2015

Final Reports
Mini-Blog System

DATABASE FOR MINI-BLOG

TEAM SIRIUS

TEAM MEMBER: WANLI MA
NAIZHEN LIU
QIAOYI HE
ZITONG LIU
LU JU

CONTENTS

PROJECT DESCRIPTION

The mini-blog is a blog system which provides platform for people to communicate with each other. Members could post messages and photos, leave messages to their friends or celebrities, and explore the trending currently. There are various entities involved in the system such as members, celebrity, and account manager.

Users could only browse mini-blog by not being a member. If users want to leave comments for someone, she/he has to be member of the system. Users could register to be a member of mini-blog. When you become a member, you could post articles, short sentences, and photos and leave message and comments for others (he/she could not be your friend). Members could pay to be VIP members that have more privileges VIP members could design their own pages, and etc. If the user is a celebrity or the user represents an enterprise, he/she could apply to be an authentic user. An authentic user has a unique "v" near their user name. Only authentic users could hold activities. All registered user could play games shows on the pages, some of which are paid games. Registered user could attend activities showed on the page.

Every time a registered user login the system, the user will be asked whether to show his/her address. If the user choose yes, this login location will be stored in the system. Registered user could see the user's own location history as well as other user's shared location. Every time a user post messages, location will also be stored in the system.

In order to make the network environment suitable for everyone, management system is an important role in the whole system. Management system includes four different managers, which are advertisement manager, message manager, order manager, and application manager. All message managers monitor messages sent by members. Once these is anything illegal shows up, the message manager could delete the message. Application managers also deal with applications turned in by users.

Technical staffs is responsible for maintain the stable of the whole system, doing backups, fixing bugs existed in the system. Technical staffs also take charge of exploring new features for the system.

ROLES AND RESPONSIBILITIES

The business functions of mini-blog is to develop new features and attract more users. At this point, the major business functions that shows up are:

1) Members activities
2) Account manager supervision
3) Financial management
4) Developer works

The company has been able to break each of these high level business functions in the list as shown below.

Visitors

1) Register to be a member
2) Pay to be VIP member
3) Apply to by authentic user

Registered User

1) Post articles, photos, comments
2) Play games showed on the page

VIP User

1) Design their own page

Authentic user

1) Hold activities

Message manager

1) Delete illegal comments
2) Deal with members' problems

Application manager

1) Check the user information
2) Change the user's Status
3) Review user's application
4) Add a special status user

Advertise manager

1) Deal with any problems in advertising

Order manager

1) Handle user orders
2) Change the user's privilege
3) Add new VIP to the system

