(20) not null,
way_station_1 varchar(20),
way_station_2 varchar(20),
way_station_3 varchar(20),
way_station_4 varchar(20),
way_station_5 varchar(20),
way_station_6 varchar(20),
way_station_7 varchar(20),
way_station_8 varchar(20),
start_time time,
end_time time,
arrive_time_1 time,
arrive_time_2 time,
arrive_time_3 time,
arrive_time_4 time,
arrive_time_5 time,
arrive_time_6 time,
arrive_time_7 time,
arrive_time_8 time
);
```

```
create table Train(
train_id char(6) primary key,
route_id char(6),
train_first_num int not null,
train_second_num int not null,
train_third_num int not null,
first_price int not null,
second_price int not null,
third_price int not null,
foreign key route_id reference Route(route_id),
);
```

```
create table Orders(
order_id bigint primary key,
u_idcard char(18),
train_id char(6),
ticket_type varchar(10),
price int not null,
order_time timestamp not null,
bill_id bigint,
if_refund int,
start_place varchar(20),
end_place varchar(20),
start_time timestamp not null,
foreign key (u_idcard) references User(u_idcard),
foreign key(train_id) references Train(train_id)
);
```

```
create table Bill(
bill_id bigint primary key,
order_id bigint,
u_id int,
amount_real int,
amount_payable int,
foreign key(u_id) references User(u_id),
foreign key(order_id)references Order(order_id)
);
```

```
create table Ticket(
train_id char(6) primary key,
train_first_num int,
train_second_num int,
train_third_num int,
t_date date primary key,
foreign key train_id reference Train(train_id)
);
```

d) 给出该数据库模式上 5 个查询语句样例，分别为：

单表查询

查询性别为男的所有用户信息

```
select * from User where u_sex = ‘男’;
```

多表连接查询

车次 id 为 1 的出发地站、终点站、出发时间、到达时间

```
select
Routine.start_station,Routine.destination,Routine.start_time,Routine.end_time
from Routine
inner join Train on Train.route_id = Routine.route_id
where Train.train_id = 1;
```

多表嵌套查询

查找用户 id = 1 的所有车票的订单 id

```
select Order.order_id
from Order
inner join Ticket on Ticket.train_id = Order.train.id
where Ticket.train_id  in (select train_id from Ticket where u_id = 1);
```

EXISTS 查询

查询账单中是否有 id = 1 的订单

