**深 圳 大 学 实 验 报 告**

**课程名称：­ 数据库系统**

**实验项目名称： 实验四 Implement the RBMS**

**学院： 计算机与软件学院**

**专业： 网络工程**

**指导教师： 马里佳**

**报告人：吴嘉鸿 学号：2017153040 班级： 网工1班**

**实验时间： 2019.11.24-2019.12.24**

**实验报告提交时间： 2019.12.24**

**教务处制**

|  |  |
| --- | --- |
| 实验目的 1、了解DBMS系统的功能、Web based数据库的软件组成及工具；  2、掌握数据库软件的使用方法、  3、掌握php＋mysql的数据库操作方法。 | |
| 实验要求 1、安装相关软件并浏览软件自带的帮助文件和功能菜单，  2、掌握PHP构建网页以及连接数据库的方法。  3、掌握phpmyadmin创建数据库与数据表的方法；  4、了解mysql的命令以及与php衔接的语句  5、了解网页构造的原理  6、掌握dreamweaver的基本操作； | |
| **实验内容：**  **实现一个RBMS系统，实现商品的展示和数据的管理**  **主要内容包括：**   1. Preparation (5 points) 2. MySQL Implementation (50 points) 3. Interface (35 points) 4. Documentation (10 points) 5. Hand-ins, Demo and Grading |
| **实验步骤：**  **1. 准备 Preparation (5 points)**  **首先在Mysql数据库数据库中创建如下的数据表**   * Employees(eid, ename, city) * Customers(cid, cname, city, visits\_made, last\_visit\_time) * Products(pid, pname, qoh, qoh\_threshold, original\_price, discnt\_rate, sid) * Suppliers(sid, sname, city, telephone) * Purchases(pur, cid, eid, pid, qty, pdate, total\_price) * Logs(logid, who, time, table\_name, operation, key\_value)   上面的六个表分别是雇员表、客户表、产品信息表、供应商表、购买记录表以及日志表  **使用phpMyAdmin以及文档所给的定义语句进行创建表**  新建一个新的数据库rb，在rb下面创建上面的六个表  **2. MySQL实现 Implementation (50 points)**  **(1) Task 1 :**  (6 points) Write a stored procedure to show the tuples in each table. For example, you can implement a procedure, say **show\_products()**, to display all products in the products table   * **实现方法**：建立一个存储过程，对于传过来的表名，可以返回表格的所有信息   一般情况下，MySQL的存储过程不接受表名变量作为参数，否则会报错。是存储过程没有正确识别出传递给它的表名参数，因为传入的参数类型的长度或许会和表名的长度不同，所以导致识别不了。  但是我们可以使用prepare语句绕过这个错误，将整条sql语句作为变量，其中穿插变量作为表名，然后用execute调用该语句即可，prepare语句对语句进行预处理代码如下：  delimiter $$  CREATE PROCEDURE show\_table(in tame varchar(15))  BEGIN  declare table\_name varchar(15) default '';  set table\_name = tame;  set @statements = **concat**('select \* from ',table\_name);  prepare stmt from @statements;  execute stmt;  deallocate prepare stmt;  END  $$  delimiter ;  其中delimiter $$语句,是将结束标识符转化为$$,而deallocate prepare stmt;可以释放执行中使用的所有数据库资源（如游标）  调用可以使用以下的语句：  call show\_table('customers');#查询表customers  call show\_table('employees');#查询表employees  call show\_table('logs');     #查询表logs  call show\_table('products'); #查询表products  call show\_table('purchases');#查询表purchases  call show\_table('suppliers');#查询表suppliers  **命令行检验**   * 显示表customers  * 显示表employees   **(2) Task 2：**  (4 points) Write a procedure to report the monthly sale information for any given product. For example, you can use a procedure, say **report*monthly*sale(prod\_id)**, for this operation.   * **实现方法**：建立一个存储过程，接收传过来的产品id号，然后根据这个pid号在products和purchases两个表进行多表查询，分别查出这个产品的产品名字、销量、时间（包括年和月）、总销售额和总销量，最后还需要计算出平均价格   创建存储过程**report*monthly*sale(prod\_id)**的mysql语句如下：  delimiter $$  CREATE PROCEDURE report\_monthly\_sale(in prod\_id varchar(4))  BEGIN  select pr.pid,pname,**left**(MONTHNAME(ptime),3) as **month**, **YEAR**(ptime) as **year**,  **sum**(qty) as total\_quantity,**sum**(total\_price) as total\_price,  **sum**(total\_price)/**sum**(qty) as average\_sale\_price  from products pr,purchases pu  where pr.pid=prod\_id and pr.pid=pu.pid group by **month**,**year** order by **year**(ptime);  END  $$  delimiter ;  *-- MONTHNAME获取月份英文名*  *-- year函数获取一个时间中的年份*  *-- left函数获取缩写*  *-- 利用group by将同一年同一月份的数据整合起来*  *-- Sum函数求和*  **调用方法**  # 删除存储过程  drop procedure report\_monthly\_sale;  # 删除视图  drop view all\_pname;  # 调用存储过程  call report\_monthly\_sale('pr00');  **命令行检验**  展示产品pr00的月季度销售情况  展示产品pr01的月季度销售情况  **(3) Task 3**  (7 points) Write procedures to add tuples into the purchases table and the products table.  **实现**方法：建立存储过程实现将一条购买记录插入到数据表purchases中，其中   * 需要**参数**：产品号pur，客户id，雇佣id，产品id，质量q * 需要计算**平均价格**，均价等于总销售额除以总销量 * **插入时间**直接使用插入数据的时间   (1) 构建一个将数据插入到表purchases的存储过程  delimiter $$  create procedure add\_purchases  (in pur\_no int,in c\_id varchar(4),in e\_id varchar(3),in p\_id varchar(4),in pur\_qty int)  BEGIN  *-- 将查询到的折后价格存到临时变量@a\_price中*  select original\_price\*(1-discnt\_rate) from products where pid=p\_id into @a\_price;  *-- 插入数据*  insert into purchases values  (pur\_no,c\_id,e\_id,p\_id,pur\_qty,now(),pur\_qty\*@a\_price);  END  $$  delimiter ;  调用语句如下：  call add\_purchases(45, 'c003', 'e02', 'pr05', 1);  (2) 构建一个将数据插入到表products的存储过程  delimiter $$  create procedure add\_products  (in p\_id varchar(4),in p\_name varchar(15),in p\_qoh int,in qoh\_t int,in o\_price decimal(6,2),in discnt\_r decimal(3,2),in s\_id varchar(2))  BEGIN  *-- 插入数据*  insert into products values  (p\_id,p\_name,p\_qoh,qoh\_t,o\_price,discnt\_r,s\_id);  END  $$  delimiter ;  调用语句如下：  call add\_products('pr06', 'apple', 5, 5, 7.99, 0.1, 's1');  **命令行检验：**  **(4) Task 4:**  .(9 points) Add a tuple to the logs table automatically whenever any table is modified.  触发器就是某个表发生一个事件（增删改操作），然后自动的执行预先编译好的SQL语句，执行相关操作。触发器事件跟触发器中的SQL语句是原子性的（要么同时执行，要么同时不执行），这样保证了数据的完整性。  涉及触发的事件:   * + 插入一条数据到表purchases中   + 更新表products的qoh值   + 更新表customers的客户的访问次数   实现三个触发器，每一个触发器对应一个事件  **事件一触发器：(插入数据到表purchases)**  三个属性的值为 table\_name='purchases';operation='insert';key-value='pur'.  代码如下：  DELIMITER $$  drop trigger if exists rb.insert\_purchases; #删除触发器  create trigger insert\_purchases  *-- 创建名为insert\_purchases的触发器*  after insert on purchases  *-- 设定触发时间为插入数据到表purchases之后*  for each row   *-- 这句话在mysql是固定的，表示任何一条记录上的操作满足触发事件都会触发该触发器。