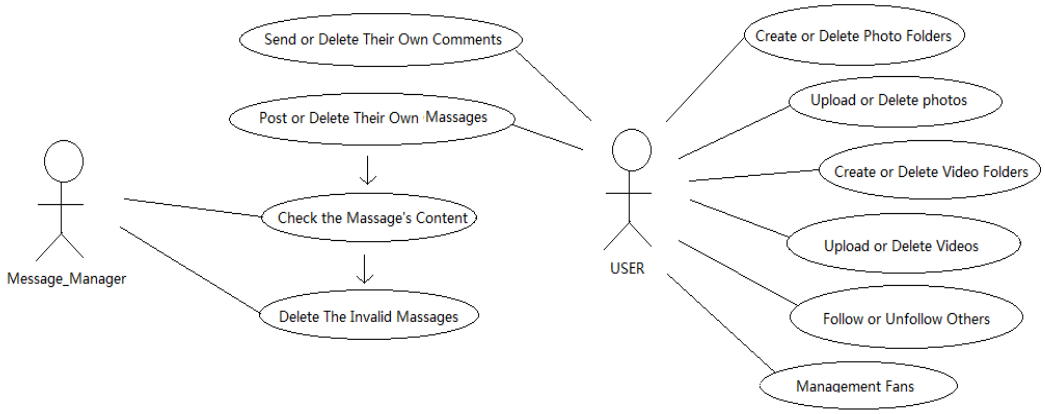
Developer

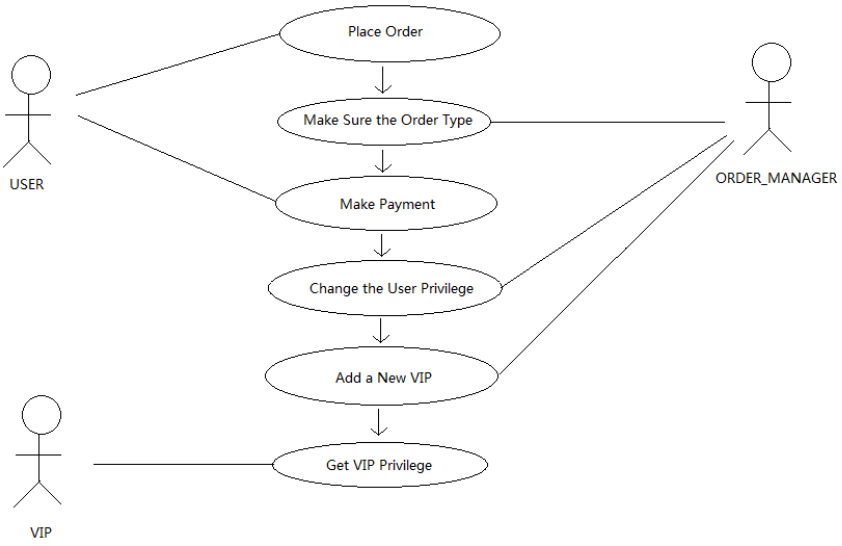
1) Maintain the system
2) Develop new features for the system

Financial department

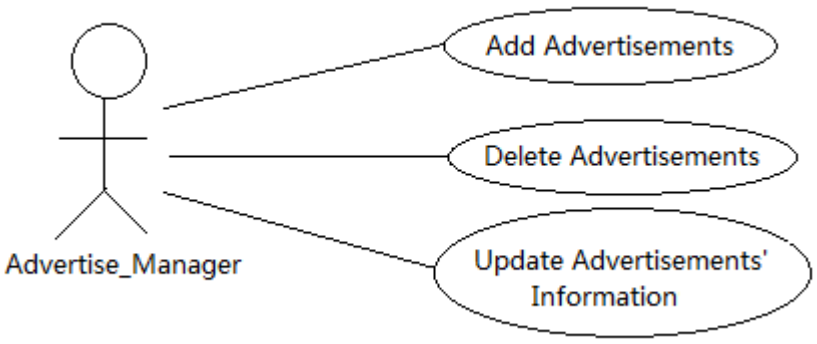
1) Payroll
2) Advertisement income
3) VIP fees

