Time-Machine 时间管理系统□

郑子威 （1300012711）

(北京大学 信息科学技术学院, 北京 100871);

E-mail: micheal_zzw@pku.edu.cn

摘要： 本课程设计通过利用数据库技术，设计并实现完成了一款基于数据库技术、Web 技术与 Laravel 框架的在线时间管理系统，该系统分为五大模块，验证模块、事件模块、朋友模块、邀请模块与统计模块。系统能够有效的对个人时间进行管理，同时通过多样的功能丰富了系统的实用性与交互性。功能的多样性使得各种数据库技术能得到针对性的利用与实践。

1 引言

现如今，我们处于一个信息爆炸的时代，一个“大数据”的时代，在现在，丰富的信息渠道大大缩减办事所需要的准备时间，使得人们的各方面的效率越来越高。在这个时代，效率已经不是制约个人的瓶颈，取而代之的是对个人时间的管理。如何在有效的时间完成尽可能多的事是现代社会学的重要课题。出于这样一种考虑，我希望能够设计一款对人们已经完成的事情进行管理的系统，不同于一般的管理系统，我设计的系统并不是一种备忘录，他的特点在于关注的是人们已经完成与正在进行的事件，通过事件类型的划定和具体的时间统计数据，来促使人们合理安排自己的时间计划。这就是我设计的在线时间管理系统——Time-Machine（时光机），一款让人们回顾过去，三省吾身的系统。他的主要功能模块有：

（1） 验证模块：用户的注册、登录、注销与密码找回。

（2） 事件模块：对于事件类型的管理，已完成，进行中的事件的创建、管理。

（3） 朋友模块：包括朋友的查找、请求与添加。

（4） 邀请模块：对于进行中的事件，能够邀请朋友共同参加。

（5） 统计模块：对近期（一周）、根据日期、总计三种统计方式，统计人们每一种事件类型所耗费的时间。

在设计该系统的过程中，由于功能的多样性，我充分合理的运用到各种数据库技术，包括数据库关系表的设计，数据库的建立，数据库综合考虑时间与空间的高效利用等方方面面，不仅是对数据库课程的一次有效总结与实践，同时也通过数据库与 Web 技术相结合的实践，收获了数据库应用设计的经验。

本文的组织：在第二部分中，我们介绍了本课程设计需要的背景知识，包括开发框架，数据库语言与众多数据库功能与技术；第三部分描述了课程设计的系统框架；第四部分给出相关的关系数据库模式和相应的操作；第五部分总结了全文，并且描述了完成本课程设计的收获。

2 相关工作

MVC 框架：MVC 是当前最为流行的应用开发框架之一，M 为 Model，包含了所有对数据的访问逻辑与操作，V 为 View，即所有的界面与视图，完成与用户的交互，C 为 Control，包含了所有的业务逻辑，同时用于操作 Model 与调用 View。

Laravel 开发框架：Laravel 是一个基于 php 语言的 Web 开发框架，在这个框架中，提供了大量简洁、清晰的语法与接口。该框架基于 MVC 框架，并且完美支持大量数据库操作与 php 语言的结合。

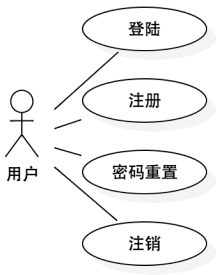
MySql：MySql 是 MySQL 公司开发的一款开源关系数据库管理系统，它支持最常用的数据库标准化语言，体积小、速度快，不过不提供部分 SQL 高级功能，如临时视图等。

数据库功能与技术：使用了绝大部分高军老师《数据库概论（实验班）》所介绍的大部分技术。

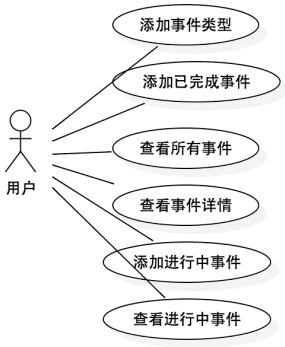
（1） 所有的基本 SQL 语言，增删改查、聚集函数。
（2） 复杂 SQL、嵌入式 SQL，用于支持复杂各种功能。
（3） 视图，用于提高安全性与存取速度。
（4） 事务，利用原子性，用于处理多张表同时处理的复杂功能。
（5） 索引、外键、完整性约束、级联删除等保证数据质量。
（6） 使用存储过程、递归 SQL 与复杂分组(groud by)、排序等，处理数据统计。
（7） 利用框架自带功能，防止 SQL 注入攻击。
（8） 使用触发器，用于处理多张表同时处理的复杂功能。

