WOLFPUB DB

For WolfCity publishing house

CSC 540 Database Management Systems Project Report #3

Apr 13, 2020.

Swathi Dinakaran(sdinaka)
Sameer Thummalapally(vthumma)
Sri Harsha Varma Uppalapati(suppala)
Sai Rama Mohan Reddy Singamala(ssingam2)

Assumptions:

1. Articles are written by Journalists and each article has exactly one journalist.

2. Distributors can order only one type of publication at a time.

3. Every Publication will have a unique id, which shall be created at the time of insertion.

4. Editors, authors will be paid as salary for staff employees and pay for invited employees.

5. Journalists are a part of the staff and are paid just as editors and authors.

6. Staff members have a unique employee id.

7. Payments shall be done at the end of every month.

8. A book can have one or more authors.

9. For staff payments, Initially, Payment Date and date of collection are the same, and we have the option to update the date of the collection later.

10. When the Order is made the order value is automatically added to the Owe's section of the Distributor. And in the same way when a distributor makes a payment the owe's section is updated.

Transactions:

1. Deleting Periodic Issue:

```
    private static PreparedStatement deletePeriodicIssueQuery;

   public static void deletePeriodicIssue_sql(int PID){
      DBManager.beginTransaction();
           String sql;
      sql = "DELETE FROM `Publications`" + " WHERE `PID` = ?;";
        deletePeriodicIssueQuery= connection.prepareStatement(sql);
      deletePeriodicIssueQuery.setInt(1, PID);
           if(!DBManager.execute(deletePeriodicIssueQuery)){
                    System.out.println("error in deleting periodic issue");
                    DBManager.rollbackTransaction();
                    return;
           }
           DBManager.commitTransaction();
  }

         public static void deletePeriodicIssue(){
      try {
         // Get staff id
         System.out.println("\nEnter the Publication ID of the Periodic issue you want to
delete:\n");
         int PID = s.nextInt();
         s.nextLine();
         // Call method that interacts with the Database
         deletePeriodicIssue_sql(PID);
         System.out.println("The Periodic issue is deleted successfully!");
      } catch (Throwable err) {
         err.printStackTrace();
```

```
    }

  }
```