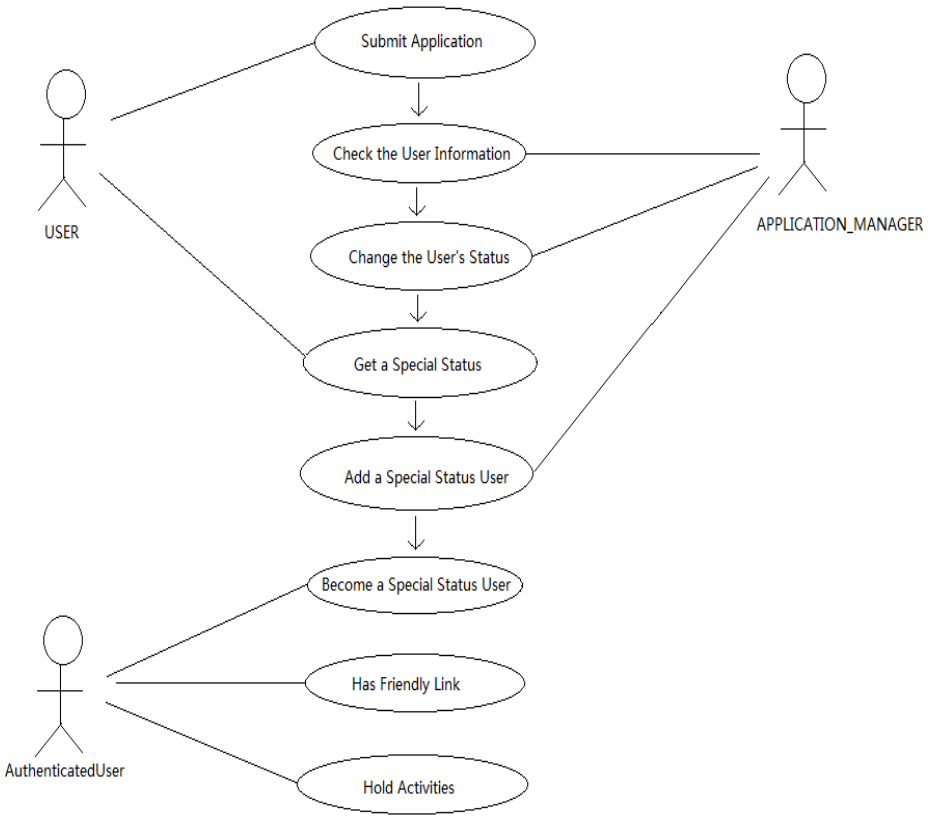
USE CASE DIAGRAM



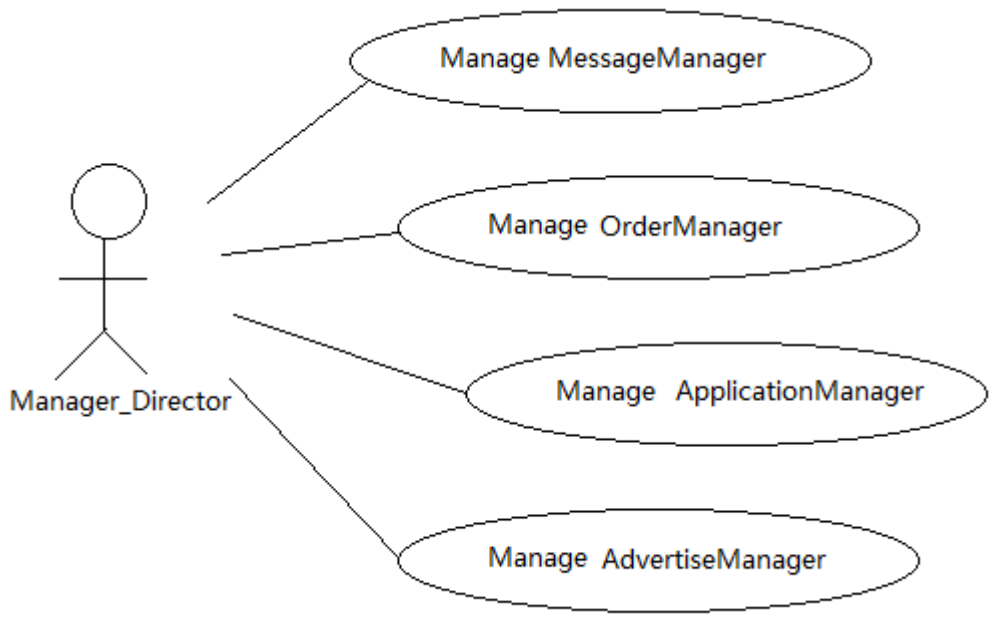


Order Manager in charges of any order placed by user. After reviewing these orders, order manager could create a new VIP id for user. By doing this, a user could be a VIP user.





Application Manager reviews applications handed in by user.



ENTERPRISE MODELING

We identified a preliminary set of 15 entity types that describe the data that needs to be stored in the database, which are: User administrant, General User, Account, Location History, VIP, Authenticated User, Order, Application, VIP Privilege, Article/Video/Photo, Comment, Activities, Friendly Link, Visitor, and Content Administrant.

