《数据库系统》课程设计报告

题目	汽车销售管理系统		
小组成员信息			
姓名	**学号**	**班级**	**分工**
刘伟东	16340155	软工四班	数据库设计，页面实现
刘笑	16340156	软工四班	数据库实现，页面设计

提交时间：　　2018 年　7 月　7 日

一．开发环境与开发工具

Windows10，MySQL Workbench6.3CE，node.js，Vue，VSCode 编辑软件。

二．系统需求分析

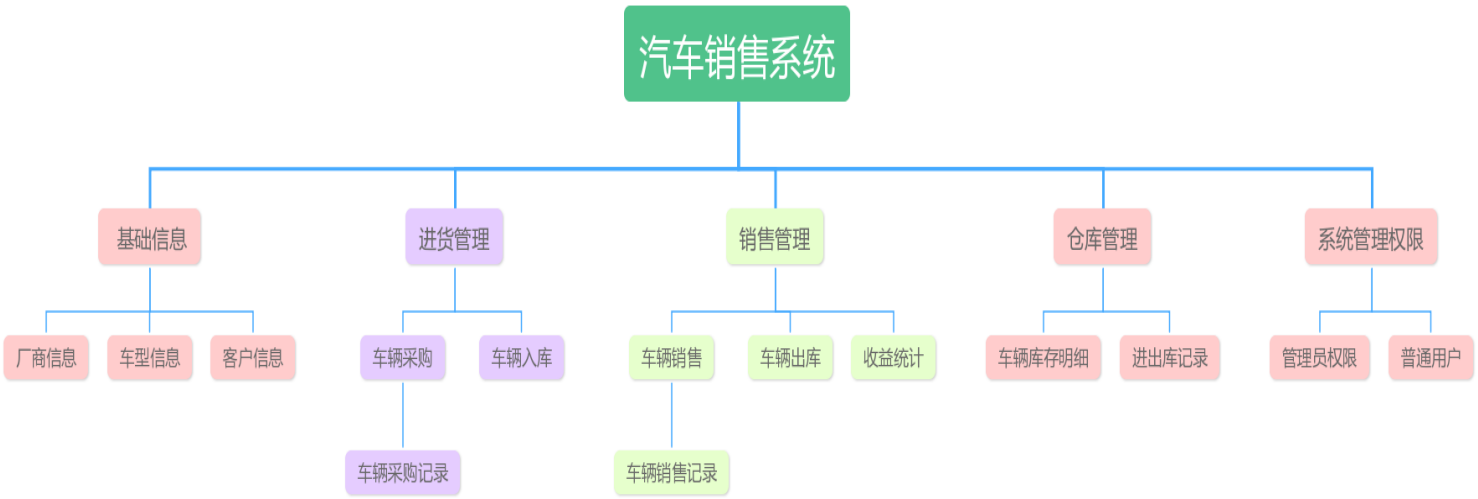
系统数据字典

TABLE_NAME	COLUMN_NAME	DATA_TYPE	COMMIT
FactoryInfo (主码-FName)	FName	Char	工厂名
CarInfo （主码-Cname） （外码-Fname）	CName	Char	车辆名
	CType	Char	车辆类型
	CPrice	Char	车辆价格
	FName	Char	工厂名
CustomerInfo （主码-CusID）	CusID	int	用户 ID
	CusName	char	用户名
	CusPhone	char	用户手机号
CarInventory (主码-CName) (外码-CName)	CName	char	车辆名
	CarNum	int	库存数量
PurchaseInfo (主码-PID) (外码-Cname)	PID	int	进货记录 ID
	CName	char	车辆名
	CNum	int	进货数量
	PPrice	int	进货价格
	PDate	Datetime	进货时间
InAndOutInfo （主码-InAndOutID） （外码-CName）	InAndOutID	int	入库出库 ID
	CName	char	车辆名
	InAndOutNum	int	入库出库数量

