一、设计题

任务：一个车辆租赁公司希望建立一个数据库来管理对用户的车辆租赁，请为这个车辆租赁公司完成逻辑数据库设计。

数据需求描述如下：

（a）该公司有不同的门店，门店有主要属性：编号、名称、地址、联系电话。

（b）每个门店有一定的员工，包括经理、高级技师、普通技工和租赁代表，员工有主要属性：编号、姓名、职位名、联系电话。一个门店只有一名经理，经理负责管理该门店所有员工；每位高级技师负责指导几名普通技工，每位普通技工由一名高级技师指导。

（c）每个门店有若干可供租赁的车辆，车辆有主要属性：编号、车牌号、型号、购置日期、生产厂家、购买价格、累计行驶路程。

（d）顾客有主要属性：编号、姓名、出生日期、驾照号码、联系电话、会员等级。顾客每次租赁一部车时会签订一份租赁合同，租赁合同有唯一的租赁号，主要记录租赁开始时间、租赁结束时间、押金金额、实际付款金额，还要记录是哪位租赁代表和顾客签订的合同。

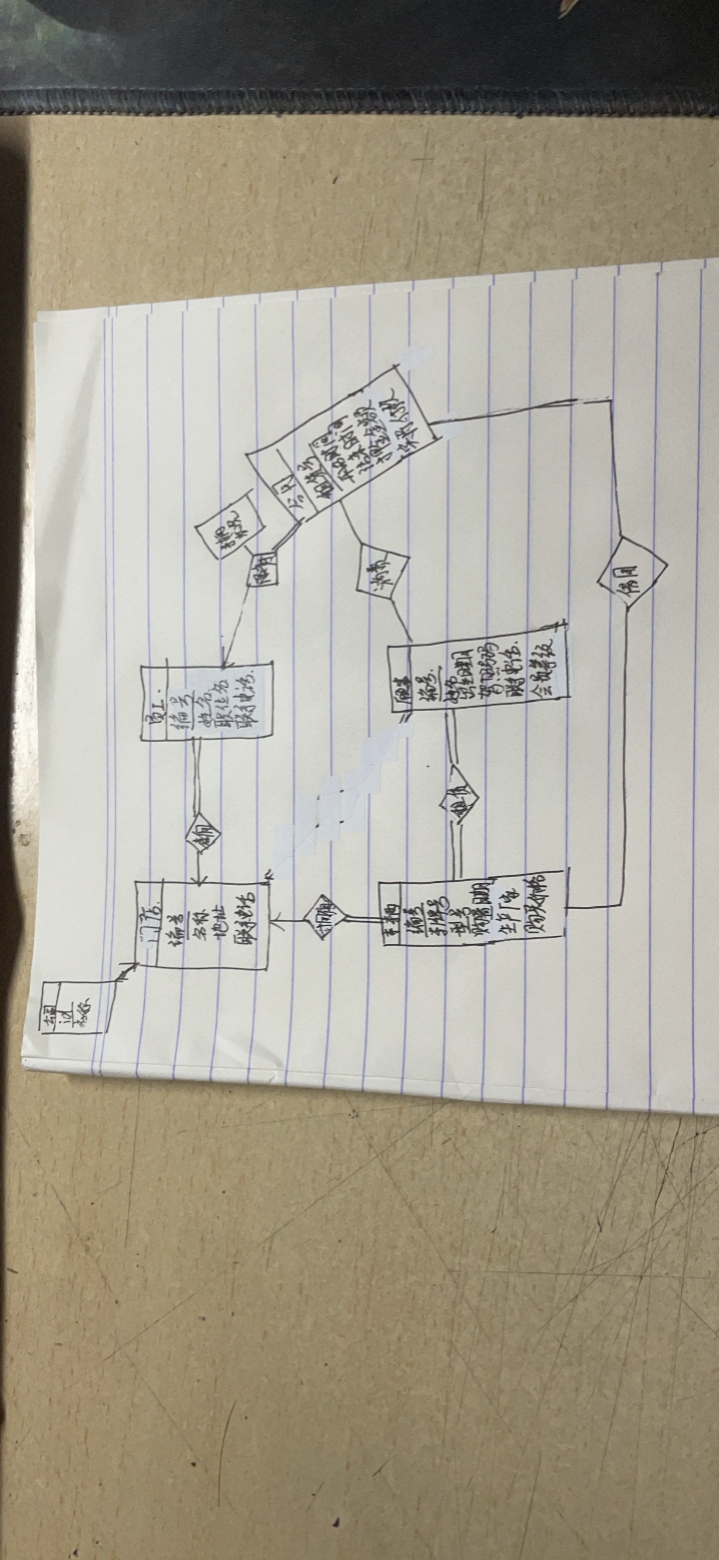
（e）每次租赁完成后，需要对车辆进行检查，需要记录下车辆的整体状况，还需要记录是哪位高级技师或普通技工在何时进行的检查。

