

数据库系统原理课程设计报告

1 背景概述

随着手机应用的发展，社交软件不断的更新，微信朋友圈成为重要的社交场所，是方便网络用户实时更新自己的动态，同时展示丰富生活的平台。本次课程设计，基于积累的数据库知识，实现微信朋友圈这一场景，复现用户发布自己的动态信息，也可以点赞或评论他人的动态信息等功能，将理论知识付诸实践。

2 系统实现方案设计

2.1 需求分析

硬件环境：64 位计算机

软件环境：1.Windows 10 2. Microsoft SQL Server 2017

用户登陆后，可以查看自己和好友的朋友圈，发布新的动态信息，可以对他人的信息进行点赞评论。

2.2 系统目标

数据库支持实现以下行为：

用户查看、编辑或删除自己已发布的动态信息

用户发布新的动态信息

用户查看他人的动态信息

用户点赞或取消点赞他人的动态信息

用户评论、编辑或删除评论他人的动态信息

数据库权限支持：

用户仅能查看好友的动态信息

2.3 系统所包含的信息

用户信息（微信号，用户名，头像，性别，家乡，密码）

朋友圈记录（发布用户微信号，发布时间，图片，文字，记录编号）

点赞信息（朋友圈记录编号，点赞用户微信号，点赞时间）

评论信息（朋友圈记录编号，评论用户微信号，评论时间，评论内容）

好友关系信息（用户 1，用户 2）

2.4 完整性约束

实体完整性：用户信息中微信号为主键，朋友圈记录中（发布用户微信号，发布时间）为主键，点赞信息中（点赞用户微信号，点赞时间）为主键，评论信息中（评论用户微信号，评论时间为主键）

域完整性：微信号、用户名、家乡，朋友圈内容、评论内容均为字符串；头像、朋友圈图片均为图片编号；性别为男或女；发布时间、点赞时间、评论时间均为日期。

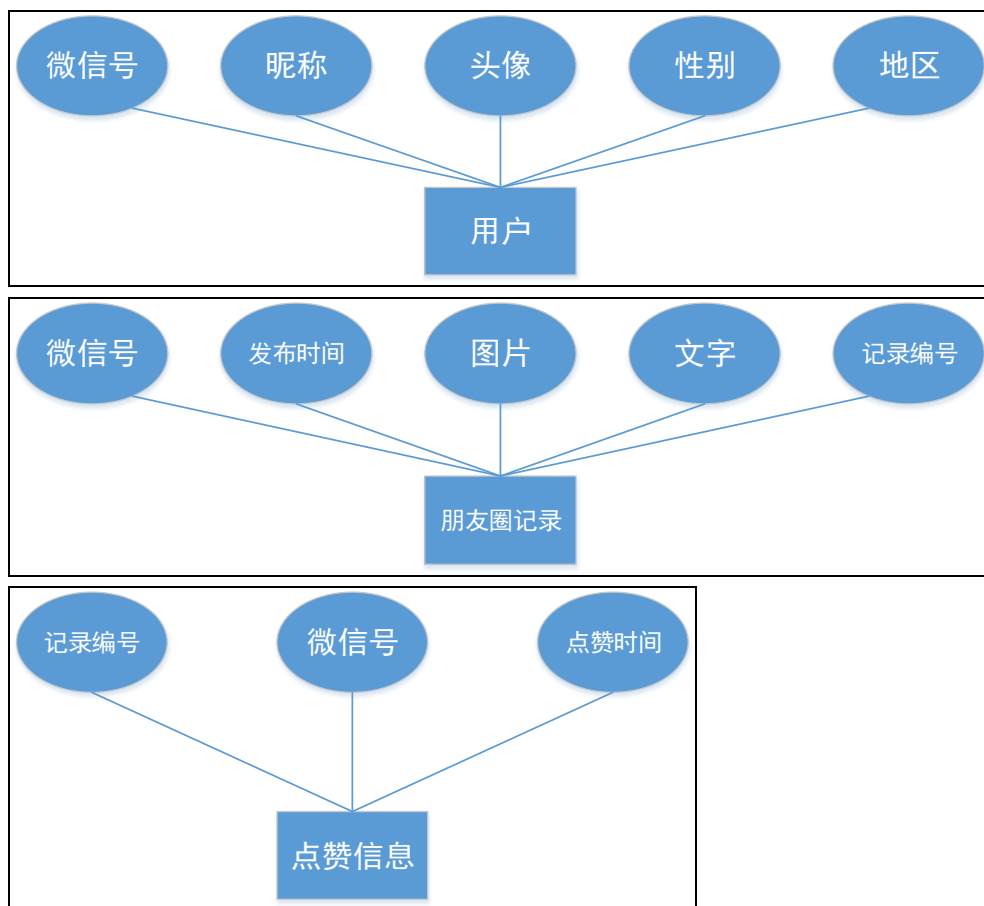
参照完整性：朋友圈记录中发布用户微信号为外键，用户信息为被参照关系。点赞信息中点赞用户微信号为外键，用户信息为被参照关系。评论信息中评论用户微信号为外键，用户信息为被参照关系。