	InAndOutType	char	入库出库类型
	InAndOutDate	Datetime	入库出库时间
SaleInfo （主码-SID） （外码-CName，CusID）	SID	int	销售 ID
	CName	char	车辆名
	SNum	int	销售数量
	SDate	Datetime	销售时间
	SPrice	int	销售价格
	CusID	int	用户 ID
users （主码-username）	username	char	用户名
	password	char	密码
	status	int	权限

三. 功能需求分析

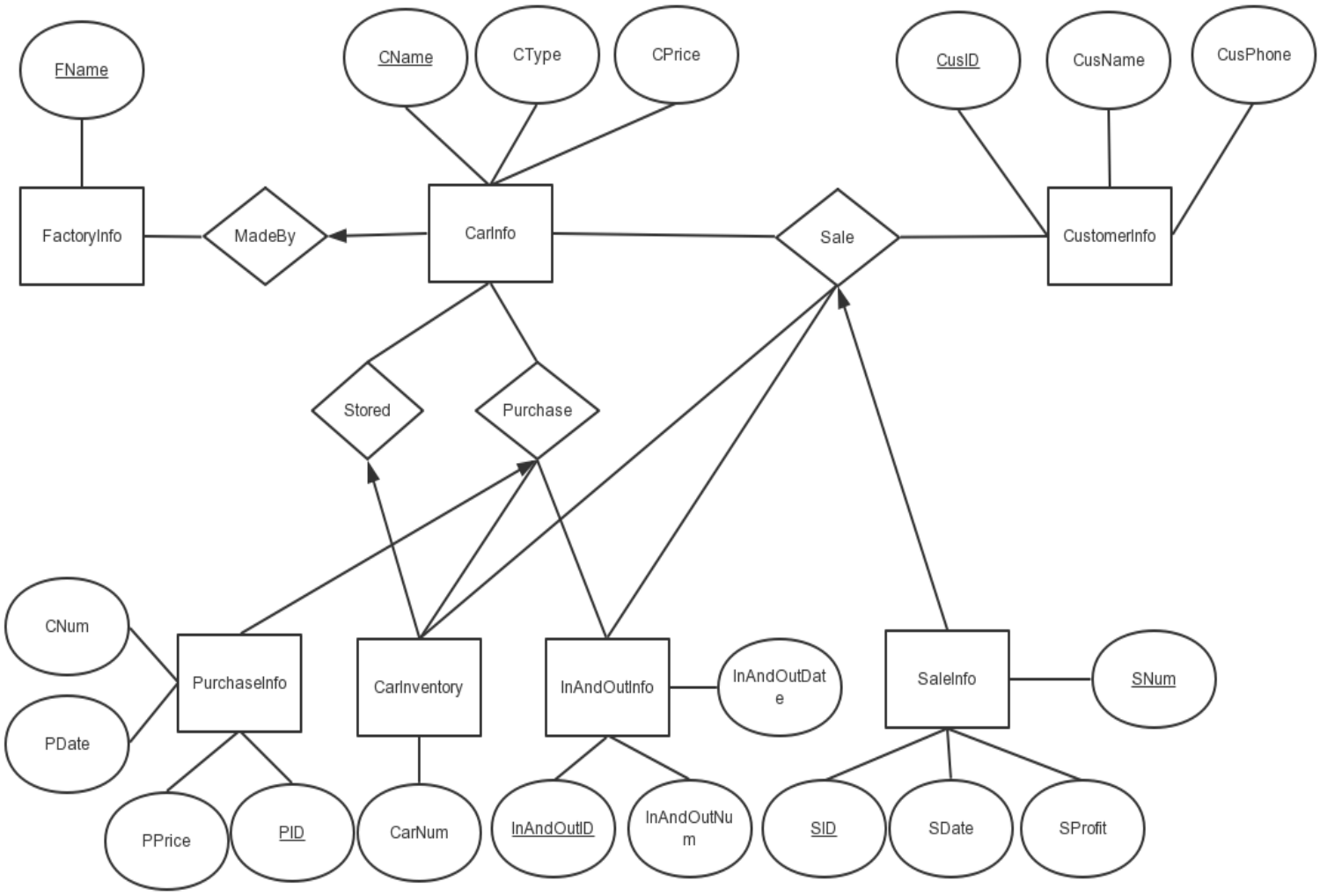
系统功能模块图



1. 基础信息模块包含了三个基础信息模块，分别是厂商信息、车型信息、客户信息。
2. 进货管理涉及车辆采购和车辆入库，采购会产生车辆采购记录。
3. 销售管理涉及车辆销售、车辆出库以及统计收益，以及维护一个车辆销售记录。
4. 仓库管理涉及车辆当前库存明细，以及维护一个进出库记录。
5. 系统管理权限分为管理员和普通用户，普通用户只能销售车辆和添加顾客。

四. 系统设计（10 分）

数据概念结构设计（系统 ER 图）



其中 Sale 销售关系有 SaleInfo 销售记录、InAndOutInfo 库存出入记录和 CarInventory 库存三个实体参与，销售记录是完全参与。Purchase 关系涉及 PurchaseInfo 进货记录，CarInventory 库存和 InAndOutInfo 入库出库记录，其中 PurchaseInfo 进货记录是完全参与。存储关系关联车辆库存实体，车辆库存实体完全参与这个关系。

数据库关系模式设计

1. **关系模式：**
   a. 工厂信息表（工厂名） 其中“工厂名”为主码。
   b. 车辆信息表（车辆名、车辆类型、车辆价格、工厂名）其中“车辆名”为主码，“工厂名”为生产关系的外码关联“工厂信息表”。
   c. 用户信息表（用户 ID、用户名、用户手机号）其中“用户 ID”为主码。
   d. 进货信息表（进货记录 ID、车辆名、进货数量，进货时间，进货价格）其中“进货记录 ID”是主码，车辆名是外码关联“车辆信息表”。

e. 车辆库存表（车辆名、车辆数量）“车辆名”是主码，且是外码关联“车辆信息表”。
f. 入库出库表（入库出库记录 ID、车辆名、入库出库数量、入库出库类型，入库出库时间）其中“入库出库 ID”是主码，“车辆名”是外码关联“车辆信息表”。
g. 销售记录表（销售记录 ID、车辆名、销售数量、销售时间、销售价格、用户 ID）其中“销售记录 ID”为主码，“车辆名”是外码关联“车辆信息表”，“用户 ID”为外码关联“用户信息表”。