请完成如下设计题：

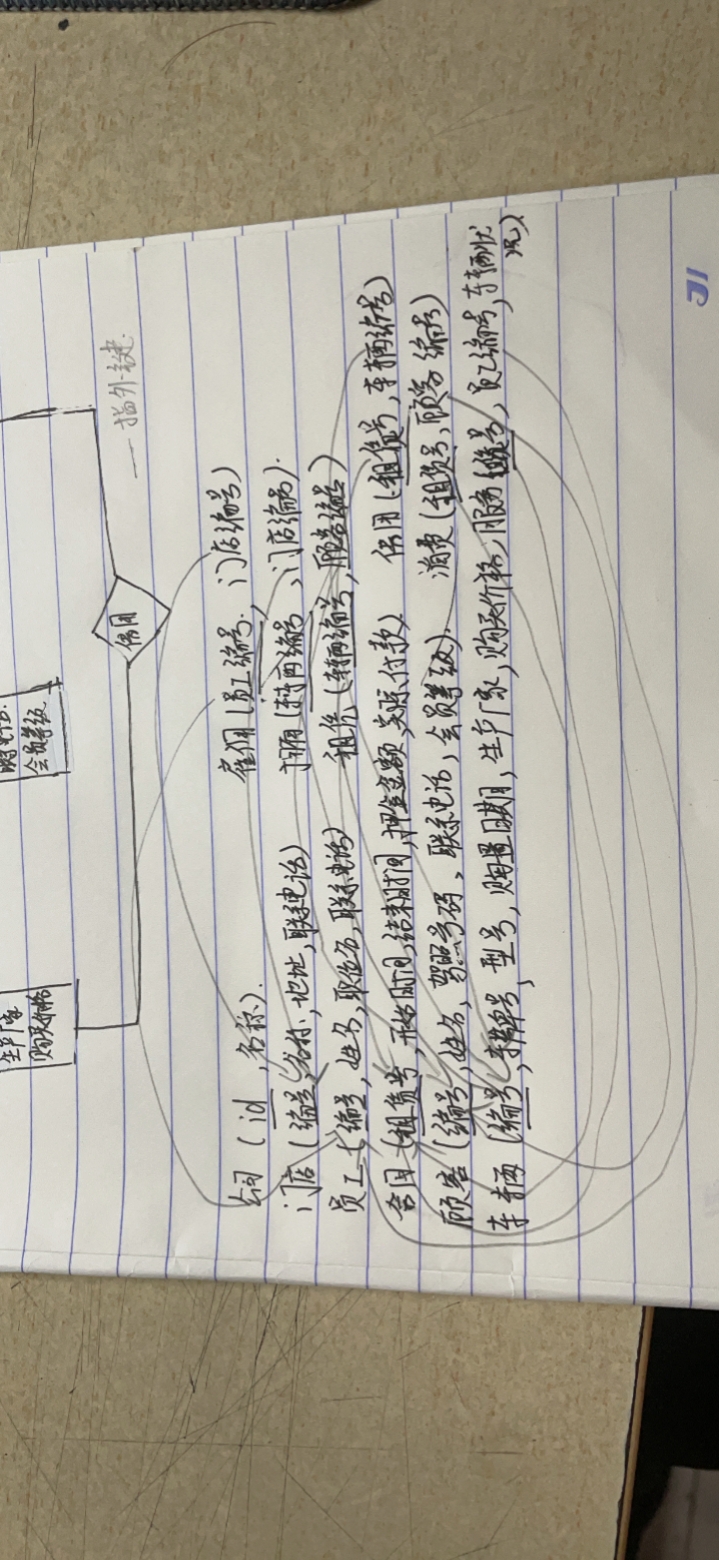
1）画出ER图，在图上注明实体名、关系名、关系多样性（映射基数）。可使用上学期的数据库ER画法和这学期的Chen氏画法，图中实体必须标注出主键，无须标注出所有属性，联系集属性（如果有）则必须标注。

2）将E-R图转换为关系模式，注明关系的属性、主键和外键。关系定义的写法举例如下：Table1(t1column1,t1column2,t1column3,…)，主键t1column1，外键t1column2引用Table2表的t2column1。

1）



2）



二、上机题

建立数据库filmdb，执行下发的FilmDB\_original.sql文件，建立数据库表和导入基本数据，各个表字段定义及含义为：

电影信息表FilmInfo

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 约束/默认/主键/外键 |
| filmID | char(16) | 电影编号 | **主键** |
| filmName | varchar(50) | 电影名称 | **非空** |
| typeID | char(2) | 电影类别编号 | **非空，外键引用FilmType的typeID字段** |
| actor | varchar(255) | 演员 |  |
| director | varchar(50) | 导演 |  |
| filmGrade | decimal(3,1) | 电影评分 |  |

