CSC540 Database Management Systems

PROJECT REPORT 3

WolfPubDB Database Management System

By

Sneha Kumar, Shubham Dilip Pampattiwar, Sreeraksha Mavinhally Sreekantha,
Team X

REPORT - 3

1. **All corrections are highlighted in yellow.**

REPORT - 1 - CORRECTIONS

5. Missing APIs: -

- Assign Editor to Publication - **PAGE 4**

  This was already added in Report 1 -

  Maintaining Publications:
  mapStaffToPublication(StaffID, PID) -> return confirmation

  Updating the name for the API -
  mapEditorToPublication(StaffID, PID) -> return confirmation

- View information on Publication - **PAGE 5**

  This was already added in Report 1 -

  RetrieveInformation:
  findPublicationByTopic(topic) -> return Publication record
  findPublicationByDate(publicationDate) -> return Publication record
  findPublicationByAuthor(name) -> return Publication record

  Adding one more API:
  findPublication() -> return Publication records

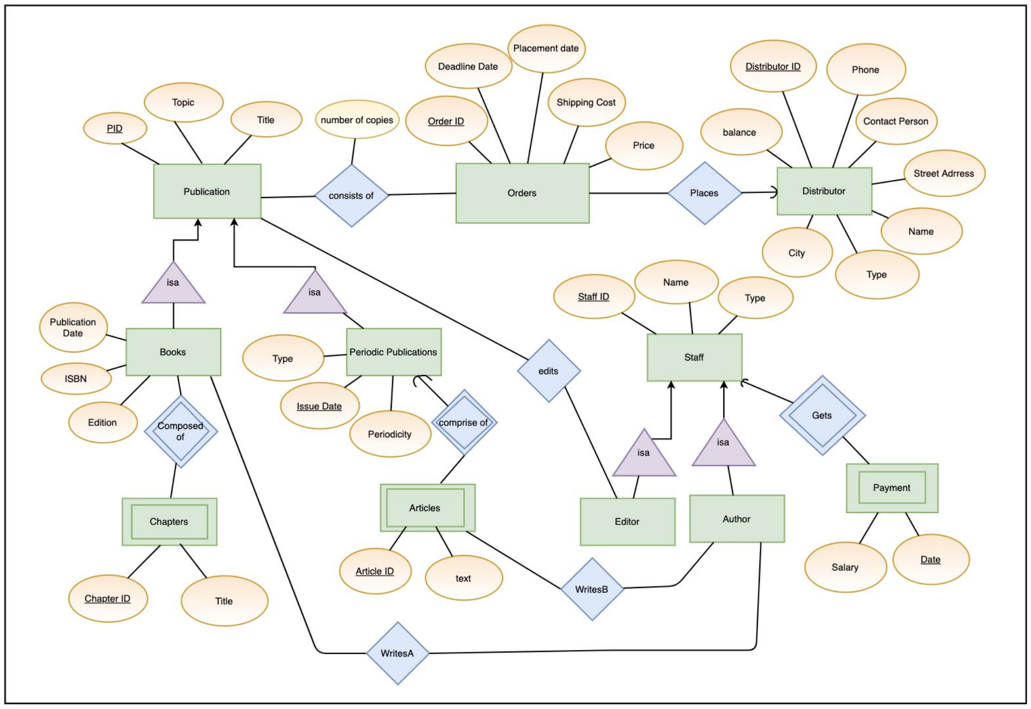
7. Subclass Key Attribute: - PAGE 7, 8, 9

Updating IssueDate in ER diagrams - as an attribute of periodic publication.(removing key attribute - which we had underlined earlier)