2. 关系模式二维表：

a. 工厂信息表：

字段	字段名	数据类型	完整性约束
FName	工厂名	Char(10)	Primary Key

b. 车辆信息表：

字段	字段名	数据类型	完整性约束
CName	车辆名	Char(10)	Primary Key
CType	车辆类型	Char(10)	
CPrice	车辆价格	Char(10)	Not Null
FName	工厂名	Char(10)	Foreign Key

c. 用户信息表：

字段	字段名	数据类型	完整性约束
CusID	用户 ID	Int	Primary Key
CusName	用户名	Char(10)	Not Null
CusPhone	用户手机号	Char(20)	

d. 进货信息表：

字段	字段名	数据类型	完整性约束
PID	进货记录 ID	Int	Primary Key
CName	车辆名	Char(10)	Foreign Key
CNum	进货数量	Int	Not Null
PPrice	进货价格	Int	Not Null
PDate	进货时间	Datetime	CURRENT_TIMESTAMP

e. 车辆库存表：

字段	字段名	数据类型	完整性约束
CName	车辆名	Char(10)	Primary Key, Foreign Key
CarNum	库存数量	Int	Default 0 and CarNum >= 0

f. 入库出库表：

字段	字段名	数据类型	完整性约束
InAndOutID	入库出库 ID	Int	Primary Key
CName	车辆名	Char(10)	Foreign Key
InAndOutNum	入库出库数量	Int	Not Null
InAndOutType	入库出库类型	Char(10)	Not Null
InAndOutDate	入库出库时间	datetime	CURRENT_TIMESTAMP

g. 销售记录表：

字段	字段名	数据类型	完整性约束
SID	销售 ID	Int	Primary Key
CName	车辆名	Char(10)	Foreign Key
SNum	销售数量	Int(11)	Not Null
SDate	销售时间	Datetime	CURRENT_TIMESTAMP
SPrice	销售价格	Int	Not Null
CusID	用户 ID	int	Foreign Key

h. 管理员信息权限表：

字段	字段名	数据类型	完整性约束
Username	用户名	Char(20)	Primary Key
Password	密码	Char(20)	Not Null
Status	权限	Int	Default 0