*  begin  *-- 触发执行的语句*  *-- 获取用户名保存到变量@u*      select **left**(user(),instr(user(),'@')-1) into @u;      insert into logs(who,time,table\_name,operation,key\_value) values(@u,now(),'purchases','insert',new.pur); *-- NEW用来表示将要(before)或已经(after)插入的新数据。*  end$$  DELIMITER ;  **事件二触发器：(更新表products的数据)**  三个属性的值为 :  table\_name='products' ; operation='update' ; key-value='pid'  代码如下：  DELIMITER $$  drop trigger if exists rb.update\_products;  create trigger update\_products  after update on products  for each row  begin      select **left**(user(),instr(user(),'@')-1) into @u;      insert into logs(who,time,table\_name,operation,key\_value) values(@u,now(),'products','update',new.pid);  end$$  DELIMITER ;  **事件三触发器：(更新表customers的数据)**  三个属性的值为:  table\_name=customers' ; operation='update' ; key-value='cid'  代码如下：  DELIMITER $$  drop trigger if exists rb.update\_customers;  create trigger update\_customers  after update on customers  for each row  begin      select **left**(user(),instr(user(),'@')-1) into @u;      insert into logs(who,time,table\_name,operation,key\_value) values(@u,now(),'customers','update',new.cid);  end$$  DELIMITER ;  **(5) Task 5：**  (4 points) Before a purchase is actually made (i.e., before a tuple is added into the purchases table), your program needs to make sure that, for the involved product, the quantity to be purchased is equal to or smaller than the quantity on hand (qoh). Otherwise, an appropriate message should be displayed (e.g., “Insufficient quantity in stock.”) and the purchase request should be rejected.   * 实现方法：在实际购买之前，程序需要确保对于产品购买的数量等于或小于手头的数量(qoh值)。否则，应显示适当的消息，可以显示“库存不足”，而且购买请求不会执行。   应该重新编写存储过程add\_purchase添加了对购买数量的判断，小于阈值才能够进行交易，代码如下，  DELIMITER $$  drop procedure add\_purchases$$  *-- 删除*  create procedure add\_purchases  *-- 重新建立*  (in pur\_no int,in c\_id varchar(4),in e\_id varchar(3),in p\_id varchar(4),in pur\_qty int)  BEGIN  select qoh into @p\_qoh from products where pid = p\_id;  *-- 获取阈值*  if @p\_qoh >= pur\_qty then   *-- 假如购买的数量不大于手头的阈值，可以进行购买*  update products set qoh=(qoh-pur\_qty) where pid = p\_id; *-- 更新产品的阈值*  *-- 将查询到的平均价格存到临时变量@a\_price中*  select original\_price\*(1-discnt\_rate) from products where pid=p\_id into @a\_price;  *-- 插入数据*  insert into purchases values  (pur\_no,c\_id,e\_id,p\_id,pur\_qty,now(),pur\_qty\*@a\_price);  else  select qoh into @tq from products where pid=p\_id;  select **concat**('库存不足！.最大容量为',@tq) as **error\_message**;  end if;  END  $$  DELIMITER ;  **(6) Task 6：**  (16 points) After adding a tuple to the purchases table, the qoh column of the products table should be modified accordingly; that is, the qoh of the product involved in the purchase should be reduced by the quantity purchased.   * 触发器执行的任务：   (1) 打印一条信息指出产品当前的qoh  (2) 自动设置为2\*oldqoh增加qoh  (3) 打印另一条消息指出手头数量已经增加了旧qoh+qty 售出  **创建触发器**  DELIMITER $$  drop trigger insert\_purchases$$  create trigger insert\_purchases after insert on purchases  for each row  BEGIN      select **left**(user(),instr(user(),'@')-1) into @u;      insert into logs(who,time,table\_name,operation,key\_value) values(@u,now(),'purchases','insert',new.pur);    *-- 插入日志数据*      select qoh into @rest\_qoh from products where pid = NEW.pid;*-- 获取剩下的数目到@rest\_qoh*      select qoh\_threshold into @qoh\_t from products where pid = NEW.pid; *-- 获取阈值*      update customers set visits\_made=(visits\_made+1) , last\_visit\_time = now() where cid= NEW.cid;      if @rest\_qoh < @qoh\_t then          update products set qoh=((qoh+NEW.qty)\*2) where pid= NEW.pid;   *-- old*          select **concat**(' the current qoh of the product  is ',@rest\_qoh) as **error\_message**;  end if;  END$$  DELIMITER ;  上面的代码会报Not allowed to return a result set from a trigger错误，触发器不能够返回集合信息  **重新修改触发器代码**  DELIMITER $$  drop trigger insert\_purchases$$  create trigger insert\_purchases after insert on purchases  for each row  BEGIN      select **left**(user(),instr(user(),'@')-1) into @u;      insert into logs(who,time,table\_name,operation,key\_value) values(@u,now(),'purchases','insert',new.pur);    *-- 插入日志数据*      select qoh into @rest\_qoh from products where pid = NEW.pid;*-- 获取剩下的数目到@rest\_qoh*      select qoh\_threshold into @qoh\_t from products where pid = NEW.pid; *-- 获取阈值*      update customers set visits\_made=(visits\_made+1) , last\_visit\_time = now() where cid= NEW.cid;  if @rest\_qoh < @qoh\_t then      update products set qoh=((qoh+NEW.qty)\*2) where pid=NEW.pid;    *-- old*  *-- 输出信息g*  end if;  END  $$  DELIMITER ;  由于触发器里面不能够输出信息，考虑把信息直接在PHP中输出，在PHP文件中查找当前产品数量q，再查找阈值q\_t,进行计算之后判断是否输出信息。  **(7) Task 7：**  (4 points) You need to make your code user friendly by designing and displaying appropriate messages for all exceptions. For example, if someone wants to find the purchases of a customer but entered a non-existent customer id, your program should report the problem clearly.  创建一个根据用户id查询购买信息的一个存储过程，需要考虑用户id不存在的情况  DELIMITER $$  drop procedure find\_customer\_purInfo$$  create procedure find\_customer\_purInfo(in c\_id  VARCHAR(4))  BEGIN      select cid into @var from customers where cid=c\_id;  if @var !=c\_id then *-- 没有查询到客户id*      select ' This CustomerID does not exist!' as msg\_error;  else      select \* from purchases where cid=c\_id;  end if;  END  $$  DELIMITER ;  **命令行测试:**  查询成功  查询失败  **3.