3 课程设计的系统框架

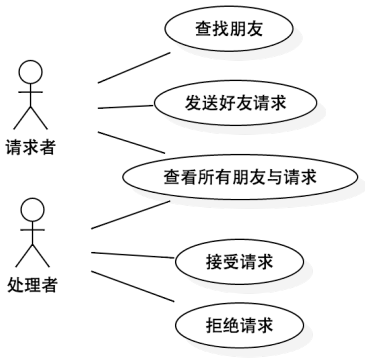
用况图：



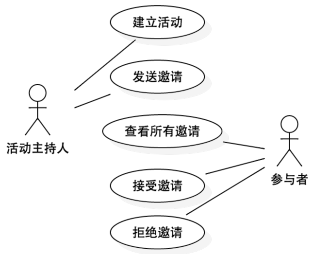
(a)验证模块



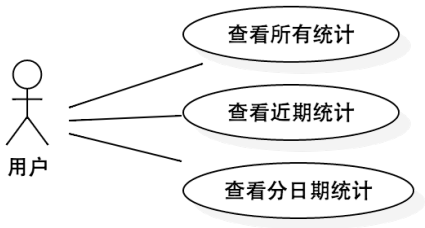
(b)事件模块



(c)朋友模块

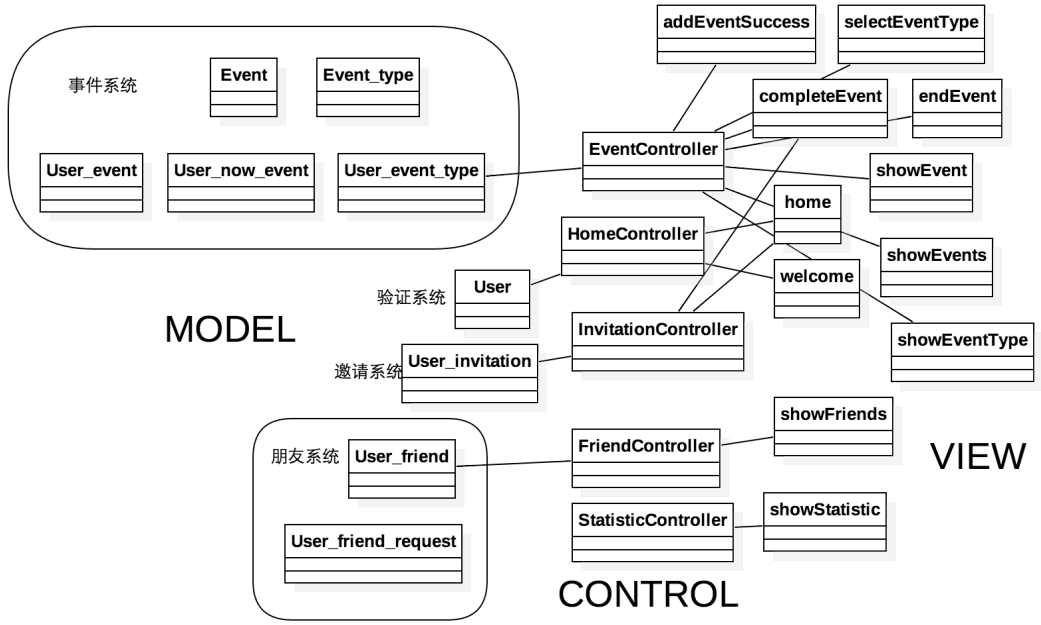


(d)邀请模块

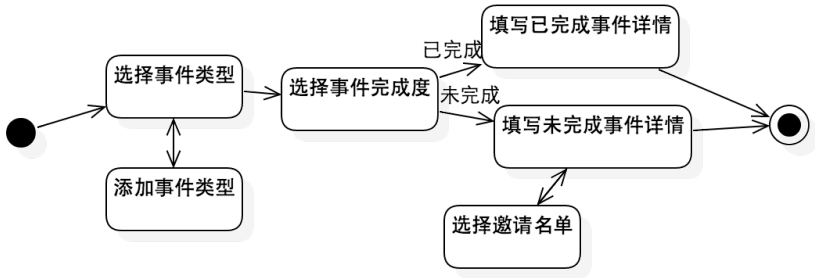


(e)统计模块

类图（MVC 模式）：

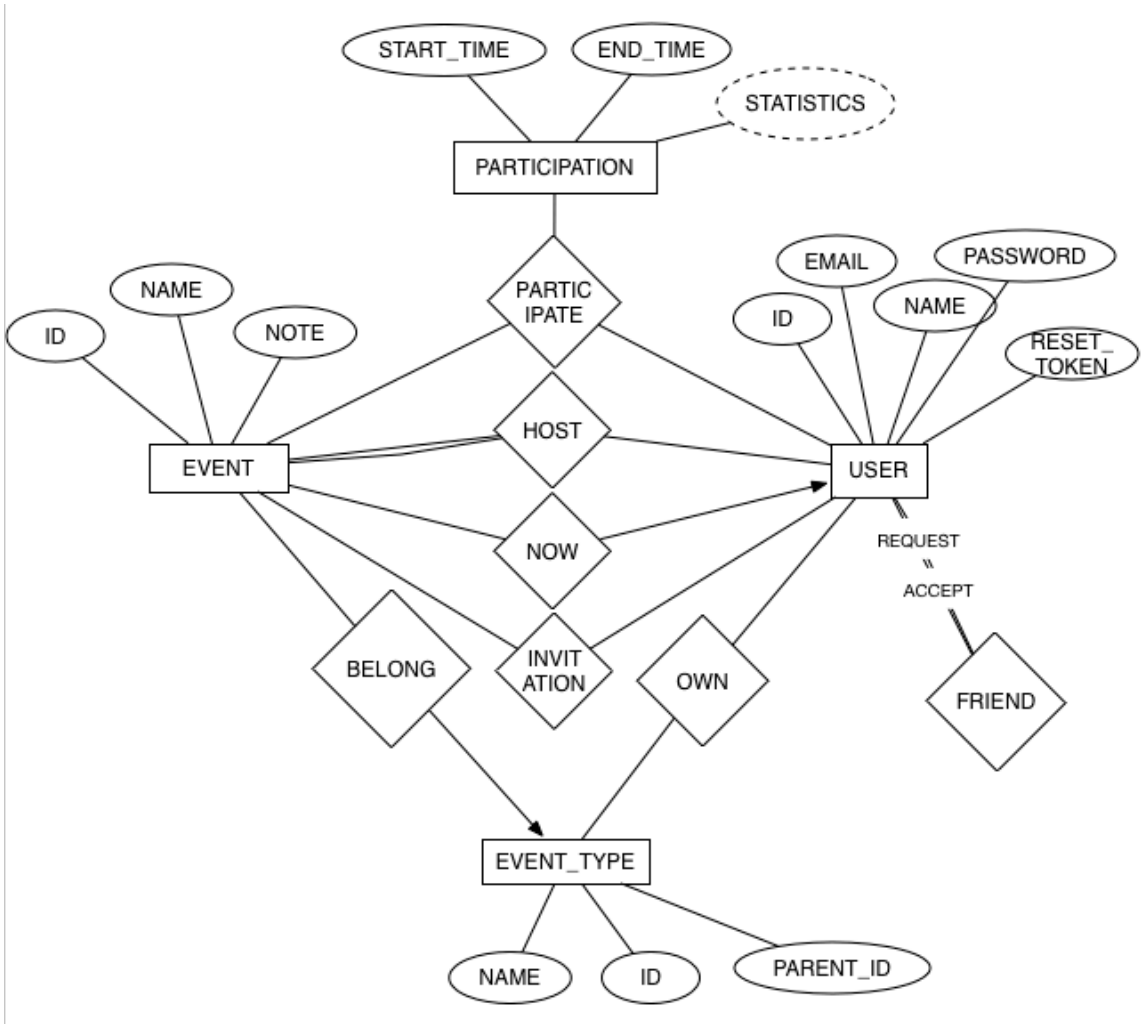


上面详细展示了每个模块的用况与类图，而整个系统最重要的流程与用户交互在于新建事件模块，因此下面给出新建事件的活动图。



4 课程设计对应的关系数据库模式

4.1 ER图



如上图所示，一共存在四个实体，User 即为用户，Event 即为一次事件，Event_type 为事件类型，Particapition 为参与纪录。可以看到：

用户与事件类型的关系是多对多，一个用户拥有一个事件类型表（own）。

用户与事件也是多对多的关系，多个用户可以被邀请参与同一个事件（invitation），一个用户也可以参与多个事件（participate），一个事件有且只有一个发起人（host），用户可能存在一个正在发生事件（now）。