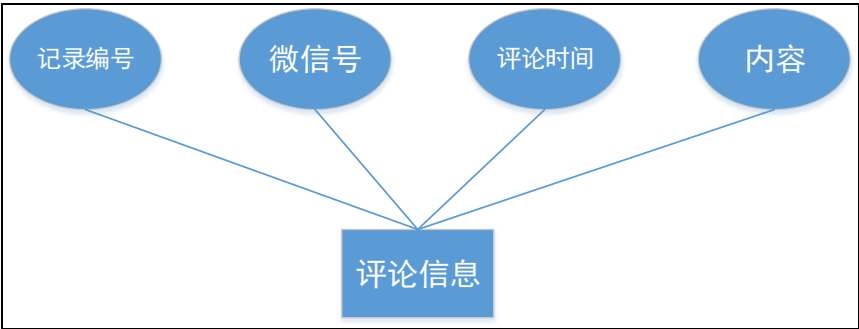
2.5 与系统的交互

不同用户需要各自建立用户名密码及好友关系，通过 Python 建立用户交互界面，支持朋友圈基础操作和部分扩展操作。

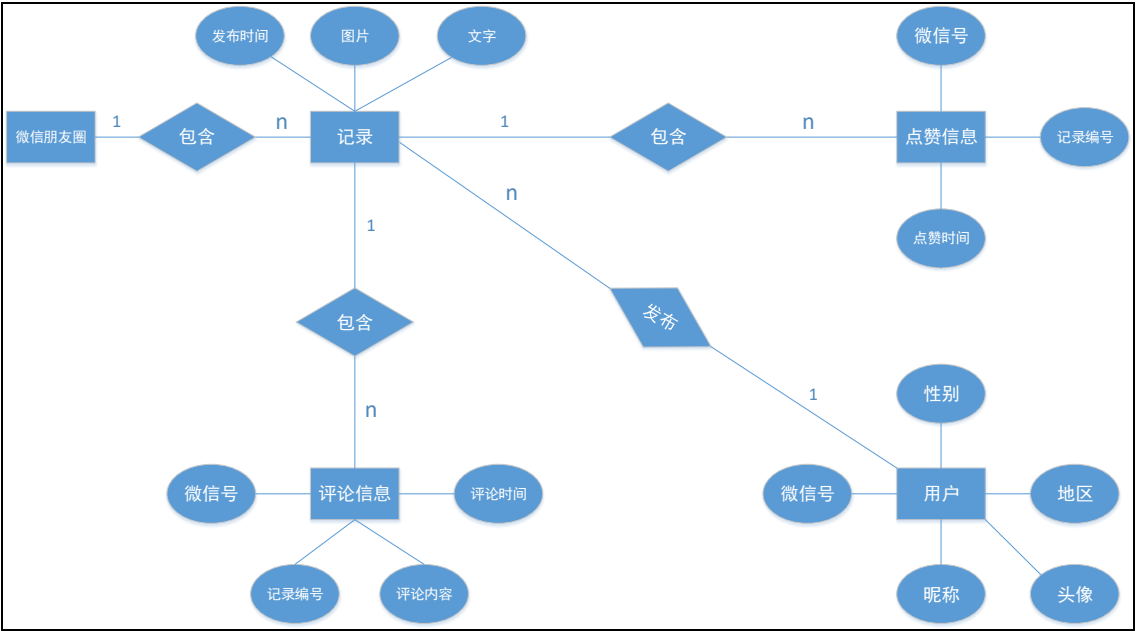
2.6 数据库逻辑结构设计

2.6.1 局部 ER 图：





2.6.2 全局 ER 图



3 系统实现方案

3.1 视图

USERINFO 表：

	userID	uName	uGender	uPassword
	123	qqq	男	123
	321	www	男	321
	456	yyy	女	456
	wxh	wzh	男	123
▶*	NULL	NULL	NULL	NULL

PYQLIST 表:

	pCreator	pCreTime	pContent	pNumber
	123	2018-12-20 1...	666	1
	123	2018-12-20 2...	赞和取消赞成...	3
	123	2018-12-21 1...	www	4
	456	2018-12-20 1...	好友朋友圈成功	2
▶▶	NULL	NULL	NULL	NULL

LIKEINFO 表:

	lNumber	lCreator	lCreTime
	1	123	2018-12-20 22:00:39.000
	2	123	2018-12-21 12:58:06.000
	1	321	2018-12-20 22:00:22.000
	3	456	2018-12-20 20:59:20.000
	2	456	2018-12-20 20:59:21.000
▶▶	NULL	NULL	NULL

COMMENT 表:

	cNumber	cCreator	cCreTime	cContent
	1	123	2018-12-21 10:58:01.000	hhh
	3	456	2018-12-21 11:06:25.000	233
▶▶	NULL	NULL	NULL	NULL

FRIENDSHIP 表:

	fU1	fU2
	123	123
	123	321
	123	456
	123	wxh