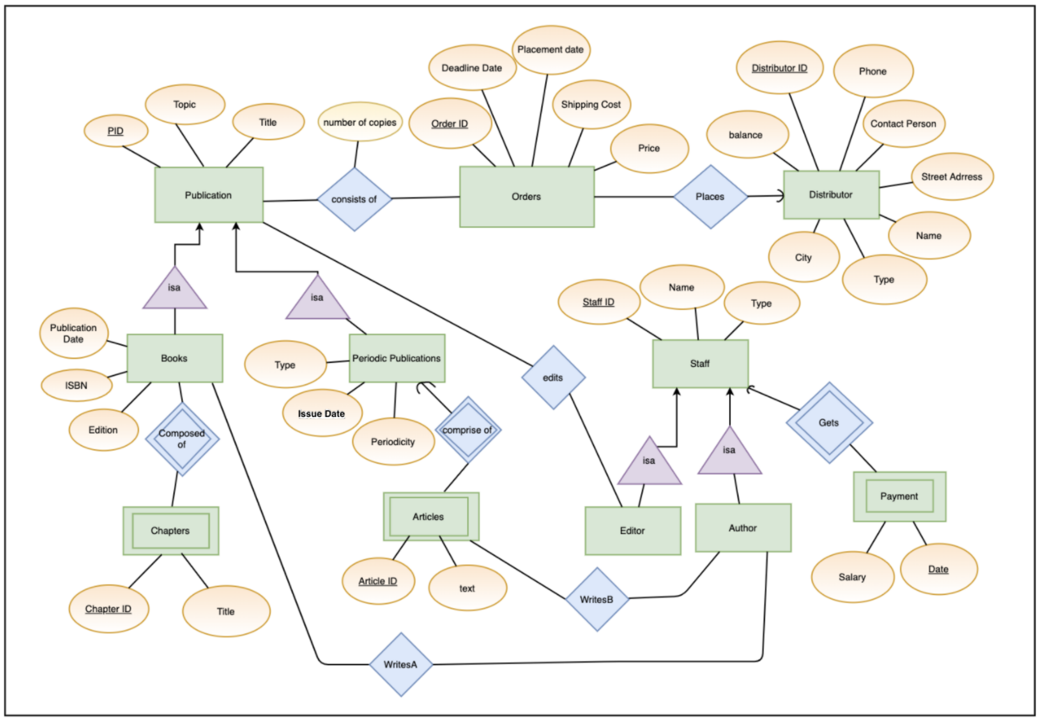
Updated ER Diagram:

Admin view:

Old:

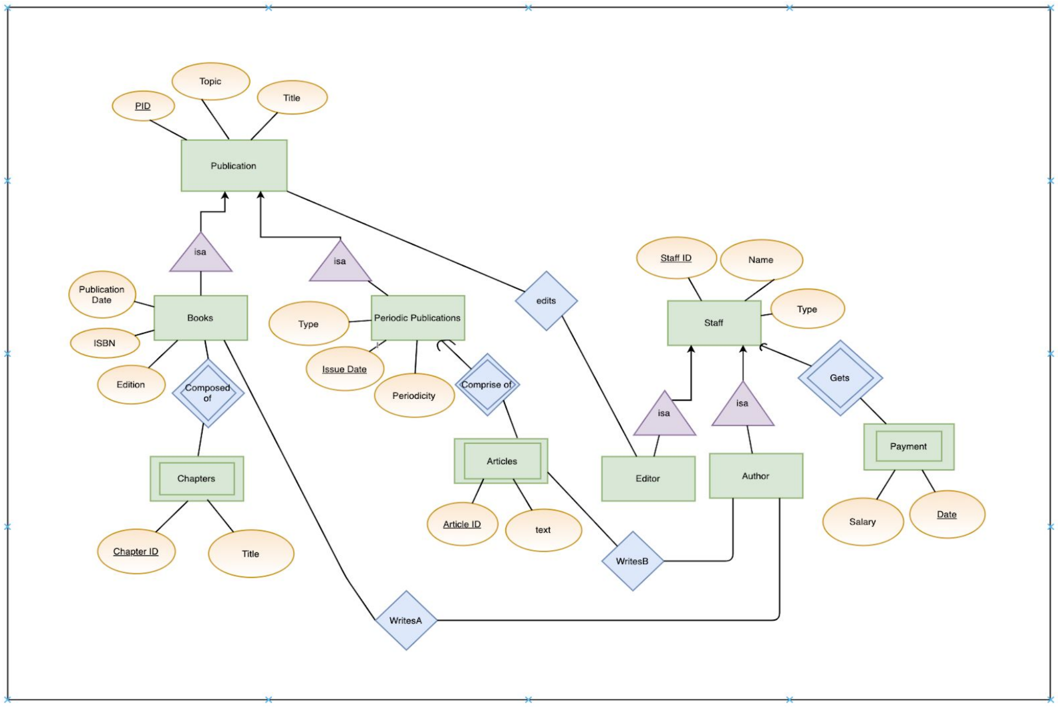


New:

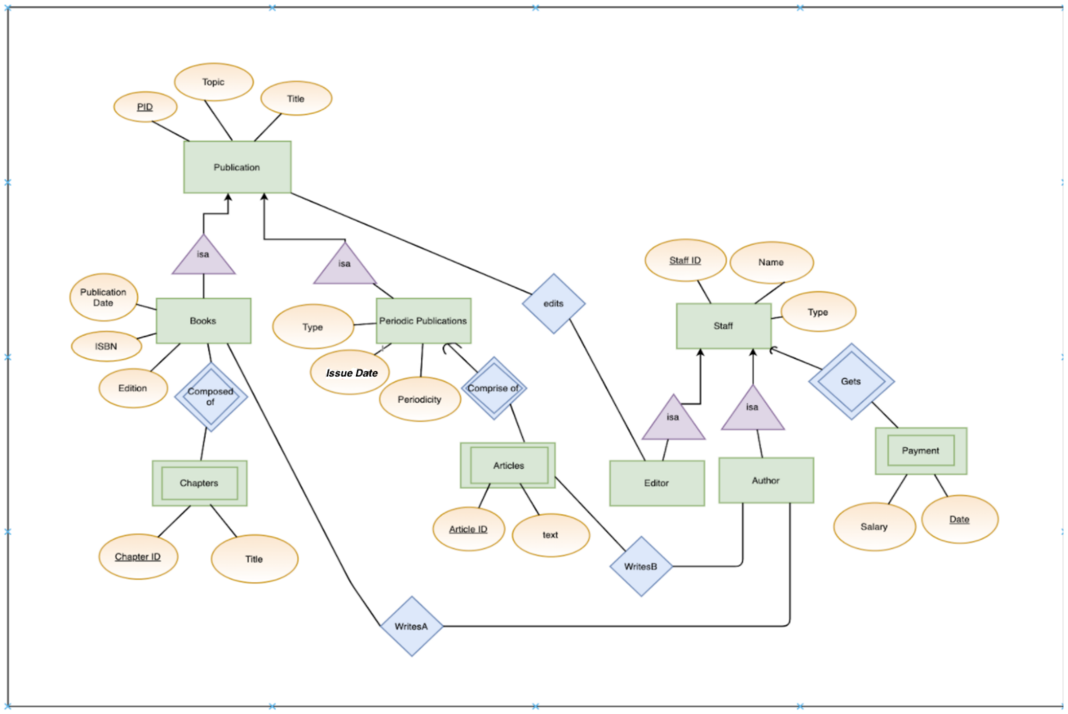


Staff view:

Old:

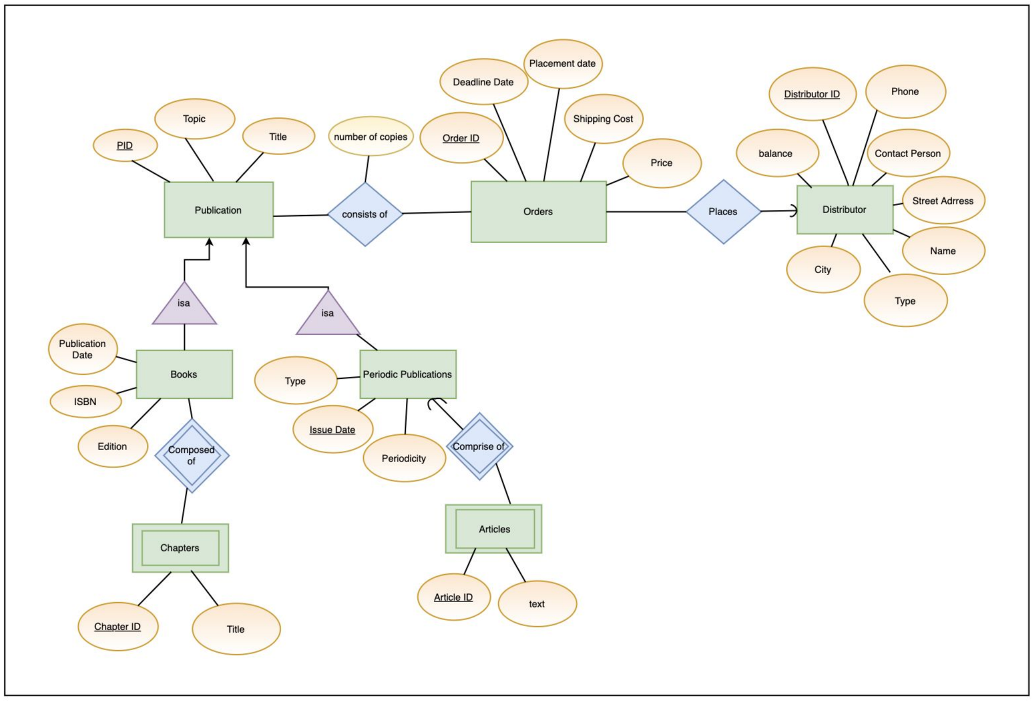


New:

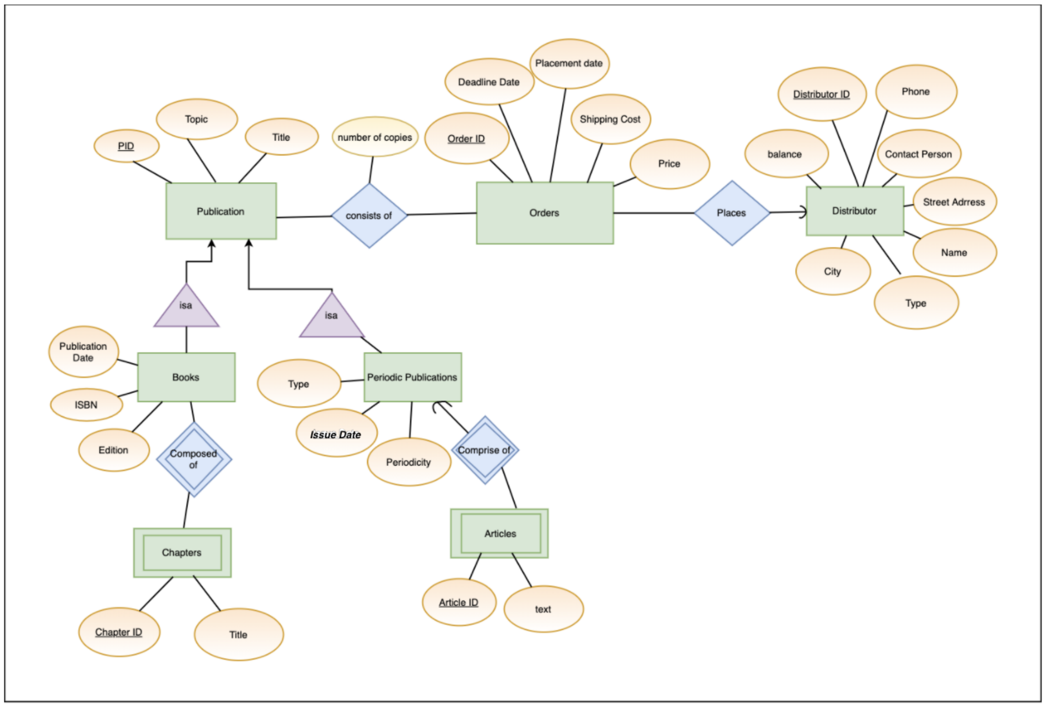


Distributor view:

Old:



New:



REPORT - 2 - CORRECTIONS

This is what was submitted as a part of report2 :

1. **Book(PID,Publication Date, ISBN, Edition) - PAGE 2**
   PID -> PID,Publication Date, ISBN, Edition

   Since Books is a subclass and Publication is the superclass, PID or the Publication ID uniquely identifies a book. Since LHS is super key, the functional dependency is in BCNF and therefore in 3NF.

   No other dependencies will hold. Publication date/edition can be the same for any number books. Therefore we cannot have these attributes determining other attributes(For example 2 books can be published on the same day or can have the same edition).

   Updating the description:

   Since Books is a subclass and Publication is the superclass, PID or the Publication ID uniquely identifies a book & thus determines all other attributes.

   Other dependencies:

   - **ISBN -> PID, Edition**

     Since every book or an edition will have a different ISBN, ISBN will uniquely identify PID and edition.
   - Edition doesn't uniquely identify other attributes because the value can be the same for any number books
   - Publication date can be the same for any number books.

Therefore we cannot have Edition/Publication date attributes determining other attributes(For example 2 books can be published on the same day or can have the same edition).

Since LHS is super key for both, the functional dependency is in BCNF and therefore in 3NF.