电影类别表FilmType

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 约束/默认/主键/外键 |
| typeID | char(2) | 电影类别编号 | **主键** |
| typeName | varchar(20) | 电影类别名称 | **非空** |

电影排片放映信息表PlayInfo

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 约束/默认/主键/外键 |
| playID | int | 放映编号 | **主键，自增1** |
| filmID | char(16) | 电影编号 | **非空，外键引用FilmInfo的filmID字段** |
| cinema\_hallID | int | 门店放映厅编号 | **非空，外键引用cinema\_hallInfo的cinema\_hallID字段** |
| startTime | datetime | 电影开始时间 | **非空** |
| endTime | datetime | 电影结束时间 | **非空** |
| price | decimal(6,2) | 原始票价 | **非空，默认为100** |

出票信息表TicketInfo

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 约束/默认/主键/外键 |
| seatID | varchar(8) | 座位编号 | **与playID联合成为主键** |
| playID | int | 放映编号 | **与seatID联合成为主键，外键引用PlayInfo的playID字段** |
| memberID | int | 会员编号 | **非空，外键引用MemberInfo的memberID字段** |
| finalPrice | decimal(6,2) | 会员票价 |  |
| grade | decimal(3,1) | 会员评分 | **非空，默认为8分** |

影院门店放映厅表cinema\_hallInfo

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 约束/默认/主键/外键 |
| cinema\_hallID | int | 门店放映厅编号 | **主键，自增1** |
| address | varchar(20) | 门店地址 | **非空** |
| hallName | varchar(20) | 放映厅名称 | **非空** |
| capacity | int | 放映厅容量 | **非空，默认为100** |

会员信息表MemberInfo

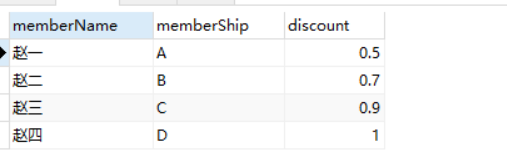
|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 含义 | 约束/默认/主键/外键 |
| memberID | int | 会员编号 | **主键，自增1** |
| memberName | varchar(20) | 会员姓名 |  |
| memberShip | char(2) | 会员等级 | **非空，默认为’D’** |
| discount | decimal(2,1) | 会员票价折扣 | **非空** |

1. 写出下面的SQL查询语句。
2. 查询会员信息中姓“赵”的会员信息，输出会员姓名、会员等级、会员折扣

SELECT memberName,memberShip,discount

FROM memberinfo

WHERE memberName LIKE "赵%"



1. 查询“这个杀手不太冷”这部电影的会员票价最高的出票详细信息，输出座位编号、放映编号、会员票价、会员评分

SELECT seatID,playinfo.playID,finalPrice,grade

FROM ticketinfo,playinfo,filminfo

WHERE filminfo.filmName = "这个杀手不太冷" AND filminfo.filmID = playinfo.filmID AND playinfo.playID = ticketinfo.playID AND finalPrice =

(

SELECT MAX(t1.finalPrice)

FROM ticketinfo AS t1,playinfo AS p1,filminfo AS f1

WHERE f1.filmID = p1.filmID AND f1.filmName = "这个杀手不太冷" AND p1.playID = t1.playID

)



1. 查询放映信息中票价在100到200之间的放映信息，输出放映编号、电影编号、电影名称、电影类别名称、原始票价

SELECT playID,filminfo.filmID,filminfo.filmName,filmtype.typeName,price

FROM playinfo,filmtype,filminfo

WHERE price >=100 AND price<=200 AND filminfo.filmID=playinfo.filmID AND filminfo.typeID=filmtype.typeID



1. 查询各门店放映厅排片信息，输出放映编号、电影编号、门店地址、电影名称、电影类别编号、演员、导演和原始票价（如果该门店放映厅没有排片信息，也要求输出相关信息）

SELECT B.playID, B.filmID, A.address, C.filmName, C.typeID, C.actor, C.director, B.price

FROM cinema\_hallinfo A left outer join playinfo B ON A.cinema\_hallID = B.cinema\_hallID left outer join filminfo C ON B.filmID = C.filmID;





1. 查询购买了所有排片电影的会员信息，输出会员姓名、会员等级、会员折扣

SELECT memberName,memberShip,discount

FROM memberinfo c

WHERE c.memberID=(

SELECT A.memberID

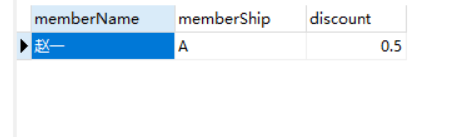
FROM ticketinfo A left outer join playinfo B on A.playID=B.playID