As with any database, determining business rules was an important part of our design process. Users have different requirements, which the database must meet. During the data modeling stage, we attempted to determine what these rules are, and to adjust the database relations accordingly. Understanding the business process and knowing how the business has decided to use the database information are critical.

By discussing with several times, reviewing the system, and studying existing information systems, we add more entities to the system to make the system works better. New entities includes: User, fans, Activity, Advertise, Advertise Manager, Application, Application Manager, AuthenUser_hold_Activity, Authenticated User, Authenticated User_has_FLink, Authenticated Type, Comment, DelMessage, Friendly Link, Hot Topic, Hot Topic Type, Hot topic_has_PhotoFolder, HotTopic_has_videoFolder, Location History, Manger, Message, Message Manager, Order manager, Order type, Orders, Photo Folder, Photos, Privileges, User Follow, VIP, Video Folder, Videos, Hot Topic, Hot Topic Type, Message.
We developed a list of business rules describing the policies of mini-Blog.
The general business rules are as follows:

1) Any registered user could pay to be a VIP member
2) VIP users have more privileges than normal users
3) A registered user could follow other registered users, including authentic users
4) Any user could post any messages, including photos and videos
5) Message Manager could delete messages posted by users
6) Advertise Manager could post and delete advertisement
7) Application Manager deals with applications
8) Order Manager deals with orders

KEY TABLES

1. User(U_ID, U_NickName, U_login, U_Email, U_Password, U_FirstName, U_LastName, U_Gender, U_RegisterTime, U_Mg_Count, U_Follow_Count, U_Fans_Count, U_Website, U_Phone, U_Info)
2. Message(Mg_ID, Mg_Content, Mg_commentNum, Mg_DateTime, U_ID, Mg_Image, Mg_Video, HotTopic_ID)
3. Photos(Photos_ID, PhotoFolder_ID, Photos_Description, Phtos_State, Photos_City, Mg_ID)
4. Videos(Videos_ID, VideosFolder_ID, Videos_Description, Videos_State, Videos_City,

Mg_ID)

5. Orders(Order_ID, Order_Date, Order_Type_ID, U_ID, Order_CardType, Order_CardNumber, Vip_ID, Manager_ID)
6. Manager(Manager_ID, Manager_FirstName, Manager_LastName, Manager_State, Manager_City, Manager_Street, Manager_login, Manger_Password, Manager_Email, Manager_Phone, Manager_Type
7. Advertise(Advertise_ID, Advertise_Type, Advertise_Info, Manager_ID, Advertise_Company, Advertise_Price)
8. Activity(Activity_ID, Activity_Title, Activity_State, Activity_City, Activity_Street, Activity_StartDate)

Relationships with Cardinalities

Relationships describe how entities are associated with each other. Following list show the important relationships with their cardinalities in mini-Blog system,

1) User and Message relationship one-to-many
2) User and PhotoFolder relationship one-to-many
3) User and VideoFolder relationship one-to-many
4) User and Comment relationship one-to-many
5) User and LocationHistory relationship one-to-many
6) User and Orders relationship one-to-many
7) User and application relationship one-to-one
8) User and user relationship many-to-many
9) Manager and DelMessage relationship one-to-many
10) AdversimentManager and Advertise relationship one-to-many
11) Application Manager and Application relationship one-to-many
12) OrderManager and Orders relationship one-to-many
13) AuthenticatedUser and FriendlyLink relationship
    many-to-many (via AuthenticatedUser_has_FLink)
14) AuthenticatedUser and Activity relationship
    Many-to-many (via AuthenUser_hold_Activity)
15) One application include one authenticated _type
16) One VIP has one privilege
17) One order has one order type
18) One PhotoFolder includes zero or many photos
19) One VideoFolder include zero or many videos
20) One message includes zero or more comments
21) One message includes zero or more photos
22) One message includes zero or more videos