This is what was submitted as a part of report2 :

2. **Distributor(<u>DistributorID</u>, Name, type, Phone, Balance, Contact Person, Street Address, City) - PAGE 4**

   DistributorID -> DistributorID, Name, type, Phone, Balance, Contact Person, Street Address, City

   Every distributor has a unique distributor ID. This ID determines all other attributes associated with a distributor, making it the super key. Since LHS is super key, the functional dependency is in BCNF and therefore in 3NF.

   Name/type cannot determine other attributes of a distributor, 2 distributors can have the same name/type. Phone/Contact Person/Street/City - can also be the same for 2 or more distributors and cannot uniquely identify one particular distributor. Balance for 2/more distributors can be the same. Therefore, no other dependencies will hold.

   Updating the description:

   Every distributor has a unique distributor ID. This ID determines all other attributes associated with a distributor, making it the super key. Since LHS is super key, the functional dependency is in BCNF and therefore in 3NF.

   Name/type cannot determine other attributes of a distributor, 2 distributors can have the same name/type. Phone/Contact Person/Street/City - can also be the same for 2 or more distributors and cannot uniquely identify one particular distributor. Balance for 2/more distributors can be the same. Therefore, no other dependencies will hold.

   Name/Contact person cannot determine phone attribute because two/more people can have the same name but their phone number would differ. Therefore, only DistributorID can uniquely identify all other attributes

2. PAGE 7

- Publication(PID, Topic, Title, Pub_No)
  - ❑ PID is the primary key
  - ❑ Topic,Title and Pub_No are not allowed to be null
- Books(PID,Publication Date, ISBN, Edition)
  - ❑ PID is the primary key
  - ❑ PID is a foreign key which references PID attribute of Publication relation
  - ❑ Publication_date , ISBN, Edition are not allowed to be null
- PeriodicPublication(PID, Type, Issue_date,Periodicity)
  - ❑ PID is the primary key
  - ❑ PID is a foreign key which references PID attribute of Publication relation
  - ❑ Type , Issue_date, Periodicity are not allowed to be null
- Chapter(<u>PID</u>, <u>ChapterID</u>, Title)
  - ❑ PID and ChapterID forms the primary key
  - ❑ PID is a foreign key which references PID attribute of BOOK relation
  - ❑ Title is not allowed to be null
- Articles(<u>PID</u>, <u>ArticleID</u>, text)
  - ❑ PID and ArticleID forms the primary key
  - ❑ PID is a foreign key which references PID attribute of PERIODICPUBLICATION relation
  - ❑ Text is not allowed to be null
- Orders(<u>OrderID</u>, Deadline date, Placement date, Shipping cost, price)
  - ❑ OrderID is the primary key
  - ❑ Placement date, Shipping cost and price not allowed to be null
  - ❑ Deadline Date is allowed to be null if the deadline date for the order is not yet specified.
- Distributor(<u>DistributorID</u>, Name, type, Phone, Balance, Contact Person, Street Address, City)
  - ❑ DistributerID is the primary key

    - ❑ Name, type, Phone, Balance, Contact Person, Street Address and City are not allowed to be null
- Staff(StaffID, Name, Type)
    - ❑ StaffID is the primary key
    - ❑ Name, Type are not allowed to be null
- Editor(StaffID)
    - ❑ StaffID is the primary key
    - ❑ StaffID is a foreign key which references StaffID attribute of Staff relation
- Author(StaffID)
    - ❑ StaffID is the primary key
    - ❑ StaffID is a foreign key which references StaffID attribute of Staff relation
- Writes_Articles(StaffID, PID, ArticleID)
    - ❑ StaffID, PID and ArticleID forms the primary key
    - ❑ StaffID is a foreign key which references StaffID attribute of AUTHOR relation
    - ❑ (PID,ArticleID) is a foreign key which references (PID,ArticleID) attribute of ARTICLE relation
- Writes_Books(StaffID, PID)
    - ❑ StaffID and PID forms the primary key
    - ❑ StaffID is a foreign key which references StaffID attribute of AUTHOR relation
    - ❑ PID is a foreign key which references PID attribute of BOOK relation
- ConsistsOf(PID, OrderID, number of copies)

    - ❑ PID and OrderID forms the primary key
    - ❑ PID is a foreign key which references PID attribute of Publication relation
    - ❑ OrderID is a foreign key which references OrderID attribute of Orders relation

