Coupay System Design Document Series

Database Design Specification

1 Incipit

1.1 Intention

This specification of database design describes the design process and structure of the database in detail. It is designed to help developers develop the system. It also provides a good instruction for the follow-up maintaining work of the database . In addition , it would be an important reference when the system is to be updated.

1.2 Background

Name of the database: CoupayDatabase
The system that uses this database: Coupay
Sponsor: Citibank, N.A.
User: developers, system managers
DBMS: MySql community server 5.5.29

1.2 Definition

register: The register users of the Coupay system,including merchant and customer.

account: After registering,users would have an system-auto-created account which can be used by the registers for deposit and transaction.

bindingBankAccount:registers can bind their account in banks which have business cooperation with the Coupay system with their account of the Coupay system for the convenience of future deposit and transactions.

transaction: Registers can transfer their money of the Coupay account to other registers' Coupay account,or they can transfer their money of their binding bankAccount to other registers' Coupay Account instead.

productInformation: If customers' transactions are used for payment of some products, that can query the information of the products they bought afterword.The product information is also used by the Coupay system for data minding.

Coupon:The merchant that have business cooperation with the Coupay system can release some coupons which will be collected by customers.Some coupons are used for discount and others for trade-in.With coupons,customer can go to the merchants that release them for consumption or they can give them to their friends in the circle as gifts instead.

vip-Card: The merchant that have business cooperation with the Coupay system can release some vip-cards.Customers can apply for the vip-cards for the privilege of their consumption.

sharing: After consumption and have transaction handled,customers can share their

consuming experience with their friends in the circle by giving comments to the products and recommending products.

friends in the circle: customers can follow each other within the Coupay System so that the can see the sharing of each other and can send coupons to each other.

followed merchant: customers can follow the merchants they are interested in within the Coupay System so as to focus on the dynamic of the merchants or to gain opportunity for their coupons and vip-cards.

message: In some cases, the Coupay system needs to send messages to the registers.For instance,a message will be sent to registers whose accounts have just received a transaction.

1.4 References

2 External design

2.1 Identifier and status

Database Identifier: CoupayDatabase
user: admin
password: admin
status: develop mode.

2.2 Target System

This database is specially designed for Coupay.

2.3 Agreement

All data items are represented in CamelCase. The selected charset is utf-8.

2.4 Supporting software

MySql community Server 5.5.29.