NORMALIZATION

1)

Photo (Before normalization)

Photo_ID	Photo_Description	Photo_State	Photo_City	Mg_Id	PhotoFolder_Title	PhotoFolder_Infor

PhotoFolder

PhotoFolder_ID	PhotoFolder_Title	PhotoFolder_Info

Photo

Photo_ID	Photo_Description	Photo_State	Photo_City	PhotoFolder_ID	Mg_ID

Photo Folder Info and Photo Folder Number are dependent on Photo Folder Title. So we created a new table called Photo Folder, anything related to photo folder could be stored in this table.

2)

Message（Before normalization）

Mg_ID	Mg_Content	Photo_ID	Photo_Description	Video_ID	Video_Description	U_ID

Message

Mg_ID	Mg_Content	Photo_ID	Video_ID	U_ID

Photo

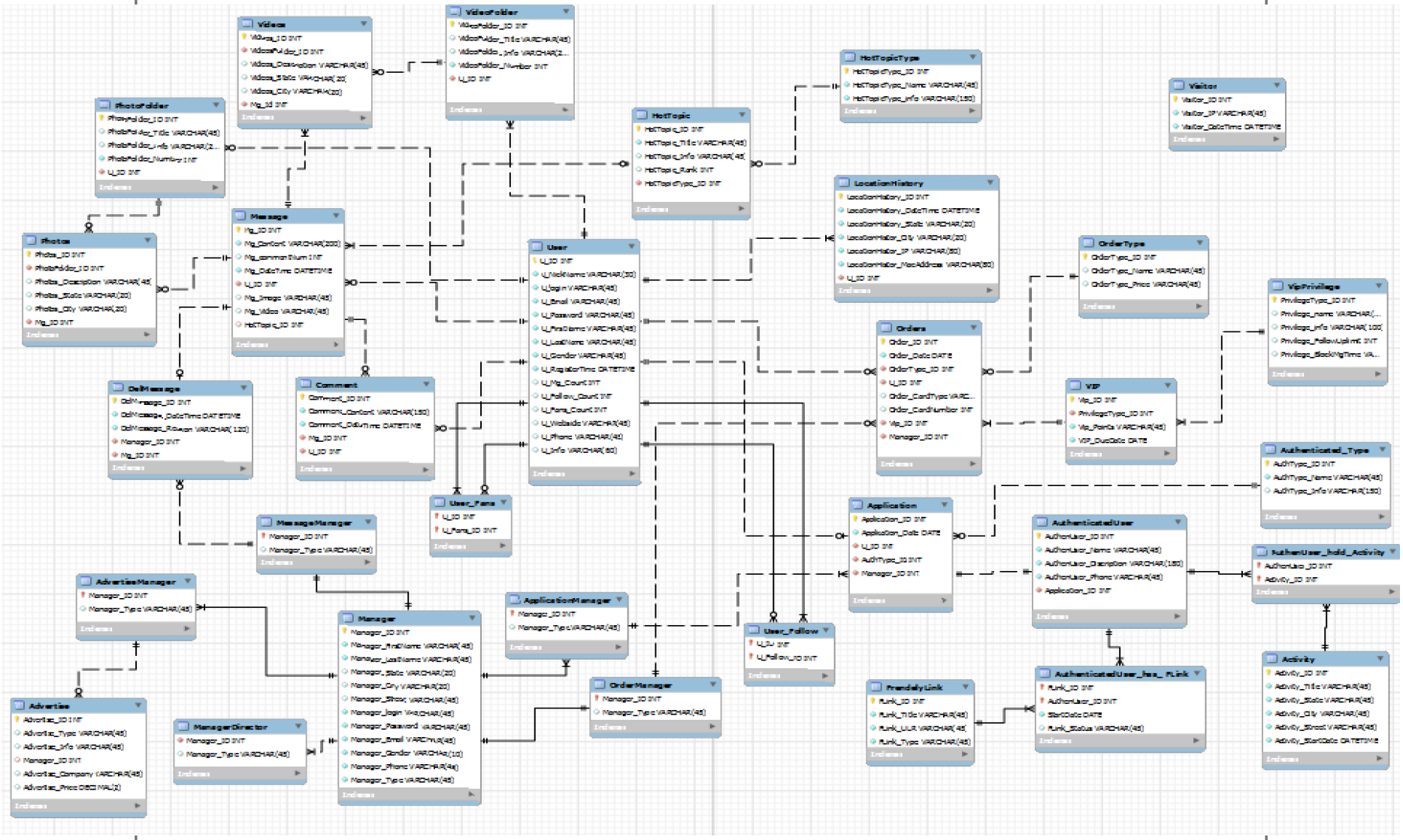
Photo_ID	Photo_Description	Photo_State	Photo_City	PhotoFolder_ID	Mg_ID