- Places(OrderID, DistributorID)
    - ❑ OrderID and DistributorID forms the primary key
    - ❑ OrderID is a foreign key which references OrderID attribute of Orders relation
    - ❑ DistributorID is a foreign key which references DistributorID attribute of Distributor relation
- Edits(StaffID, PID)
    - ❑ StaffID and PID forms the primary key
    - ❑ StaffID is a foreign key which references StaffID attribute of EDITOR relation
    - ❑ PID is a foreign key which references PID attribute of Publication relation

- Payment(StaffID, Date, Salary)
    - ❑ StaffID and Date forms the primary key
    - ❑ StaffID is a foreign key which references StaffID attribute of Staff relation
    - ❑ Salary is not allowed to be null

ASSUMPTIONS:

1. Enter Total Payment as 0 in the ORDERS table initially.
2. Value of Total Payment attribute in Orders Table is calculated as (Number of Copies * Price) + Shipping_Cost
3. Total revenue is calculated as Total payment - Shipping Cost
4. Every edition of a book has a different ISBN.
5. PUB_NO is unique to a particular book edition or periodic publication issue.

2. SOURCE CODE - SUBMITTED

3. TRANSACTIONS:

a) INSERT BOOK:

Transactions for 2 Insert operations: Insert into Publication and Insert into Book.
File: crud/PublicationCRUD.java

When entering book information, the first insertion checks if the PID already exists in the PUBLICATION table and accordingly accepts/rejects both insertion operations.

The second insertion checks if there is an entry with the same value as new ISBN in BOOK table. If yes, rejects the insertion operations and rolls back both.

```
public static boolean insertPublication(Integer PID, String topic, String title, String pub_no, String PublicationDate,
                                        String ISBN, Integer Edition) throws SQLException {
  boolean trans1 = false;
  boolean trans2 = false;
  Connection conn = null;
  try {
     conn = DbConnection.getConnection();
     conn.setAutoCommit(false);
     String query = "insert into PUBLICATION(PID, TOPIC, TITLE, PUB_NO) values (?,?,?,?)";
     PreparedStatement st = conn.prepareStatement(query);
     st.setInt(1, PID);
     st.setString(2, topic);
     st.setString(3, title);
     st.setString(4, pub_no);
     st.executeUpdate();

     trans1 = true;
     trans2 = BookCRUD.insertBook(conn, PID, PublicationDate, ISBN, Edition);
```

```
        if(trans1 && trans2){
            conn.commit();
            System.out.println("Transaction successful");
            return true;
        }else{
            conn.rollback();
            System.out.println("Transaction Failed");
            return false;
        }
    } catch (SQLException ex) {
        conn.rollback();
        System.out.println("Transaction Failed");
        return false;
    } finally {
        if(conn != null){
            conn.setAutoCommit(true);
        }
    }
}


public static boolean insertBook(Connection conn, Integer PID, String publicationdate, String isbn, Integer edition)
throws SQLException {
        try {
//          Connection conn = DbConnection.getConnection();
            String query = "insert into BOOK(PID, PUBLICATIONDATE, ISBN, EDITION) values (?,?,?,?)";
            PreparedStatement st = conn.prepareStatement(query);
            st.setInt(1, PID);
            st.setString(2, publicationdate);
            st.setString(3, isbn);
            st.setInt(4, edition);
            st.executeUpdate();

            return true;
        } catch (SQLException ex) {
          // ex.printStackTrace();
            return false;
        }
    }
```

b) INSERT ORDERS:

Transaction for 3 insert operations - ORDERS, PLACES & CONSISTS OF
File: transaction/insertTransactions.java

First insertion is to insert into the ORDERS table with a new OrderID, If it already exists rejects the insertion operation & transaction fails.

Second insertion is to insert into the PLACES table with a OrderID of the new order & existing DID, if DID does not exist the transaction fails - both the insertion fails.