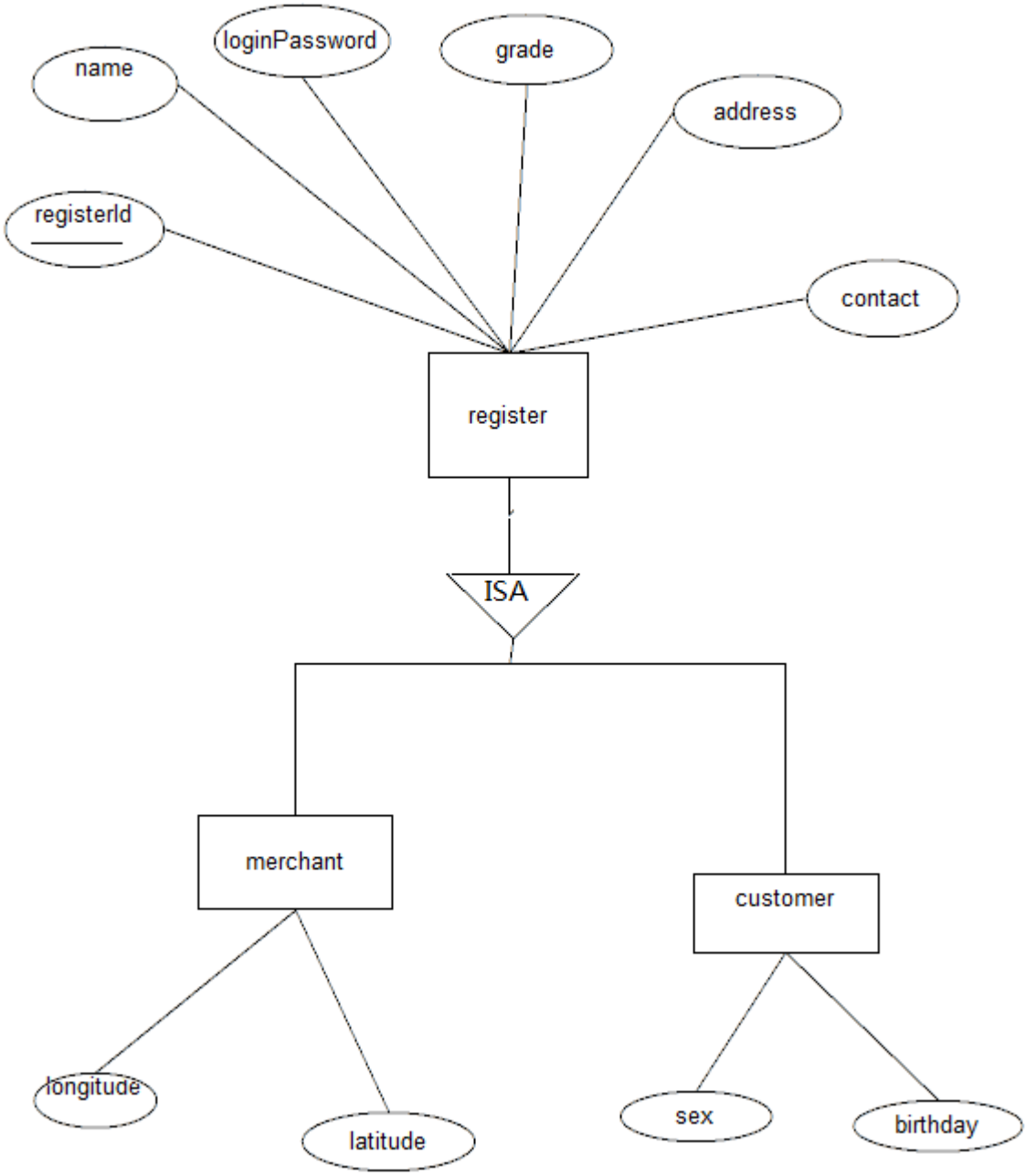
MySql WorkBench.