Video

Video_ID	Video_Description	VideoFolder_ID	Video_State	Video_City	Mg_ID

Message includes photo description and video description before normalization. Both attributes are dependent on key attribute via another non-key attributes. After normalization, Message table is split into three tables which are message, photo, and video. We also add more attributes to table photo and video.

EER DIAGRAM


Video
VideoFolder
PhotoFolder
HotTopicType
HotTopic
Visitor
LocationHistory
Message
User
OrderType
Photos
Orders
VipPrivilege
DelMessage
Comment
VIP
User_Fans
Authenticated_Type
MessageManager
Application
AuthenticatedUser
AuthenUser_hold_Activity
AdvertiseManager
ApplicationManager
Manager
User_Follow
Activity
Advertise
OrderManager
ManagerDirector
FriendlyLink
AuthenticatedUser_has_FLink

SECURITY ISSUE

To ensure security of this database, it should satisfied the following required:

Physical integrity databases;

Logical integrity of databases;

User identification;

Apart from those database can have following security issues:

1) User who wants to upgrade to VIP user should be charged for the order after user has been upgrade. In that case he also should not be charged for the same order twice.

2) Data association problem arises. The list containing the Authenticateduser name and a list containing the Authenticateduser phone are unclassified, but a combiled list containing the names and phone of Authenticateduser is considered classified.

3) Some users might try to create multiple numbers of orders to create concurrency problem in database, making data out of sync in case of order and their payments.

4) The order is made secure by assigning unique usernames to all users. And password is invisible to any of the system users and might be visible to some of managers.

PRIVILEGE

1) Registered User

Select on fans and follows tables

Select on comment table

Select on hot topics table

Select one location history table

2) VIP

Possess all privilege a registered user have

3) Authentic User

Hold all privilege a registered user have

Select, insert, update on authenticateuser_has_flink table

Select, insert, update on activity table

Select, insert, update on frendelylink table

4) Application Manager

Select on user_fans, user_follow tables

Select on comment, message, hottopic, locationhistory tables

Select on Activity, frendelylink, vipprivilege tables

Select on videos, videofolder, photos, photofolder tables
Select, insert, delete, update on Application table
Select, insert,delete, update on AuthenticatedUser table
Select, insert, delete, update on Authenticated_Type table

5) Advertisement Manager
Select on user_fans, user_follow table
Select on comment, message, hottopic, hottopictype tables
Select on locationhistory table
Select on Activity table
Select on frendelylink table
Select on videos, videofolder, photos, photofolder tables
Select on vipprivilege, authenticated_type table
Select on advertise table

6) Message Manager
Select, delete, update, insert on message table
Select on advertise table
Select on activity table
Select on application table
Select on comments, hottopic, hottopictype tables
Select on photos, photofolder, videos, videofolder table

7) Order manager
Select, delete, update, insert on order table
Select on advertise table
Select on activity table
Select on application table
Select on comments table
Select on hottopic, hottopictype tables
Select on photos, photofolder, videos, videofolder tables