```
select exists(select * from bill where order_id = 1);
```

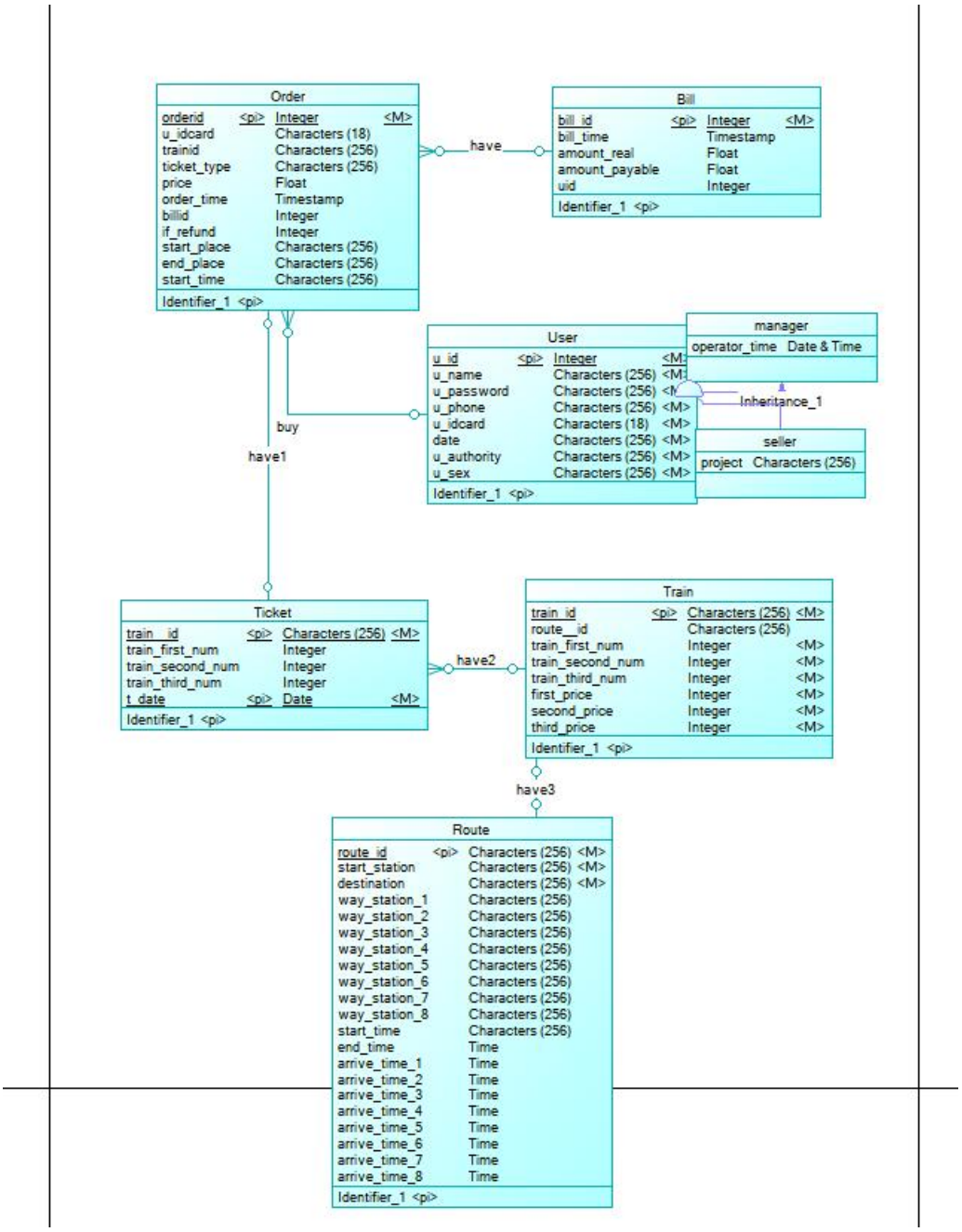
聚合操作查询

查寻 id = 1 的账单数

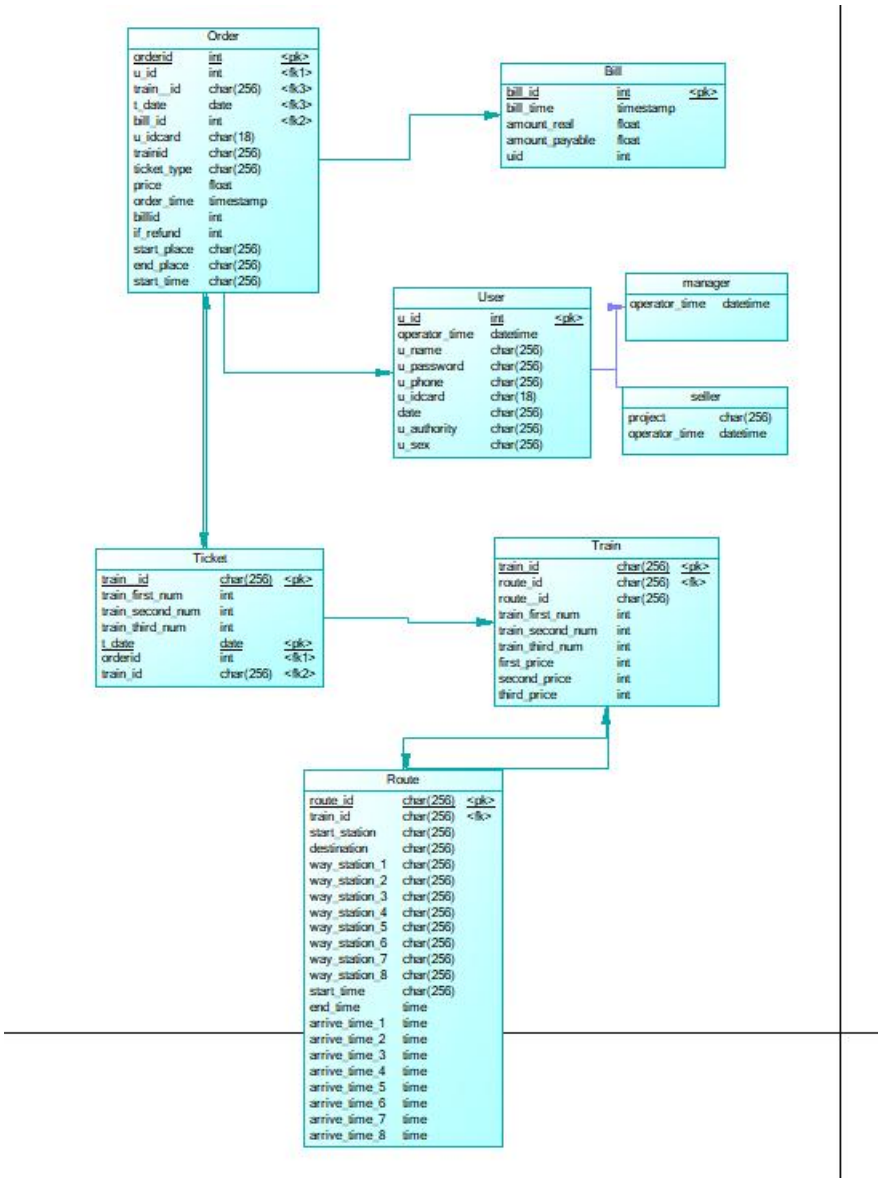
```
select count(*) from Bill where u_id = 1;
```

3. 使用 PowerDesigner 工具实现如下设计：

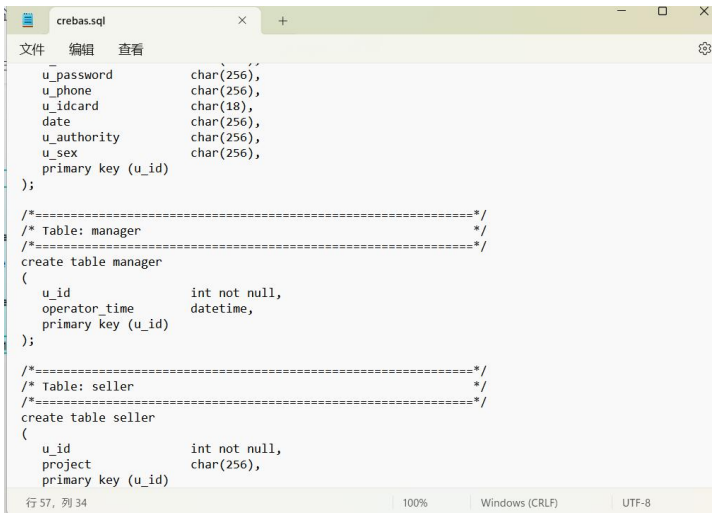
a) 画出该领域的概念模型 ER 图，给出 ER 图截图；



b) 使用 PowerDesigner 工具，将上述 ER 图转为关系模型图，给出关系模型图截图；



c）使用 PowerDesigner 工具，生成创建数据库的 SQL 语句。



4. 分析比较采用上述两种方法

a）概念模型、关系模型两种关系模式的设计是否存在差异？如有差异，这种差异是否对后期的实现带来不同的影响？

概念模型是一种高层次的数据模型，用于描述业务实体、属性和它们之间的关系。概念模型通常使用实体-属性-关系图（ER 图）进行表示。

而关系模型是一种底层的数据模型，用于描述数据表、列和它们之间的关系。关系模型通常使用关系型数据库进行实现。

在概念模型的设计中，需要考虑业务实体和它们之间的关系，而在关系模型的设计中，需要考虑数据表和列的设计和规范化。不同的模型转换可能会带来不同的影响，如数据冗余、性能问题等。

b) PowerDesigner 工具生成的 SQL 语句有什么样的特点？ 为什么会出现一些附加语句？它的作用
是什么？

生成的 SQL 语句包含完整的表结构，包括表名、列名、数据类型、约束、索引等。

使用标准的 SQL 语法，可以在不同的关系型数据库中使用。

具有可读性和可维护性，易于进行修改和优化。

附加语句是指在 PowerDesigner 生成的 SQL 语句中，除了表结构定义以外的其他语句，如创建序列、触发器等。这些附加语句通常用于实现一些高级特性，如自增主键、联级操作等。它们的作用是为数据库提供更多的功能和灵活性，但同时也可能增加了数据库的复杂度和维护成本。