3. 创建数据库的完整语句与约束：

```sql
create database if not exists CSMS;

CREATE USER 'csmsAdmin'@'localhost' IDENTIFIED BY 'sysu615@';
GRANT ALL ON csms.* TO 'csmsAdmin'@'localhost';

use CSMS;


-- 用户表
create table if not exists users (
    username char(20) not null,
    password char(20) not null,
    status int default 3,
    primary key(username)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- 工厂信息表
create table if not exists FactoryInfo(
    FName char(10) not null,
    primary key(FName)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;


insert into factoryinfo values ('通用汽车');

insert into factoryinfo values ('大众汽车');

insert into factoryinfo values ('福特汽车');

insert into factoryinfo values ('本田汽车');

insert into factoryinfo values ('丰田汽车');


-- 车辆信息表
create table if not exists CarInfo(
    CName char(10) not null,
    CType char(10),
    CPrice char(10) not null,
    FName char(10) not null,
    primary key(CName),
    foreign key (FName) references FactoryInfo(FName) on delete no
action
```

```
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

insert into carinfo values ('雪佛兰', '轿车', '250000','通用汽车');

insert into carinfo values ('别克', '轿车', '290000','通用汽车');

insert into carinfo values ('斯柯达', '轿车', '160000','大众汽车');

insert into carinfo values ('帕萨特', '轿车', '240000','大众汽车');

insert into carinfo values ('途昂', '越野车', '400000','大众汽车');

insert into carinfo values ('野马', '跑车', '500000','福特汽车');

insert into carinfo values ('F-150', '皮卡车', '600000','福特汽车');

insert into carinfo values ('思域', '轿车', '200000','本田汽车');

insert into carinfo values ('雅阁', '轿车', '200000','本田汽车');

insert into carinfo values ('卡罗拉', '轿车', '150000','丰田汽车');

insert into carinfo values ('凯美瑞', '轿车', '220000','丰田汽车');


-- 用户信息表
create table if not exists CustomerInfo(
    CusID int auto_increment,
    CusName char(10) not null,
    CusPhone char(20),
    primary key(CusID)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

insert into customerinfo values ('0','刘伟东','16340155');

insert into customerinfo values ('0','刘亚辉','16340157');

insert into customerinfo values ('0','刘宇庭','16340158');


-- 库存表
create table if not exists CarInventory(
```

```
    CName char(10) not null,
    CarNum int default 0,
    primary key(CName),
    foreign key(CName) references CarInfo(CName) on delete no action
)ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- 库存数量约束
delimiter //
create trigger carinventory before update on CarInventory
for each row
begin
if new.CarNum < 0 then
signal sqlstate '12345' set MESSAGE_TEXT = '库存不能小于0';
end if;
end;
//
delimiter ;


insert into CarInventory values ('雪佛兰', '20');

insert into CarInventory values ('别克', '10');

insert into CarInventory values ('斯柯达', '15');

insert into CarInventory values ('帕萨特', '20');

insert into CarInventory values ('途昂', '5');

insert into CarInventory values ('野马', '5');
insert into CarInventory values ('F-150', '3');
insert into CarInventory values ('思域', '25');

insert into CarInventory values ('雅阁', '35');

insert into CarInventory values ('卡罗拉', '14');

insert into CarInventory values ('凯美瑞', '21');



-- 进货
```

```
create table if not exists PurchaseInfo(
   PID int auto_increment,
   CName char(10) not null,
   CNum int not null,
   PPrice int not null, -- 进货单价
   PDate Datetime DEFAULT CURRENT_TIMESTAMP,
   primary key(PID),
   foreign key(CName) references CarInfo(CName) on delete no action
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 库存改变
create table if not exists InAndOutInfo(
   InAndOutID int auto_increment,
   CName char(10) not null,
   InAndOutNum int not null,
   InAndOutType char(10) not null,
   InAndOutDate Datetime DEFAULT CURRENT_TIMESTAMP,
   primary key(InAndOutID),
   foreign key(CName) references CarInfo(CName) on delete no action
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- 出货/用户购买信息
create table if not exists SaleInfo(
   SID int auto_increment,
   CName char(10) not null,
   SNum int not null,
   SDate Datetime DEFAULT CURRENT_TIMESTAMP,
   SPrice int not null,
   CusID int,
   primary key(SID),
   foreign key(CName) references CarInfo(CName) on delete no action,
   foreign key(CusID) references CustomerInfo(CusID) on delete no
action
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

DROP PROCEDURE IF EXISTS purchase;
DELIMITER //
CREATE PROCEDURE purchase(in param_name char(10), in param_num int, in
param_price int)
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
```

```
begin
rollback;
signal sqlstate '12346' set MESSAGE_TEXT = 'purchase EXCEPTION';
end;
START TRANSACTION;
insert into carinventory values (param_name,param_num) on duplicate key
update CarNum = CarNum + param_num;
insert into purchaseinfo (CName,CNum,PPrice) values
(param_name,param_num,param_price);
insert into inandoutinfo (CName,InAndOutNum,InAndOutType) values
(param_name,param_num, '入库');
select * from purchaseinfo;
COMMIT;
END//
DELIMITER ;

DROP PROCEDURE IF EXISTS sale;
DELIMITER //
CREATE PROCEDURE sale(in param_name char(10), in param_num int, in
param_price int, in param_cusid int)
BEGIN
DECLARE EXIT HANDLER FOR SQLEXCEPTION
begin
rollback;
signal sqlstate '12347' set MESSAGE_TEXT = 'sale EXCEPTION';
end;
START TRANSACTION;
update carinventory set CarNum = CarNum - param_num where CName =
param_name;
insert into saleinfo (CName,SNum,SProfit,CusID) values
(param_name,param_num,param_price,param_cusid);
insert into inandoutinfo (CName,InAndOutNum,InAndOutType)
values(param_name,param_num,'出库');
select * from saleinfo, customerinfo, carinfo
where saleinfo.CusID = customerinfo.CusID and saleinfo.CName =
carinfo.CName;
COMMIT;
END//
DELIMITER ;
```