VIEW

1) For message manager

(all delete message with user name and message content)

```
CREATE view v_UserDelMessag
AS
SELECT DelMessage.DelMessage_ID, Message.Mg_Content, User.U_ID,
User.U_NickName
from
(User inner join Message
on User.U_ID= Message.U_ID)
inner join DelMessage
on Message.Mg_ID= DelMessage.Mg_ID;

CREATE view v_UserDelMessage
AS
SELECT DelMessage.DelMessage_ID, Message.Mg_Content, User.U_ID,
User.U_NickName
from
(User inner join Message
on User.U_ID= Message.U_ID)
inner join DelMessage
on Message.Mg_ID= DelMessage.Mg_ID;

SELECT * FROM v_userdelmessage;
```

2) Hot topic for user

```
CREATE view v_HottopComment
AS
SELECT Hottopic.HotTopic_Title, Hottopic.HotTopic_Info, Message.Mg_Content,
comment.Comment_Content
from
(HotTopic inner join Message
on HotTopic.HotTopic_ID= Message.HotTopic_ID)
inner join comment
on Message.Mg_ID= comment.Mg_ID;

select * from v_HottopComment;
```

3) Photo with message

```
CREATE view v_PhotoMessage
AS
SELECT Message.Mg_Content, Photos.Photos_Description
from
Message inner join Photos
on Message.Mg_ID= Photos.Mg_ID;

select * from v_photoMessage;
```

4) Video with message

```
CREATE view v_VideoMessage
AS
SELECT Message.Mg_Content, Videos.Videos_Description
from
Message inner join Videos
on Message.Mg_ID= Videos.Mg_ID;

select * from v_videoMessage;
```

5) Application Manager

```
CREATE view v_AuthenUser -- applicationmanager
AS
SELECT    User.U_NickName, Authenticated_Type.AuthType_Name,
Application.Application_Date
from
(User inner join Application
on User.U_ID= Application.U_ID )
inner join Authenticated_Type
on Application.AuthType_ID= Authenticated_Type.AuthType_ID;

select * from v_authenuser;
```

6) All comments of each user

```
create view v_CommentsOfUser
as
select a.U_ID, b.Mg_ID,b.Comment_Content,b.Comment_DateTime
FROM user as a left join Comment as b
on a.U_ID = b.U_ID
order by a.u_id;

select * from v_commentsofuser;
```

7) Activity

```
CREATE VIEW v_activity AS
SELECT Activity_Title, Activity_City, Activity_StartDate
FROM activity
GROUP BY Activity_StartDate;

SELECT * FROM V_ACTIVITY;
```

8) Advertisement Manager

```
CREATE VIEW v_advertisemanager AS
SELECT Advertise_Info, Advertise_Company, Advertise_Price
FROM advertise
GROUP BY Advertise_Company;

SELECT * from V_advertisemanager;
```

STORE PROCEDURE

1) search fans procudure

```
delimiter //
create procedure sp_searchFans
(In userid int)
begin
    select U_NickName,U_Gender,U_Email, U_Webside, concat(u_firstname, ' ', u_lastname)
    as U_FullName from user
    where u_id in (select U_Fans_ID from user_fans where u_id = userid);
    end //
    delimiter ;

call sp_searchFans(1);
```

2) search following procedufe

```
delimiter //
create procedure sp_searchfollowing(
In userid int
)
begin
    select U_NickName,U_Gender,U_Email, U_Webside, concat(u_firstname, ' ', u_lastname)
    as U_FullName from user
    where u_id in (select U_Follow_ID from user_follow where u_id = userid);
    end//
    delimiter ;
call sp_searchfollowing(1);
```

3) User can use to search all of the messages he or she have posed

```
delimiter //
create procedure sp_searchOwnMessage(
In userid int
)
begin
    select Mg_ID,Mg_Content,Mg_DateTime, Mg_Video, Mg_Image,Mg_commentNum
    from message
    where u_id = userid;
    end//
    delimiter ;
    call sp_searchOwnMessage(1);
```

