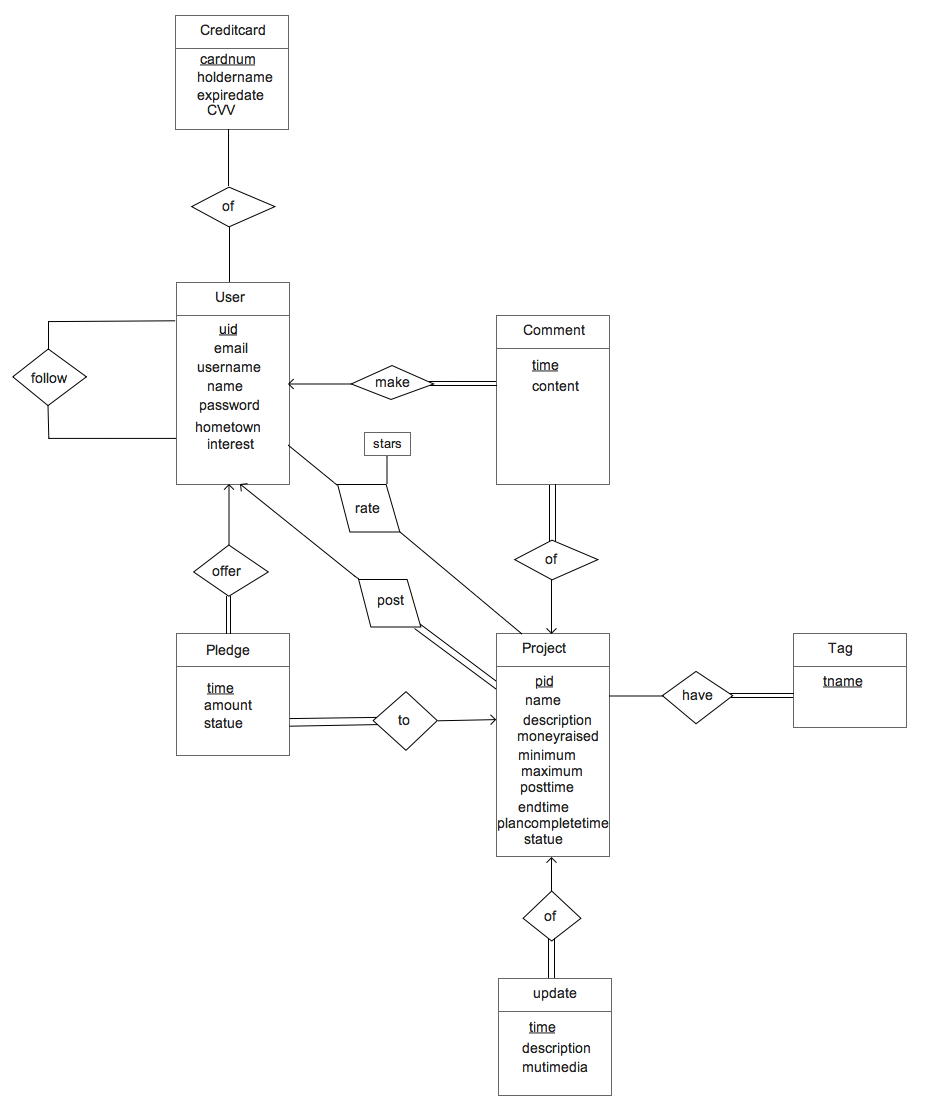
**ER diagram**



**Schema**

User(uid, email, username, name, password, hometown, interest)

Creditcard(cardnum, holdername, expiredate, CVV)

CardOwnership(uid, cardnum)

Project(pid, uid, name, description, moneyraised, minimum, maximum, posttime, endtime, completetime, statue)

Update(pid, time, description, multimedia)

Follow(uid, followerID)

Comment(uid, pid, time, content)

Rate(uid, pid, stars)

Tag(pid, tname)

Pledge(uid, pid, time, amount, statue)

uid in CardOwnership referencecs to uid in User;

uid in Project references to uid in User;

pid in Update references to pid in Project;

uid in Follows references to uid in User;

followerID in Follows references to uid in User;

uid in Comment references to uid in User;

pid in Comment references to pid in Project;

uid in Rate references to uid in User;

pid in Rate references to pid in Project;

pid in Tag reference to pid in Project;

uid in Pledge reference to uid in User;

pid in Pledge reference to pid in Project;

**DDL:**

CREATE TABLE `User` (

`uid` int NOT NULL AUTO\_INCREMENT,

`email` varchar(45) NOT NULL,

`username` varchar(20),

`name` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`hometown` varchar(45),

`interest` varchar(100),

PRIMARY KEY (`uid`));

CREATE TABLE `Creditcard` (

`cardnum` varchar(20) NOT NULL,

`holdername` varchar(40) NOT NULL,

`expiredate` DATE NOT NULL,

`CVV` int NOT NULL,

PRIMARY KEY (`cardnum`));