数据库物理结构设计

物理设计是为逻辑数据模型选取一个包括存储结构和存取方法在内的最合适的应用环境的物理结构。

物理设计应该分为两步：

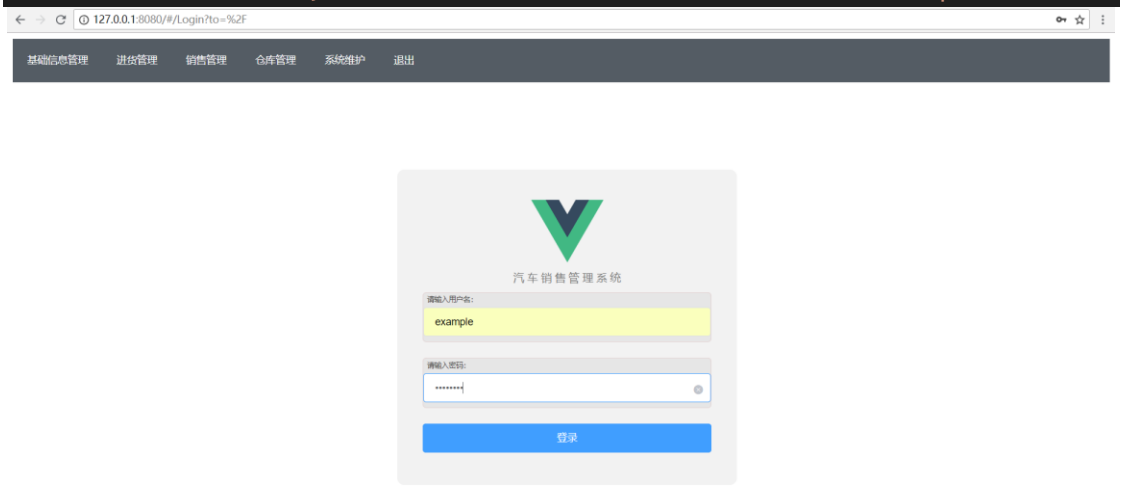
1. 确定数据库的物理就团购，在关系型数据库中指定存取方法和存储结构。
2. 对物理结构进行性能检查，评价重点为时间和空间效率。