3 Structure design

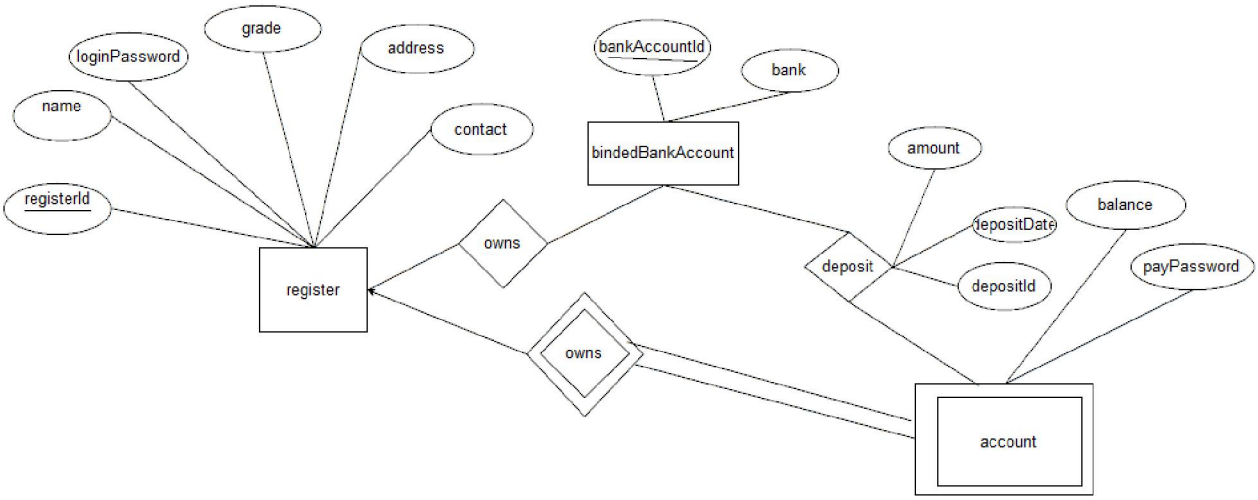
3.1 Conceptual structure design

3.1.1 partial view

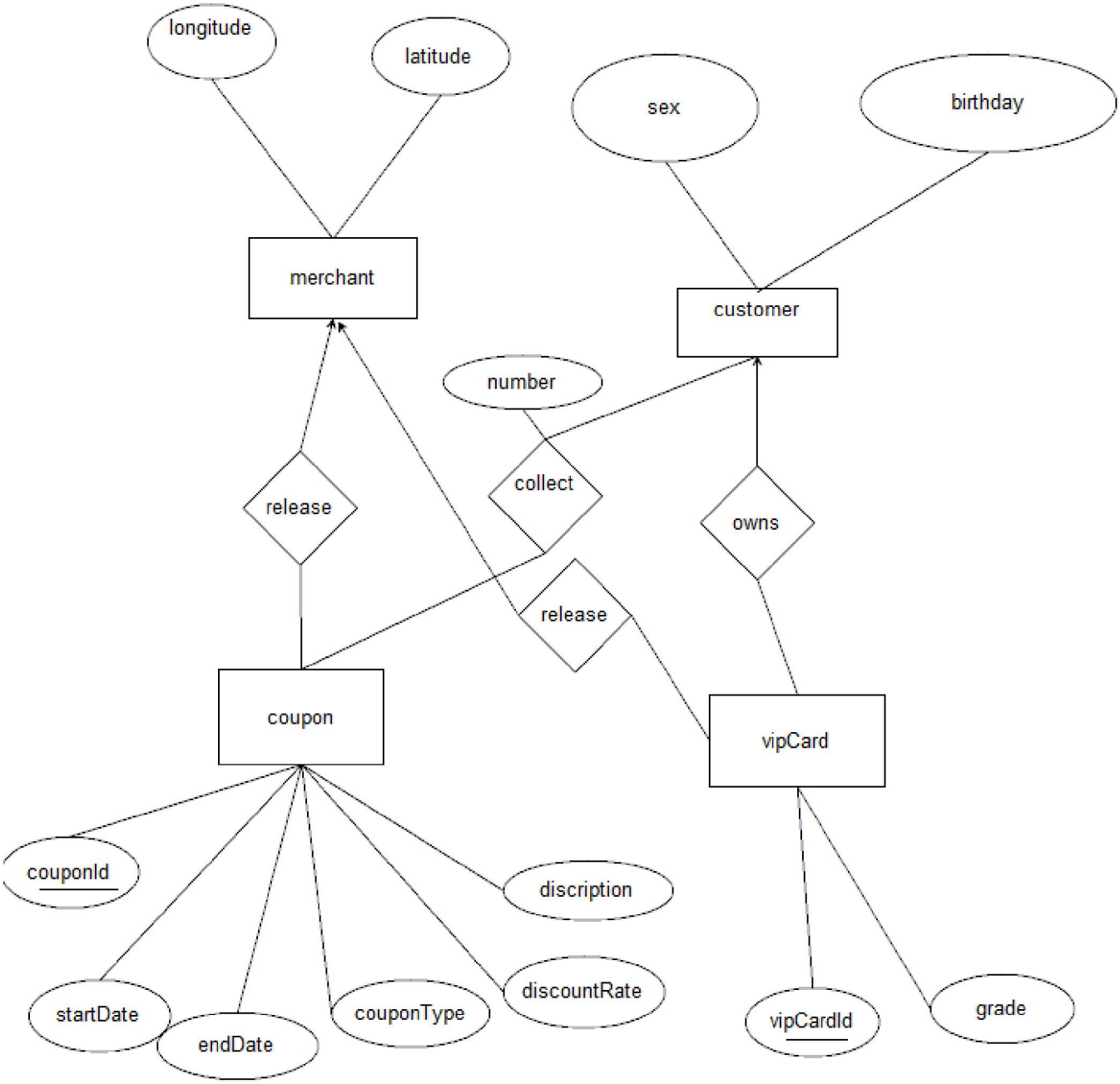
basic message of registers:



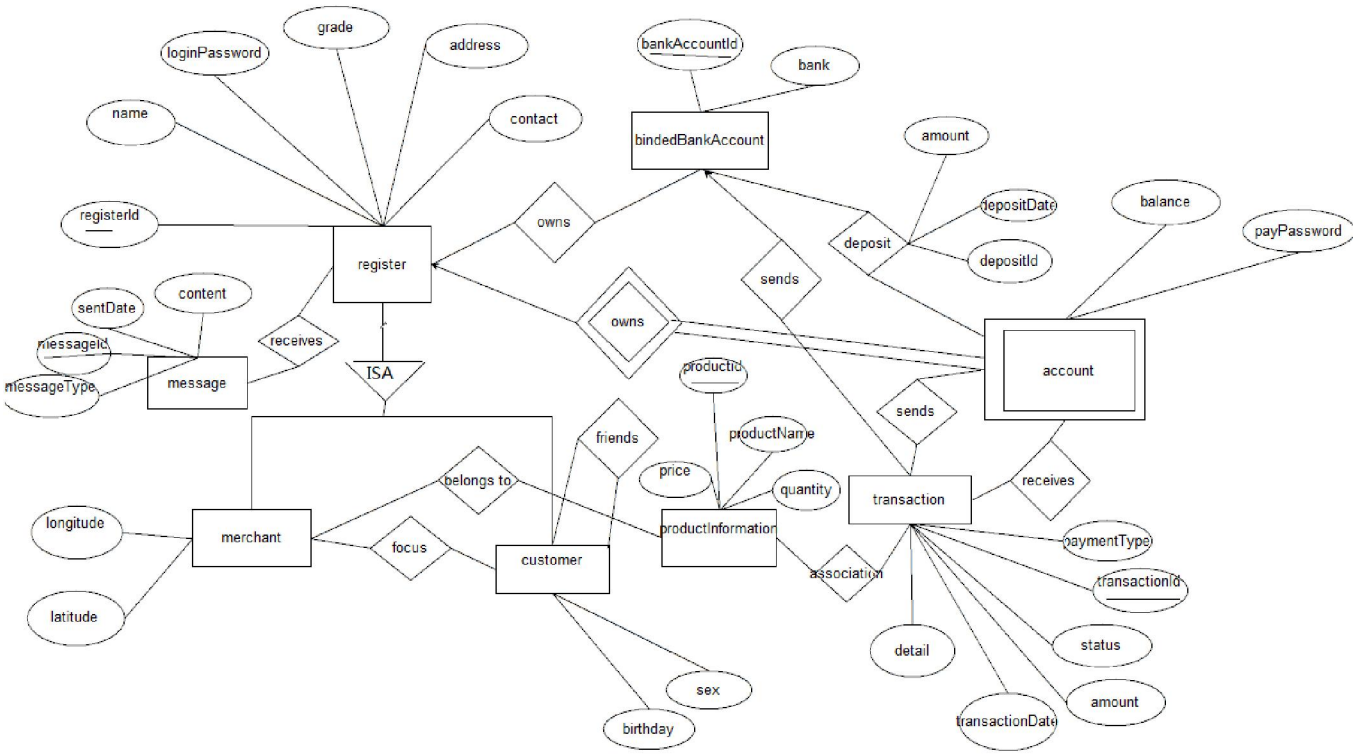
management of account:



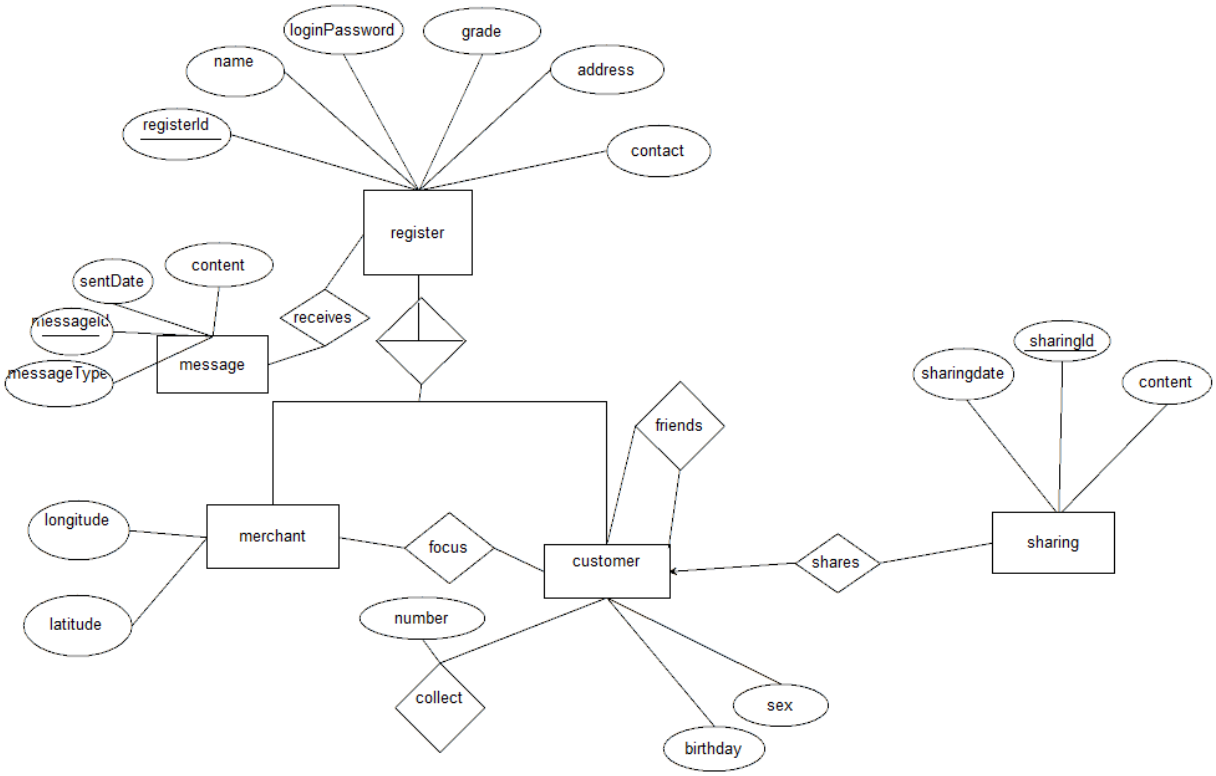
management of coupon and vipCard:



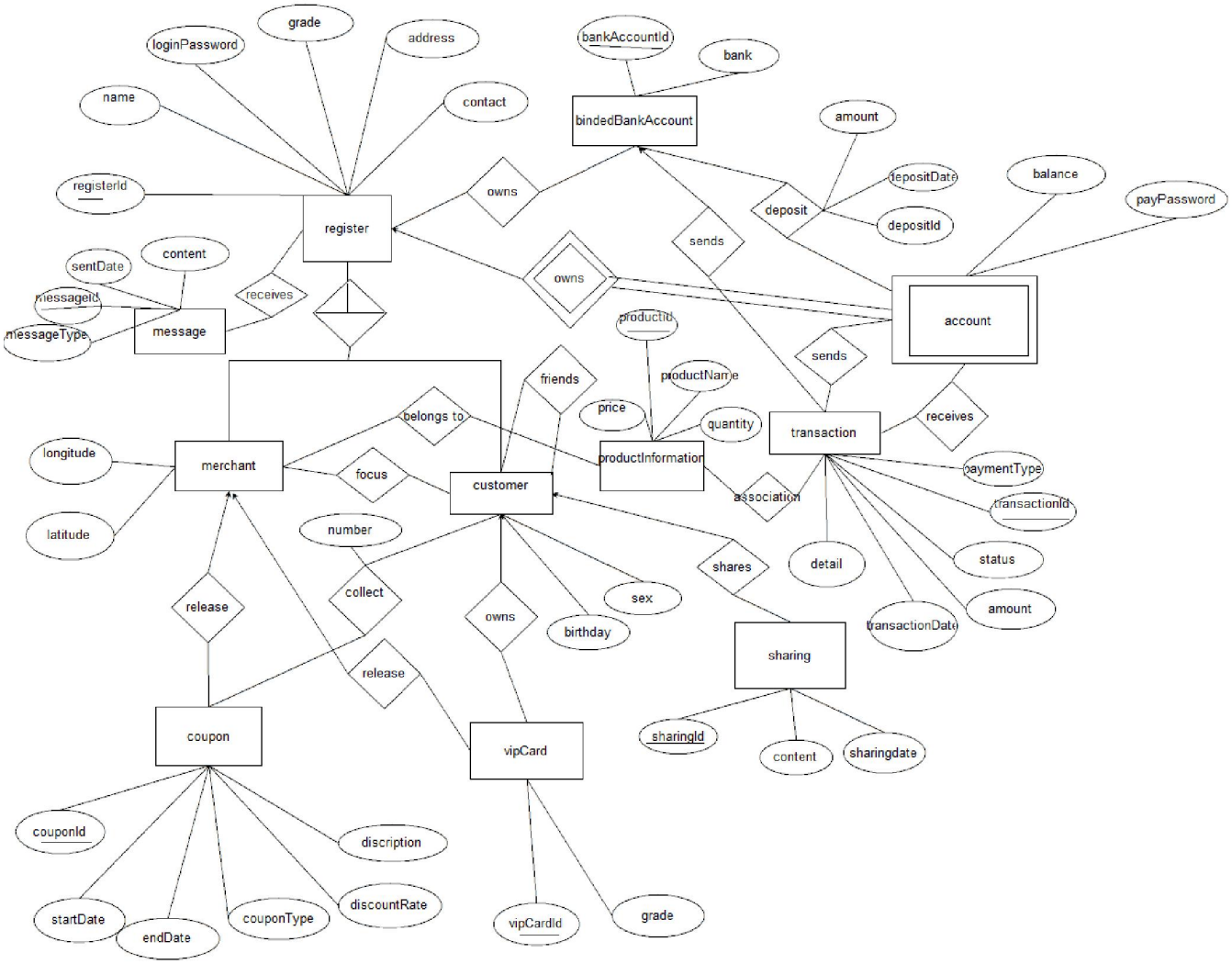
management of transaction :



management of the community:



3.1.2 Integrated view



3.2 Logical structure design

Relation schema:

register（registerId，name, email,phoneNumber, address, loginPassword, grade）

merchant (merchantId,longitude,latitude)

customer(customerId, sex, birthday)

account (accountId, payPassword，balance)

bindedBankAccount（bankAccountId ,registerId，bank）

depositRecord (depositeId,account,bankAccount,amount,depositDate)

coupon (couponId,merchantId,startDate,endDate,couponType,discountRate,description)

couponCollection(customerId,couponId,number)

vipCard (vipCardId, merchantId, ownerId, grade)

transactionRecord(transactionId,sender,receiver,amount,transactionDate,transactionType, detail)

transactionBankAccount（transactionId,bankAccountId）
productInfomation（productId, associatedTransactionId, productName, price, quantity）
sharing (sharingId,sharerId,content,sharingDate)
message （messageId，registerId，content，messageType）
friendship（customerId1，customerId2）
followedMerchant(merchantId,customerId)

Views:
CustomerBasicInformation(registerId,name,email,phoneNumber,address,grade,sex,birthday)
CustomerInportantInformation(registerId,loginPassword,payPassword,balance)
MerchantBasicInformation(registterId,name,email,phoneNumbe,address,grade,longitude, latitude)
MerchantInportantInformation(registerId,loginPassword,payPassword)
CustomerCouponCollection(customerId,merchantId,couponId,number,startDate,endDate, couponType,discountRate)