	321	123
	321	321
	456	123
	456	456
	456	wxh
	wxh	123
	wxh	456
	wxh	wxh
▶▶	NULL	NULL

3.2 存储过程

(1) 注册用户 (CREATEUSER_PROCEDURE)

```
CREATE PROCEDURE CREATEUSER_PROCEDURE
    @id varchar(10),
    @name varchar(10),
    @gender varchar(2),
    @password varchar(10)
AS
BEGIN
    INSERT INTO USERINFO(userID, uName, uGender, uPassword) VALUES (@id, @name,
    @gender, @password);
    INSERT INTO FRIENDSHIP(fU1, fU2) VALUES (@id, @id);
END
```

(2) 创建新朋友圈 (ADDPYQ_PROCEDURE)

```
CREATE PROCEDURE ADDPYQ_PROCEDURE
    @creatorID varchar(10),
    @time datetime,
    @content varchar(100)
AS
BEGIN
    INSERT INTO PYQLIST(pCreator, pCreTime, pContent) VALUES (@creatorID,
    @time, @content)
END
```

(3) 添加新朋友 (ADDFRIEND_PROCEDURE)

```
CREATE PROCEDURE ADDFRIEND_PROCEDURE
    @id1 varchar(10),
    @id2 varchar(10)
AS
BEGIN
    INSERT INTO FRIENDSHIP(fU1, fU2) VALUES (@id1, @id2);
    INSERT INTO FRIENDSHIP(fU1, fU2) VALUES (@id2, @id1);
END
```