Third insertion is to insert into the CONSISTSOF table with a OrderID of the new order & existing PID, if PID does not exist the transaction fails - all three insertions will fail.

```
public static boolean insertOrder(Integer orderid,String deadline_date,String placement_date,Float shipping_cost,Float price,Float total_pay,Integer did,
                              Integer pid,Integer noc) throws SQLException {
    Connection conn = null;
    boolean first = false,second = false,third = false;
    try {
        conn = DbConnection.getConnection();
        conn.setAutoCommit(false);
        try {
            String query = "insert into ORDERS values (?,?,?,?,?,?)";
            PreparedStatement st = conn.prepareStatement(query);
            st.setInt(1, orderid);
            st.setString(2, deadline_date);
            st.setString(3, placement_date);
            st.setFloat(4, shipping_cost);
            st.setFloat(5, price);
            st.setFloat(6, total_pay);
            st.executeUpdate();
            first = true;
        }
        catch(SQLException ex){
            System.out.println("First Insertion Failed");
            first = false;
            return false;
        }
        try {
            String query1 = "insert into PLACES values (?,?)";
            PreparedStatement st1 = conn.prepareStatement(query1);
            st1.setInt(1, orderid);
            st1.setInt(2, did);
            st1.executeUpdate();
            second = true;
        }
        catch(SQLException ex1){
            System.out.println("Second Insertion Failed");
            conn.rollback();
            second = false;
            return false;
        }
        try {
            String query2 = "insert into CONSISTSOF values (?,?,?)";
            PreparedStatement st2 = conn.prepareStatement(query2);
            st2.setInt(1, orderid);
            st2.setInt(2, pid);
            st2.setInt(3, noc);

            st2.executeUpdate();
            ResultSet rs2 = st2.executeQuery("select count(*) as count_val from PLACES where ORDERID=" + orderid);
            int order_id2 = 0;
            while (rs2.next())
                order_id2 = rs2.getInt("count_val");

            if (order_id2 != 1) {
                System.out.println("Third Insertion Failed");
            }
            third = true;
        }
        catch(SQLException ex2){
            System.out.println("Third Insertion Failed");
            conn.rollback();
            return false;
        }
    if(first && second && third){
        conn.commit();
        System.out.println("Transaction Successful!");
    }
    else{
        conn.rollback();
        System.out.println("Transaction Failed");
    }
        conn.setAutoCommit(true);
        return true;
    } catch (SQLException ex) {
        ex.printStackTrace();
        conn.rollback();
        return false;
    }
    finally {
        if(conn!=null) {
            conn.setAutoCommit(true);
        }
```

```
      }
   }
}
```

DESIGN:

The application has a main menu which gives the option to select ADMIN, EDITOR, DISTRIBUTOR.
ADMIN, EDITOR, DISTRIBUTOR represents the different views available. User enters a number 0-4 based on the kind of action he wants to perform.

ADMIN has access to all the tasks and operations which includes CRUD operations for Publications, Distributors, Orders, Staff, Staff Payment and also has access to view all the reports.

EDITOR has access to a subset of operations mentioned above. An editor can view his publication information and add, update or delete Chapters and Articles.

DISTRIBUTOR has access to view his information.

All the menu operations are provided by the UI classes which help in switching between various options which is under the “menu” package.

All CRUD operations are under a separate package called “crud” - which includes create/read/update/delete operations.

All base classes with get/set methods are under the “dbclasses” package.

We have a separate package “dbConnection” to establish connection with the database and shared across all classes.

__FUNCTIONAL ROLES:__

REPORT - 1
Software Engineer (*prime* and *backup*) - Sreeraksha(Prime), Shubham(Backup)
Database Designer / Administrator (*prime* and *backup*) - Sneha(Prime), Sreeraksha(Backup)
Application Programmer (*prime* and *backup*) - Shubham(Prime), Sneha(Backup)
Test Plan Engineer (*prime* and *backup*) - Sneha(Prime), Shubham(Backup)

REPORT - 2
Software Engineer (*prime* and *backup*) - Shubham(Prime), Sneha(backup)
Database Designer / Administrator (*prime* and *backup*) - Sneha(Prime), Sreeraksha(Backup)
Application Programmer (*prime* and *backup*) - Shubham(Prime), SreeRaksha(Backup)
Test Plan Engineer (*prime* and *backup*) - Sreeraksha(Prime), Sneha(Backup)

REPORT - 3
Software Engineer (*prime* and *backup*) - Sneha(Prime), Shubham(Backup)
Database Designer / Administrator (*prime* and *backup*) - Sreeraksha(Prime), Sneha(Backup)
Application Programmer (*prime* and *backup*) - Sreeraksha(Prime), Shubham(Backup)
Test Plan Engineer (*prime* and *backup*) - Shubham(Prime), Sreeraksha(Backup)