2.Dirstributor making an order:

```
        public static void insertIntoOrder_Tables_sql(int OID,int noc,String
order_date,String order_due_date,int order_value,int shipping_cost){

            sql = "INSERT INTO `Order_tables` (`Orderno`, `noofcopies`, `orderdate`,
`orderduedate`, `ordervalue`, `shippingcost`) VALUES (?, ?, ?, ?, ?,?);";
            PreparedStatement insertIntoOrder_TablesQuery =
connection.prepareStatement(sql);
                DBManager.beginTransaction();
                insertIntoOrder_TablesQuery.setInt(1, OID);
          insertIntoOrder_TablesQuery.setInt(2, noc);
          insertIntoOrder_TablesQuery.setString(3, order_date);
          insertIntoOrder_TablesQuery.setString(4, order_due_date);
          insertIntoOrder_TablesQuery.setInt(5, order_value);
          insertIntoOrder_TablesQuery.setInt(6, shipping_cost);
          if(!DBManager.execute(insertIntoOrder_TablesQuery)){
                            System.out.println("error in inserting into order table");
                            DBManager.rollbackTransaction();
                            return;
                    }
                DBManager.commitTransaction();

        }

   public static void insertIntoOrders_sql(int PID, int DAccnumber,int OID,String Title){
        sql = "INSERT INTO `Orders` (`PID`, `DAccnumber`, `Orderno`, `Title`) VALUES
(?, ?, ?, ?);";
                  PreparedStatement insertIntoOrdersQuery =
connection.prepareStatement(sql);
                     DBManager.beginTransaction();
              insertIntoOrdersQuery.setInt(1, PID);
              insertIntoOrdersQuery.setInt(2, DAccnumber);
```

```
            insertIntoOrdersQuery.setInt(3, OID);
            insertIntoOrdersQuery.setString(4, Title);
            if(!DBManager.execute(insertIntoOrdersQuery)){
                        System.out.println("error in inserting into order table");
                        DBManager.rollbackTransaction();
                        return;
                    }
              DBManager.commitTransaction();
   }

        public static void addDistributorOwe_sql(int DAccnumber,int order_value){
        sql = "UPDATE `Distributors` set owe=owe+? WHERE DAccnumber=?;";
                PreparedStatement addDistributorOweQuery =
connection.prepareStatement(sql);
                DBManager.beginTransaction();
            addDistributorOweQuery.setInt(1, order_value);
            addDistributorOweQuery.setInt(2, DAccnumber);
            if(!DBManager.execute(addDistributorOweQuery)){
                        System.out.println("error in inserting into order table");
                        DBManager.rollbackTransaction();
                        return;
                }
           DBManager.commitTransaction();
   }



        public static void distributorOrder(){
        try {
          System.out.print("\n Input Order  \n> ");
           System.out.print("\n Enter the DAccnumber of the distributor who is making the
order\n");
                int DAccnumber = s.nextInt();
                s.nextLine();
        System.out.print("\n Enter the PID of the Publication the distributor want to
order\n");
        int PID=s.nextInt();
                s.nextLine();
        String title=getPIDTitle_sql(PID);
```

```
        int OID=newOrderId();
        System.out.print("\n Enter the Number of copies you want to order\n");
        int noc=s.nextInt();
                s.nextLine();
        System.out.print("\n Enter the Order daten");
        String order_date=s.nextLine();
        System.out.print("\n Enter the Order due daten");
        String order_due_date=s.nextLine();
        int price_PID=getPricePID_sql(PID);
        int order_value=price_PID*noc;
        int shipping_cost=order_value/10;

insertIntoOrder_Tables_sql(OID,noc,order_date,order_due_date,order_value,shipping_
cost);
        insertIntoOrders_sql(PID,DAccnumber,OID,title);
        addDistributorOwe_sql(DAccnumber,order_value);
                } catch (Throwable err) {
                        err.printStackTrace();
                }
        }
```

Design Decisions:

The system has a main menu that gives 5 options for 5 different kinds of operations and ask the user to enter the input which is a number between 0-5. The operations are as follows:

1. Editing and Publishing
2. Productions
3. Distribution
4. Reports
5. Exit

For each option, the user has selected we again display a submenu that has another set of operations listed. We have developed the system in such a way that the Main

menu is written in the main class and with the help of switch case we call the function that is related to that specific operation. This was done to break the application into more manageable and maintainable pieces of code.

Our program has 2 sets of helper functions. The first set of helper functions are used to display the list of entities in an entity set so that selecting a specific attribute value becomes easy. For example, when we want to update the certain attribute value of a publication, we first show all the publications in the table and then ask the user to enter the PID of the publication they want to update , then the attribute they want to update followed by value. By doing this way the user would have a clear idea of what they are doing. The other class is Input, which is a class that attempts to abstract the details of getting different types of input from the user, and includes a method for menu formatting, as well as a helper function that assists with constructing SQL UPDATE statements.

Functional Roles:

Part 1:

Software Engineer: Ram Mohan (Prime), Harsha (Backup)

Database Designer/Administrator: Harsha (Prime), Swathi(Backup)

Application Programmer: Swathi (Prime), Sameer(Backup)

Test Plan Engineer: Sameer(Prime), Ram Mohan(Backup)

Part 2:

Software Engineer: Sameer (Prime), Swathi (Backup)

Database Designer/Administrator: Swathi (Prime), Ram Mohan (Backup)

Application Programmer: Ram Mohan (Prime), Harsha (Backup)

Test Plan Engineer: Harsha (Prime), Sameer (Backup)

Part 3:

Software Engineer: Swathi (Prime), Ram Mohan(Backup)

Database Designer/Administrator: Sameer(Prime), Swathi(Backup)

Application Programmer: Harsha (Prime), Sameer(Backup)

Test Plan Engineer: Ram Mohan (Prime), Harsha(Backup)