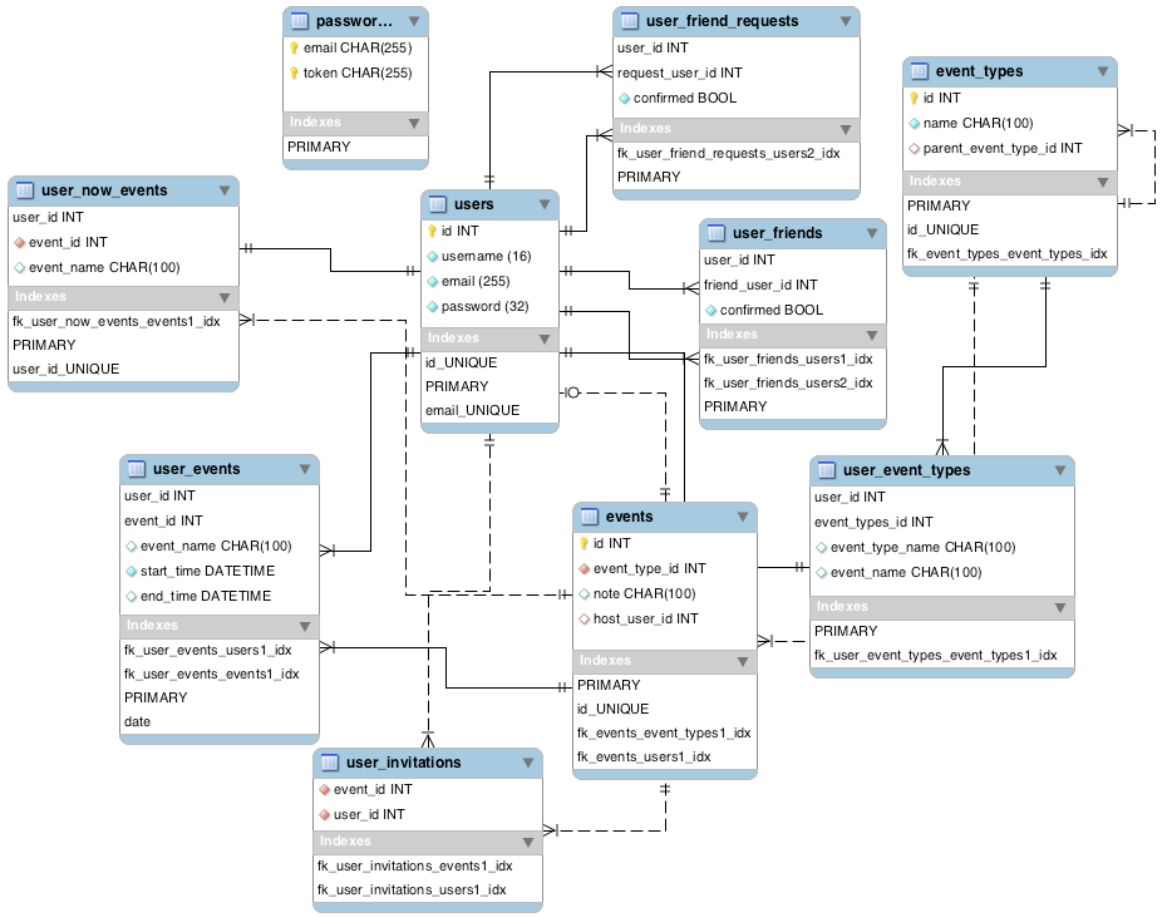
一次参与纪录对应一个用户和一次事件，记录了发生的时间。

一个事件有且只有一个事件类型（belong）。

用户与用户之间存在好友关系（friend），有一个请求与接受的关系。

事件记录存在统计数据。

4.2 关系数据库模式描述



该模型图由 SQL workbench 生成，生成文件参见附件 EER.mwb，总共有 10 张图，其中：

users 是用户验证表，events 事件表，event_types 事件类型表，user_events 事件记录表，分别对应 ER 图中四个实体。

user_now_events 对应 Now 关系，user_invitations 对应 Invitation 关系。

Belong、Host 作为 Event 的属性。

user_event_types 对应 Own 关系。

user_friends，user_friend_requests 对应 Friends 关系的两条边。

范式分析：

表名	含义	范式	理由
users	用户验证表	2NF	Id 与 email 都可以推出其余属性。
events	事件表	4NF	Id 是唯一候选码，其余属性之间不存在函数依赖
event_types	事件类型表	4NF	Id 是唯一候选码，其余属性之间不存在函数依赖
user_events	事件记录表	4NF	user_id 与 event_id 互相多值推出，此外无多值依赖。
user_now_events	对应 Now 关系	2NF	user_id 为码，event_id 推出 event_name，传递依赖

user_invitations	对应 Invitation 关系	4NF	user_id 与 event_id 互相多值推出，此外无多值依赖。
user_event_types	对应 Own 关系	4NF	user_id 与 event_type_id 互相多值推出，此外无多值依赖。
user_friends	对应 Friend 关系	4NF	user_id 与 user_id 互相多值推出，此外无多值依赖。
user_friend_requests	对应 Friend 关系	4NF	user_id 与 user_id 互相多值推出，此外无多值依赖。
password_resets	对应 Reset_token	4NF	email 是唯一候选码，其余属性之间不存在函数依赖