此次课程设计数据量不大，MySQL 默认的存储结构可以很好地满足要求，所以没有调整，仅仅使用 InnoDB 数据引擎来保证操作地原子性。

五. 系统功能的实现 (5 分)

1. 登录界面：将查询 users 表，使用语句

```
'select username, status from users where username=? and password=?'
```



2. 登录之后，将导航带**销售管理**界面，并查询表 saleinfo，使用语句：

```
'select * from ??'
```

3.点击“卖车”按钮后，将出现下面的场景，车名是通过上述语句查询表 carinfo 得到的可选列表，购买数量不能为负，顾客姓名是存在表 customerinfo 中的人，可通过另一个页面添加顾客。

使用下面语句：

```
"call sale(?,?,?,?)"
```

这个语句将调用前面设定好的流程，完成对 carinventory，saleinfo，inandoutinfo 表的插入，并查询 saleinfo。

4. 下面是**基础信息管理界面**，在这个界面中，可以在顾客信息中添加顾客：

```
'insert into customerinfo values(0, ?, ?)'
```

车型信息将查询表 carinfo：

```
'select * from ??'
```

厂商信息将查询表 factoryinfo

5. 进货管理：这个页面需要用户权限，普通用户只能查看记录，root 用户才能更改，查询语句和上相同。

使用 root 用户登陆后，效果如上图。

这里使用下述语句插入记录：

```
CALL purchase(?,?,?)
```

这个语句将调用前面设定好的流程，完成对 carinventory，purchaseinfo，inandoutinfo 表的插入，并查询 purchaseinfo。

6. **仓库管理**：这个界面会查询 carinventory

车名	库存
F-150	3
凯美瑞	21
别克	15
卡罗拉	12
帕萨特	20
思域	25
斯柯达	15
途昂	5
野马	5
雅阁	35
雪佛兰	20

7. **系统维护**：这个界面能够查看权限低于当前用户的所有用户和入库出库操作记录。

用户列表

用户名	权限
administor	root
example	普通用户
root	root

操作日志

操作记录	车名	操作时间	入库/出库	数量
1	别克	2018-07-08 06:40:18	入库	5
2	卡罗拉	2018-07-08 06:44:40	出库	2

六. 总结

在这次课程设计的实验中，我们首先讨论确认了数据库的基本实体和关系，然后再绘制 ER 图，并进行简化和确认关系。其中用到了二元关系和约束将实体之间通过关系有效地结合起来。

然后再开始具体实现数据库创建代码，此时充分考虑了主码，完整性约束和通过外码约束来实现关系，并且使用 InnoDB 来实现事物的原子性。最后首先在 workbench 上对数据库的各种设计进行检验修改。

接下来是选择了 WEB 来开发此应用，首先在一个文件中实现所有需要的数据库操作，然后再实现各个页面并调用这些操作，并且需要考虑前端的设计布局。

最后是进行各种操作测试，检查 BUG 并处理。这次实验时间有限，所以在只是实现了各种初步的功能。但是在这个过程中我们不仅回顾了之前学习的数据库知识和 MySQL 语句，而且更加深入地理解了数据库与应用地有机结合，学习了数据库地结构化和流程化设计，受益匪浅。