(4) 添加新评论 (ADDCOMMENT_PROCEDURE)

```
CREATE PROCEDURE ADDCOMMENT_PROCEDURE
    @index int,
    @account varchar(10),
    @time datetime,
    @content varchar(50)
AS
BEGIN
    SET IDENTITY_INSERT COMMENT ON;
    INSERT INTO COMMENT(cNumber, cCreator, cCreTime, cContent) VALUES (@index,
@account, @time, @content);
    SET IDENTITY_INSERT COMMENT OFF;
END
```

3.3 触发器

在删除朋友圈之后，自动启用触发器删除有关点赞信息和评论

```
CREATE TRIGGER DELETERELATE
ON PYQLIST
AFTER DELETE
AS
BEGIN
    declare @del_id int
    select @del_id = pNumber from deleted
    delete from LIKEINFO where lNumber = @del_id
    delete from COMMENT where cNumber = @del_id
END
```

3.4 规范化要求

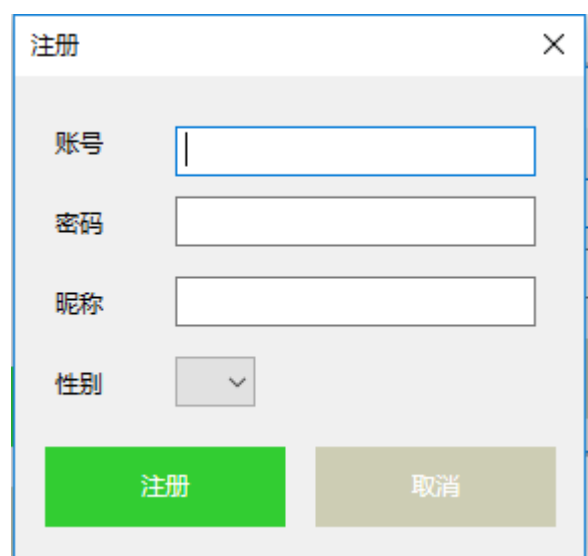
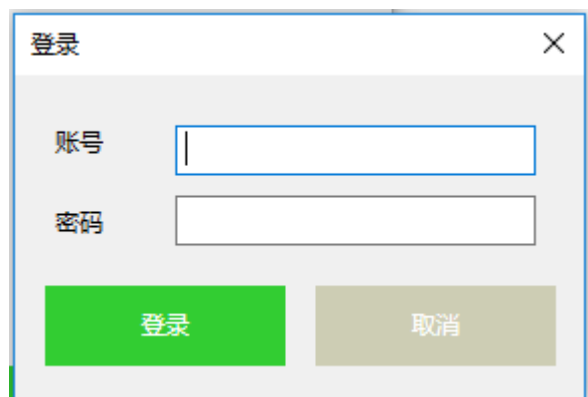
此设计满足 4NF