Sql statements:

```
use coupon;
create table register (
    registerId int auto_increment primary key,
    name varchar(30) not null,
    loginPasswor char(128) not null,
    email char(40),
    phoneNumber varchar(40),
    address varchar(40),
    grade int not null
);

create table customer (
    customerId int,
    sex char(1) check (sex in ('男' , '女')),
    birthday date,
    foreign key(customerId) references register(registerId)
);

create table merchant (
    merchantId int,
    longitude decimal(9 , 6 ),
    latitude decimal(9 , 6 ),
    foreign key(merchantId) references register(registerId)
);
```

```
create table account (
	accountId int primary key,
	foreign key (accountId)
		references register (registerId),
	balance double (10 , 2 ) check (balance >= 0),
	payPassword char(128) not null
);

create table bindedBankAccount (
	bankAccountId varchar(25) primary key,
	registerId int,
	foreign key (registerId)
		references register (registerId),
	bank varchar(30) not null
);

create table depositRecord (
	depositId int auto_increment primary key,
	bankAccountId varchar(25),
	amount double (10 , 2 ) check (amount > 0),
	depositDate date not null,
	foreign key (bankAccountId)
		references bindedBankAccount (bankAccountId)
);

create table coupon (
	couponId int auto_increment primary key,
	merchantId int,
	startDate date,
	endDate date not null,
	couponType varchar(10) not null,
	discountRate double (10 , 2 ),
	decription varchar(200),
	foreign key (merchantId)
		references merchant (merchantId)
);

create table couponCollection (
	customerId int,
	couponId int,
	number int,
	primary key (customerId , couponId),
	foreign key (customerId)
```

```
            references customer (customerId),
        foreign key (couponId)
            references coupon (couponId)
);

create table vipCard (
        cardId varchar(25) primary key,
        merchantId int,
        customerId int,
        grade int,
        foreign key (merchantId)
            references merchant (merchantId),
        foreign key (customerId)
            references customer (customerId)
);

create table transactionRecord (
        transactionId char(22) primary key,
        sender int,
        receiver int,
        amount double (10 , 2 ) not null,
        transactionDate datetime not null,
        transactionType varchar(15) not null,
        detail text,
        foreign key (sender)
            references register (registerId),
        foreign key (receiver)
            references register (registerId)
);

create table productInfomation (
        productId varchar(20),
        transactionId char(22),
        primary key (productId , transactionId),
        foreign key (transactionId)
            references transactionRecord (transactionId)
);

create table sharing (
        sharingId int auto_increment primary key,
        shareId int,
        content varchar(140) not null,
        sharingDate datetime not null,
        foreign key (shareId)
```

```
        references customer (customerId)
);

create table message (
    messageId int auto_increment primary key,
    receiverId int,
    content varchar(200) not null,
    messageType varchar(15) not null,
    foreign key (receiverId)
        references register (registerId)
);

create table friendship (
    customerId1 int,
    customerId2 int,
    primary key (customerId1 , customerId2),
    foreign key (customerId1)
        references customer (customerId),
    foreign key (customerId2)
        references customer (customerId)
);

create table followdMerchant (
    merchantId int,
    customerId int,
    primary key (merchantId , customerId),
    foreign key (merchantId)
        references merchant (merchantId),
    foreign key (customerId)
        references customer (customerId)
);
```

3.3 physical structure design

Creating indexes：
All the tables already have primary-key-index.In addition,after considering the query condition of the data, index should be created on transactionRecor table and sharing table.Sql-statements are as following.

```
create index transactionDateIndex on transactionRecord(transactionDate desc);

create index sharingDateIndex on sharing(sharingDate desc)
```