GROUP BY A.memberID

HAVING COUNT(DISTINCT(filmID))

= (SELECT COUNT(DISTINCT(filmID))

FROM playinfo))



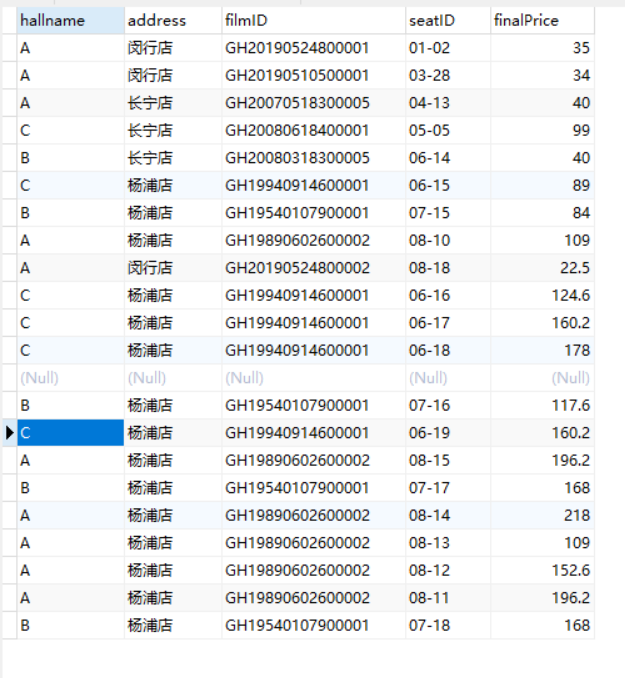
1. 按照上述说明在数据库中建立各个主键、外键、约束、默认。
2. 在数据库中建立一个名为member\_ticket的视图，该视图查询出每个会员的出票情况，包括放映厅名称、门店地址、电影名称、座位编号和会员票价。

DROP VIEW IF EXISTS member\_ticket;

CREATE VIEW member\_ticket AS

SELECT hallname,address,filmID,seatID,finalPrice

FROM memberinfo a LEFT OUTER JOIN ticketinfo b ON a.memberID=b.memberID LEFT OUTER JOIN playinfo c ON b.playID=c.playID LEFT OUTER JOIN cinema\_hallinfo d ON d.cinema\_hallID=c.cinema\_hallID



1. 为TicketInfo表建立一个插入触发器t\_insert\_ticketinfo，要求在出票信息表中插入某个出票信息（即会员购票）时，会依据memberID的会员折扣，自动计算出票的会员票价finalPrice。

DROP TRIGGER IF EXISTS t\_insert\_ticketinfo;

CREATE TRIGGER t\_insert\_ticketinfo BEFORE INSERT

ON ticketinfo FOR EACH ROW

BEGIN

DECLARE final DECIMAL(10,2) DEFAULT 0;

DECLARE ori DECIMAL(10,2) DEFAULT 0;

DECLARE cou DECIMAL(10,2) DEFAULT 0;

SELECT disount INTO cou

FROM memberinfo

WHERE memberinfo.memberID = new.memberID;

SELECT price\*cou INTO final

FROM playinfo

WHERE playinfo.playID=new.playID;

UPDATE ticketinfo SET finalPrice = final WHERE seatID=new.seatID AND playID=new.playID AND memberID=new.memberID;

END;

1. 为TicketInfo表建立一个更新触发器t\_update\_ticketinfo，当会员观看完电影之后，在出票信息表中更新某个电影评分（grade）时，依据更新后的会员评分grade自动计算出FilmInfo表中的电影平均评分filmGrade。

DROP TRIGGER IF EXISTS t\_insert\_ticketinfo;

CREATE TRIGGER t\_insert\_ticketinfo BEFORE INSERT

ON ticketinfo FOR EACH ROW

BEGIN

DECLARE num INT DEFAULT 0;

DECLARE sum0 DECIMAL DEFAULT 0;

DECLARE str VARCHAR(20) DEFAULT'';

SELECT COUNT(\*) INTO num

FROM ticketinfo

WHERE playID = new.playID;

SELECT SUM(grade) INTO sum0

FROM ticketinfo

WHERE playID = new.playID;

SET num = num+1;

SET sum0 = sum0+new.grade;

SELECT filmID INTO str

FROM playinfo

WHERE playID = new.playID;

SET sum0 = sum0/num;

UPDATE filminfo SET filmGrade = sum0 WHERE filmID=str;

END;

**保存所有操作之后**，在Navicat中选中数据库filmdb，右键选择【转储SQL文件…】，文件名设置为**filmdb\_学号**，单击【保存】，导出成功后，点击【关闭】。