3.5 索引

已为所有表自动创建主键索引

4 系统功能实现

1 用户登录与注册



2 用户查看自己和好友的朋友圈



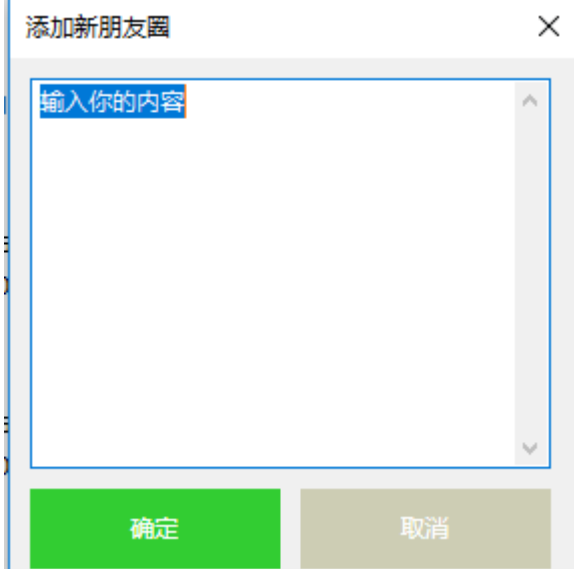
3 添加好友 添加新朋友圈



添加朋友

输入ID

确定 取消

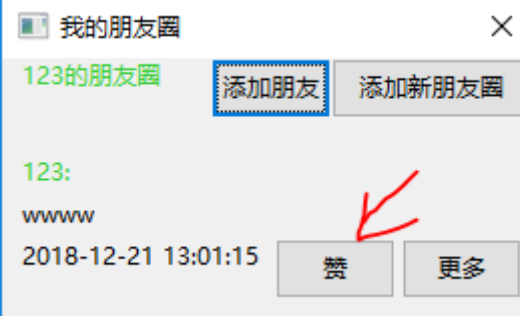


添加新朋友圈

输入你的内容

确定 取消

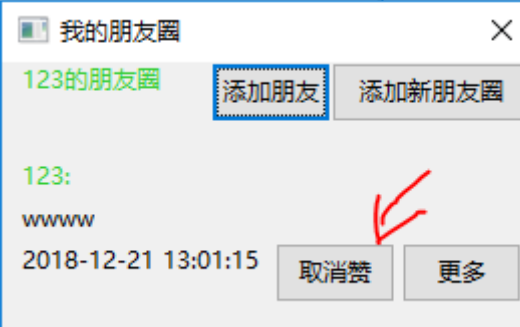
4 点赞和评论



我的朋友圈

123的朋友圈 添加朋友 添加新朋友圈

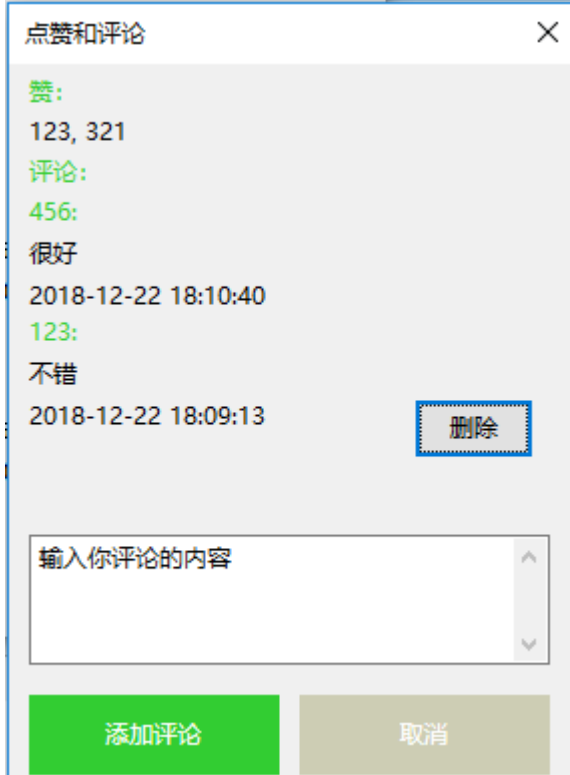
123:
www
2018-12-21 13:01:15 赞 更多



我的朋友圈

123的朋友圈 添加朋友 添加新朋友圈

123:
www
2018-12-21 13:01:15 取消赞 更多



点赞和评论

赞:
123, 321
评论:
456:
很好
2018-12-22 18:10:40
123:
不错
2018-12-22 18:09:13 删除

输入你评论的内容

添加评论 取消

5 系统设计总结

朋友圈的基本功能都已经实现，可以正常的进行交互。在设计过程中，界面的设计是一个很大的难点，尽管耗费很多时间克服，但还是不算精致。在设计过程中，发现关系数据库还是有部分局限性，在特定的应用场景下，有数据冗余的风险。最终，回复评论的功能还没有实现，是一个不小的缺憾。通过此次实验，我对数据库的基本操作和设计流程加深了印象，为今后能很好参与到用数据库的工作打下了坚实的基础。