可以看到部分表中存在冗余信息，如 event_name 等，这些信息是考虑到这些表会被频繁的访问，与其每次都从另一张中读取，不如就将其记录下来。

所有用户的外键采用 Cascade 级联删除，所有 Event 外键与 Event_type 外键采用 Restrict 删除，保证数据安全性。

使用了三个 Trigger，分别是：

user_friends 的 Insert，操作是对应的在 user_friend_requests 中加入请求。

user_friend_requests 的 Delete，操作是对应在 user_friends 的删除，即如果对方拒绝，删除好友请求信息。

user_events 的 Insert，操作是对应在 user_event_types 中检查是否存在新加入事件的类型，不存在则加入，这是针对邀请活动功能的措施。

针对每一个用户，生成该用户在 user_events 中的视图，这是考虑到每一个用户应该只能看到自己的事件，而事件是规模最庞大、操作最频繁的数据，生成视图也有助于加快访问速度。

使用了一个存储过程，用于递归得到一个事件类型的所有子类型，用于数据统计功能。

4.3 关系数据库模式之上的操作

验证模块：用户的注册、登录、注销与密码找回，这个主要是通过 Laravel 自带的验证系统实现，数据库方面不存在太大难点。

事件模块：事件的展示、事件类型的选择与添加都是最简单的 Select 与 Insert，不做赘述。主要在于事件的建立，构建了一个事务：

```
DB::transaction(function() use($eventPara,$user_eventPara,$user_now_eventPara)
        {
            $event=Event::create($eventPara);
            $user_eventPara['event_id'] = $event->id;
            $user_now_eventPara['event_id'] = $event->id;
            User_event::create($user_eventPara);
            User_now_event::create($user_now_eventPara);
        });
```