CREATE TABLE `CardOwnership` (

`uid` int NOT NULL,

`cardnum` varchar(20),

PRIMARY KEY (`uid`, `cardnum`),

FOREIGN KEY (`uid`) REFERENCES `User`(`uid`),

FOREIGN KEY (`cardnum`) REFERENCES `Creditcard`(`cardnum`));

CREATE TABLE `Project` (

`pid` int NOT NULL AUTO\_INCREMENT,

`uid` int NOT NULL,

`name` varchar(20) NOT NULL,

`description` varchar(3000),

`moneyraised` int DEFAULT 0,

`minimum` int NOT NULL,

`maximum` int NOT NULL,

`posttime` DATE NOT NULL,

`endtime` DATE NOT NULL,

`plancompletetime` DATE NOT NULL,

`statue` ENUM('ongoing', 'successed', 'failed') DEFAULT "ongoing",

PRIMARY KEY (`pid`),

FOREIGN KEY (`uid`) REFERENCES `User`(`uid`));

CREATE TABLE `Update` (

`pid` int NOT NULL,

`time` DATE NOT NULL,

`description` varchar(3000),

`multimedia` BLOB,

PRIMARY KEY (`pid`, `time`),

FOREIGN KEY (`pid`) REFERENCES `Project`(`pid`));

CREATE TABLE `Follow` (

`uid` int NOT NULL,

`followerID` int NOT NULL,

PRIMARY KEY (`uid`, `followerID`),

FOREIGN KEY (`uid`) REFERENCES `User`(`uid`),

FOREIGN KEY (`followerID`) REFERENCES `User`(`uid`));

CREATE TABLE `Comment` (

`uid` int NOT NULL,

`pid` int NOT NULL,

`time` DATE NOT NULL,

`content` varchar(3000),

PRIMARY KEY (`uid`, `pid`),

FOREIGN KEY (`uid`) REFERENCES `User`(`uid`),

FOREIGN KEY (`pid`) REFERENCES `Project`(`pid`));

CREATE TABLE `Rate` (

`uid` int NOT NULL,

`pid` int NOT NULL,

`stars` int NOT NULL,

PRIMARY KEY (`uid`, `pid`),

FOREIGN KEY (`uid`) REFERENCES `User`(`uid`),

FOREIGN KEY (`pid`) REFERENCES `Project`(`pid`));

CREATE TABLE `Tag` (

`pid` int NOT NULL,

`tname` varchar(20),

PRIMARY KEY (`pid`, `tname`),

FOREIGN KEY (`pid`) REFERENCES `Project`(`pid`));

CREATE TABLE `Pledge` (

`uid` int NOT NULL,

`pid` int NOT NULL,

`time` DATE NOT NULL,

`amount` numeric(10,2) NOT NULL,

`staute` ENUM('pending', 'released', 'refund') DEFAULT 'pending',

PRIMARY KEY (`uid`, `pid`),

FOREIGN KEY (`uid`) REFERENCES `User`(`uid`),

FOREIGN KEY (`pid`) REFERENCES `Project`(`pid`));

**Trigger:**

CREATE TRIGGER update\_moneyraised

AFTER INSERT

ON Pledge FOR EACH ROW

BEGIN

UPDATE Project

SET moneyraised = moneyraised + NEW.amount

WHERE pid = NEW.pid

END;

**Event:**

On Project, once moneyraised > maximum, update staute into successed.

On Project, once endtime > DATE(NOW()) and moneyraised < minimum, update statue into “failed”.

On Project, once endtime > DATE(NOW()) and moneyraised > minimum, update statue into “successed”.

DELIMITER //

CREATE PROCEDURE update\_statue()

BEGIN

update Project set status = 'successed' where moneyraised > maximum;

update Project set status = 'failed' where endtime > DATE(NOW()) and moneyraised < minimum;

update Project set status = 'successed' where endtime > DATE(NOW()) and moneyraised > minimum;

END; //

DELIMITER ;

DROP EVENT IF EXISTS modify\_project;

CREATE EVENT update\_project

on schedule EVERY 1 DAY

do CALL update\_statue();

**DML:**

- Create a record for a new user account, with a name, a login name, and a password.

//Login with email before username has been set, so email here is equivalent //to login name

INSERT INTO User(name, email, password)

- List all projects that contain the keyword ‘‘jazz’’ and that are currently looking for funds, sorted in descending order by posting time.

- List all users who have given money for projects containing the tag or category ‘‘jazz’’ in the past, sorted by the total amount they have successfully pledged (meaning, money that was actually charged).

- List all users who have completed at least 3 projects, and where each of their projects received an average rating of 4 stars or higher from its sponsors.

- List all comments by users that are followed by user ‘‘BobInBrooklyn’’. .

- Insert a new project for a particular user, with a name, description, and other needed info.

- Insert a pledge to sponsor a project, for a particular user, project, and amount.

- Write queries for the end of a funding campaign. E.g., you could use triggers to detect when a campaign is fully funded or time is up; if successfully funded, generate charges to sponsors’ credit cards.