4) User can search all of the comments of his a certain message

```
delimiter //
create procedure sp_searchComments(
In userid int,
In MessageId int
)
begin
    select a.Mg_ID, Mg_Content,Comment_ID, Comment_Content, Comment_DateTime
    from Message as a Left join Comment as b on a.Mg_ID = b.Mg_ID
    where a.Mg_id = MessageId;
    end//
    delimiter ;

call sp_searchComments(1, 9000045);
```

5) Find all of the comments a user has made

```
delimiter //
create procedure sp_usercomments(
In Userid int
)
begin
    select b.u_id, c.U_NickName, a.Mg_Content, b.Comment_Content
    from message as a left join v_commentsofuser as b
    on a.Mg_ID = b.Mg_ID
    left join User as c
    on a.u_ID = c.U_ID
    where b.U_ID = Userid
    order by c.U_NickName;
    end//
    delimiter ;
call sp_usercomments(1);
```

6) Order search

```
delimiter //
CREATE procedure sp_UserOrder
(in UserID int)
 begin
   select User.U_ID, User.U_NickName,    Orders.OrderType_ID, Orders.Order_Date
   from User inner join Orders
   on User.U_ID = Orders.U_ID
   where User.U_ID= UserID;
   end //
   delimiter ;

   call sp_userorder(2);
```

TRIGGER

1) Add CommentNumber

```
delimiter //
create trigger ComIncrease after insert on comment
for each row
begin
update Message
set Mg_commentNum = Mg_commentNum + 1
where Mg_ID= new.Mg_ID;
end //
delimiter ;
```

2) Delete relational information when delete a message

```
    delimiter //
CREATE TRIGGER message_Photos
after delete ON message
FOR EACH ROW
BEGIN
delete  from  photos
where message.Mg_ID = photos.Mg_ID;
END;
delimiter ;
```

```
delimiter //
CREATE TRIGGER message_comments
after delete ON message
FOR EACH ROW
BEGIN
delete from comments
where message.Mg_ID = comments.Mg_ID;
END;
delimiter ;

delimiter //
CREATE TRIGGER message_delmessage
before delete ON message
FOR EACH ROW
BEGIN
insert into delmessage(Mg_ID)
values (message.Mg_ID);
END;
delimiter ;

delimiter //
CREATE TRIGGER videofolder_video
after delete ON videofolder
FOR EACH ROW
BEGIN
delete from videos
where video.VideoFolder_ID = videofolder.VideoFolder_ID;
END;
delimiter ;
```

TRANSACTION

```
Start transaction;
insert into Activity
Values(100040, 'happy','MA','Boston','Place road','20150915'),(100041,'unhappy','NY','New
York','Huntting Street','20150917');
COMMIT;
SELECT * FROM Activity;
```

```
START TRANSACTION;
UPDATE USER SET    U_NickName='CANDY'
WHERE U_ID=1;
SELECT * FROM USER ORDER BY U_ID;
ROLLBACK;
SELECT * FROM USER ORDER BY U_ID;
```

```
START TRANSACTION;
UPDATE HOTTOPIC SET Hottopic_info=' THIS TYPE IS MOST POPULAR'
WHERE HottopicType_ID=002000;
SAVEPOINT SAVAPOINT1;
update hottopic set hottopic_info='THIS TYPE IS NOT POPULAR'
WHERE hottopicType_ID=002001;
select * from hottopic where hottopicType_ID=002000 or hottopicType_ID=002001
order by HottopicType_ID;
rollback to SAVEPOINT1;
COMMIT;
SELECT * from hottopic where hottopicType_ID=002000 or hottopicType_ID=002001
order by HottopicType_ID;
```

```
START TRANSACTION;
savepoint savepoint1;
update Activity set    Activity_City='Boston' where Activity_ID =100001;
select * from Activity where Activity_ID=100001;
rollback to savepoint1;
commit;
select * from Activity where Activity_ID=100001;
```