需要建立事件，建立记录，同时更新用户当前所做的事，一旦出错需要回滚，所以使用事务。

朋友模块：包括朋友的查找、请求与添加。朋友的查找就是简单的 Select。获取自己的好友名单需要一个复杂 SQL：

```
SELECT user_friends. friend_user_id, user_friends.friend_user_name,
       user_now_events.event_name, user_friends.confirmed
       FROM user_friends, user_now_events
       WHERE user_friends.confirmed = true
              AND user_friends.friend_user_id=user_now_events.user_id
                  AND user_friends.user_id=$user_id;
```

user_friends 与 user_friend_requests 中的 confirmed 都是表示对方是否同意。前者用于查看是否朋友关系，后者用于接受与拒绝。请求与拒绝分别利用一个 Trigger 实现：

请求：

```
CREATE TRIGGER send_request
AFTER INSERT on user_friends
FOR each ROW
       INSERT INTO  user_friend_requests (user_id,request_user_id,request_user_name,
confirmed ,created_at,updated_at)
       SELECT new.friend_user_id, new.user_id, users.name, false , now(), now() FROM
users
       WHERE
                  new.confirmed=false AND users.id=new.user_id;
```

拒绝：

```
CREATE TRIGGER request_refuse
AFTER DELETE on user_friend_requests
FOR each ROW
      DELETE     FROM     user_friends     WHERE     old.confirmed=false     AND
user_friends.user_id=old.request_user_id    AND    user_friends.friend_user_id    =
old.user_id;
```

邀请模块：对于进行中的事件，能够邀请朋友共同参加。邀请就是在创建成功之后，对于邀请名单中每个用户发送邀请，即添加到 user_invitation 中，同时 host 的事件结束之后，要终止该事件的所有邀请。

此外，由于邀请事件可能不在被邀人的事件类型表中，因此使用一个 trigger 来自动添加类型：

```
CREATE TRIGGER check_user_own_this_type
BEFORE INSERT on user_events
FOR each ROW
       INSERT INTO  user_event_types
(user_id,event_type_id,event_type_name,created_at,updated_at)
       SELECT new.user_id, events.event_type_id, event_types.name, now(), now() FROM
event_types,events
       WHERE
              new.event_id=events.id AND events.event_type_id=event_types.id
              AND events.event_type_id NOT IN
                        (SELECT    event_type_id    FROM    user_event_types    WHERE
user_id=new.user_id);
```

统计模块：对近期（一周）、根据日期、总计三种统计方式，统计人们每一种事件类型所耗费的时间。使

用复杂嵌套 SQL 与存储过程实现。存储过程用于求一个事件类型的所有子类型：

```
delimiter $$
CREATE FUNCTION `findAllChildEventType` (event_type_id INT)
RETURNS VARCHAR(4000)
BEGIN
DECLARE sTemp VARCHAR(4000);
DECLARE sTempChd VARCHAR(4000);
SET sTemp = '$';
SET sTempChd = cast(event_type_id as char);
WHILE sTempChd is not NULL DO
SET sTemp = CONCAT(sTemp,',',sTempChd);
SELECT group_concat(id) INTO sTempChd FROM event_types where
FIND_IN_SET(parent_event_type_id,sTempChd)>0;
END WHILE;
return sTemp;
END$$
delimiter ;
```

统计利用复杂嵌套 SQL 与扩展 GroupBy 功能实现：（以 1 号用户为例）：

```
SELECT DATE_FORMAT(uv.start_time, "%Y-%m-%d") AS day, user_event_types.event_type_id,
       user_event_types.event_type_name,
       sum(timestampdiff(minute,uv.start_time,uv.end_time)) AS totaltime
       FROM user_event_types, user_1_view AS uv,events
       WHERE user_event_types.user_id = 1 AND uv.end_time>0
              AND uv.event_id=events.id
              AND
FIND_IN_SET(events.event_type_id,findAllChildEventType(user_event_types.event_type_id
))
          GROUP BY user_event_types.event_type_id , day WITH ROLLUP
```

5 总结

通过这一次项目，我极为全面的对数据库技术进行了一次实践，并且在项目实现与设计的过程中，不断提出问题、解决问题的过程中，对数据库的设计原理也有了全新的认识。但就现在为止，这些都还是数据库基本应用与设计，希望在今后的学习中，我能够利用这些知识，提出一些更具有创新性的内容来。

References:

[1] https://www.mysql.com/

[2] http://www.golaravel.com/

[3] http://laravel.com/