** **Interface (35 points) PHP实现**  使用 PHP 实现 Web 交互界面。   1. **登陆界面**   包括用户注册和用户登陆以及密码提示,简单的HTML页面实现，设置一个粉红色的背景图片，然后新建一个两层的表单，然后提交表单到文件conn\_and\_display.php文件中进行判断用户可否登陆，而注册界面的两行数据则是提交到register.php文件中调用root账户进行注册并且赋予权限，如下是主要的代码  数据库创建用户和赋予权限的代码  CREATE USER 'Victor'@'localhost' IDENTIFIED BY '19981022';  *-- 创建用户及密码*  GRANT ALL ON rb.\* TO 'Victor'@'localhost';  *-- 赋予权限*  FLUSH PRIVILEGES;  实现的效果如下：  图表 1 登陆界面  图表 2 注册界面  图表 3 注册成功   1. **展示商品信息**   新建一个PHP文件disPro.php展示商品的信息，通过调用数据库进行查询将得到的信息返回给前端，其中使用的函数封装在operate.php文件中，用一个函数showProducts()实现，如下是具体的代码，**调用存储过程show\_table，参数为products**即可。  PHP网页实现效果如下：  图表 4 商品主界面  **实现购买商品弹出框：**  使用JS进行调用一个静态的购买栏，设计的原理是首先在PHP网页中新建一个固定大小的在页面正中央的购买页，首先让它不显示，然后当鼠标左键单击的时候会自动调用JS使得购买栏显示，就实现弹出框的效果了。而其中调用的数据库操作只需要调用存储过程add\_purchases，参数需要进行补全  实现的JS主要代码如下：  实现的效果如下：  图表 5 弹出购买栏  在上面的弹出框中包含PHP代码以及一个表单，当点击购买的按钮会将表单提交到add\_purchases.php文件中进行数据更新，需要调用Mysql的存储过程进行更新，并且会**触发触发器insert\_purchases。**插入数据logs表会更新。  购买成功会有如下的提示：  库存不足会自动进行进货，增加产品的数量然后输出提示信息   1. **展示RBMS数据表和查询数据**   **实现展示出所有的RBMS数据信息**  首先新建一个文件为demo.php，该文件进行展示所有的RBMS中的数据表，然后通过混合HTML、PHP以及JS进行输出表格中的信息，  在PHP文件中用函数进行封装。在PHP已经实现两个函数，一个接收参数为表名然后返回输出的二维数组数据，另一个是展示所有表名的函数，默认进入该界面会展示所有的表格名。类似的需要调用上面说的存储过程。  如下是关键代码：  图表 6 查询表信息  图表 7 查询表名  实现效果如下:  图表 8 customers表  图表 9 logs表  **实现查询销量和购买信息**  使用MySQL查询，PHP获得查询到的二维数组进行循环输出查询到的表格信息。一个用于查询商品的销售记录，一个查询用户的购买记录。  主要函数如下：  最终页面的实现效果如下:  图表 10 查询输出商品的销量信息  图表 11 查询输出用户id的购买记录  PHP主要函数作用如下:   1. 1. displayTablesName() *列表形式展示所有表名* 2. 2. function descTable($table\_name) *获取表格所有的属性名* 3. 3. function showProducts() *展示所有的产品* 4. 4. function showTinfo($table\_name) *表格形式展示表中的所有信息* 5. 5. get\_products\_names() *获取所有的产品名称* 6. 6. function showAllSales() *展示所有的销售记录* 7. 7. function report\_monthly\_sale(pid) *报告月季度销量* 8. 8. function report\_customer\_pur($cid) *报告用户购买记录* |
| **实验结果：**  （此页附完成的实验结果、并给出个人对结果的分析、结论）   1. 实现了一个简单的RBMS系统，能够对商品信息进行管理和查询 。 2. 实现了MySQL数据库存储RBMS的数据然后编写了主要的存储程序和触发器，使得插入数据与更新数据更加方便。 3. PHP+HTML实现用户登陆界面的功能。 4. 使用PHP语言，还拓展使用了JS，编写了前端的页面展示RBMS的商品信息和数据的表格信息。 5. 实现PHP语言编写一个页面能够对输入的产品id查询月季度销售情况，对输入的客户id，能够查询该用户的购买记录信息。 |
| **实验小结：**  （实验中出现问题的解决方法，实验心得体会等）   1. 用MYSQL数据库存储数据并且在MYSQL中进行编写存储程序和触发器，可以大大的见减少人工的繁琐操作，只需要在PHP中调用一个存储程序就能够得到返回结果，非常便利。 2. 实验中涉及许多变量调用的问题，比如PHP希望调用JS中的变量，还有PHP希望调用HTML的数据，发现同语言的变量进行调用非常麻烦，很难进行传输。解决方法是尽量少设置不同语言之间互相使用的变量。 3. 实现购买栏弹出框单单使用HTML是很难实现的，调用以下JS的语言脚本就能够比较简单的实现一个静态的弹出框，而那些动态的弹出框则需要更加复杂的JS语法。 4. 实现一个好看的网页通常需要多种语言的配合，例如HTML负责网页架构、JS负责网页脚本、CSS负责网页的美化，而PHP负责与数据库的连接调用。多种语言的协同往往效果更好。 |
| **指导教师批阅意见：**  **成绩评定：**  **指导教师签字：**  年 月 日 |
| 备注： |