4. Practical design

4.1 data directory design

register table：

dataItem identifier	**data type**	**constraint**	**remark**
registerId	int	primary key	auto-increment
name	varchar(30)	not null	
loginPassword	char(128)	not null	encrypted using md5
email	varchar(40)		
phoneNumber	varchar(20)		
grade	int	not null	
address	varchar(40)		

merchant table：

dataItem identifier	**data type**	**constraint**	**remark**
merchantId	Int	foreign key primary key	references register table
longitude	decimal(9,6)		accurate to six decimal places
latitude	decimal(9,6)		accurate to six decimal places

customer table：

dataItem identifier	**data type**	**constraint**	**remark**
customerId	Int	foreign key primary key	references register table
sex	char(1)	check in（’男’,’女’）	
birthday	date		

account table：

dataItem identifier	**data type**	**constraint**	**remark**

accountId	Int	foreign key	references register table
balance	double(12,2)	not null check (amount >= 0.00)	balance must not be less than zero.
payPassword	char(128)	not null	encrypted using md5

bindedBankAccount table：

dataItem identifier	**data type**	**constraint**	**remark**
registered	int	foreign key primary key	references register table part of composite key
bankAccountId	varchar(25)	not null	references bindedBankAccount table part of composite key
bank	varchar(30)	非空	

depositRecord table：

dataItem identifier	**data type**	**constraint**	**remark**
depositId	int	primary key	auto-increment
accountId	Int	foreign key	references account table
bankAccountId	varchar(25)	foreign key	references bindedBankAccount table
amount	double(10,2)	check(amount >0)	it should be greater than zero
depositDate	date	not null	

coupon table

dataItem identifier	**data type**	**constraint**	**remark**
couponId	int	primary key	auto-increment
merchantId	int	foreign key	references merchant table
startDate	date		
endDate	date	not null	head time of the period of validity
couponType	varchar(10)	not null	
discountRate	double(10,2)	not null	
Description	varchar(200)		brief description of the coupon

couponCollection table

dataItem identifier	**data type**	**constraint**	**remark**
couponId	int	foreign key	references coupon table

customerId	int	foreign key	references customer table
number	int	not null	

vipCard table:

dataItem identifier	**data type**	**constraint**	**remark**
cardId	varchar(25)	primary key	
merchantId	int	foreign key	references merchant table
customerId	int	foreign key	references customer table
grade	int	not null	

transactionRecord table：

dataItem identifier	**data type**	**constraint**	**remark**
transactionId	char(22)	primary key	generated by the system
sender	int	foreign key	references account table
receiver	int	foreign key	references register table
amount	double(10,2)	not null	
transactionDate	datetime	not null	
transactionType	varchar(15)	not null	
detail	text		detail information of the transaction

transactionBankAccount table:

dataItem identifier	**data type**	**constraint**	**remark**
transactionId	char(22)	foreign key primary key	references transactionRecord table
bankAccountId	varchar(25)	foreign key	references bindedBankAccount table

productInfomation table

dataItem identifier	**data type**	**constraint**	**remark**
productId	varchar(20)	primary key,	
transactionId	char(22)	foreign key, primary key	references transaction table
productName	varchar(25)	not null	
price	double(10,2)	not null	
quantity	double(10,2)	not null	

sharing table:

dataItem identifier	**data type**	**constraint**	**remark**
sharingId	int	primary key	auto-increment
sharerId	int	not null	
content	varchar(140)	not null	
sharingDate	datetime	not null	

message table：

dataItem identifier	**data type**	**constraint**	**remark**
messageId	int	primary key	auto-increment
receiverId	int	foreign key	references register table
content	varchar(200)	not null	
messageType	varcahr(15)	not null	

friendship tale：

dataItem identifier	**data type**	**constraint**	**remark**
customerId1	int	foreign key primary key	references customer table part of composite key
customerId2	int	foreign key primary key	references customer table part of composite key

followedMerchant table：

dataItem identifier	**data type**	**constraint**	**remark**
customerId	int	foreign key primary key	references customer table part of composite key
merchantId	int	foreign key primary key	references merchant table part of composite key

4.2 Security and Secrecy

Ordinary system visitors access the data through the system. Programs are designed to manage the authority of the visitors. Visitors are allowed to access the data only after they are authorized. Password are used to check visitors' authority. The system would keep track of visitors' CRUD

operations on the data they are accessing to.

For system managers, different views would be created and granted to different managers. They are also granted different authorities to operate on the data. System managers also need to pass the password authentication to access to the data.