

1. Introduction  
    Tadika Mesra School Library Management System is designed to manage the operations of the public library at Tadika Mesra School. This project aims to simulate the actions and processes related to book management, student interaction to library, and library administration of a school. The project uses MySQL Workbench as its main database management and JAVA programming language as the platform to realize the program, including necessary triggers, stored procedures and views for its logical design, indexes for its physical design and an E-R diagram for its conceptual design. The management system serves as an environment for students and library administrators of Tadika Mesra School. It offers basic library functionalities such as book borrowing, renewal, return, and query for students. While for administrators, it offers book addition, edit, delete and student records modification to verify library transactions. Users may interact with the program based on their roles, whether as a student or as an administrator.

The project is done through five main steps, starting from analyzing the general requirement of a school library management system, designing the the data flow diagram and data dictionary, conceptualize the E-R diagram, leading to the logical design where the relational schemes are finalized, then the physical design where index design is included, and finally program creation using MySQL and JAVA.

1. Goals  
   1. Utilize database concepts for a real world scenario project.

2. Understand the general workflow of database designing.

3. Analyze the database requirement for a library management system through a data flow diagram and data dictionary.

4. Create database concept design of a library management system through E-R diagram.

5. Create database logical design through relational schema, triggers, views and stored procedures.

6. Optimize database design to fulfill 3NF.

7. Create database physical design through indexes.

8. Actualize the database into a program using an object oriented programming platform.

1. Environment

MySQL Workbench 8.0 CE  
Java 11

Eclipse 2022.

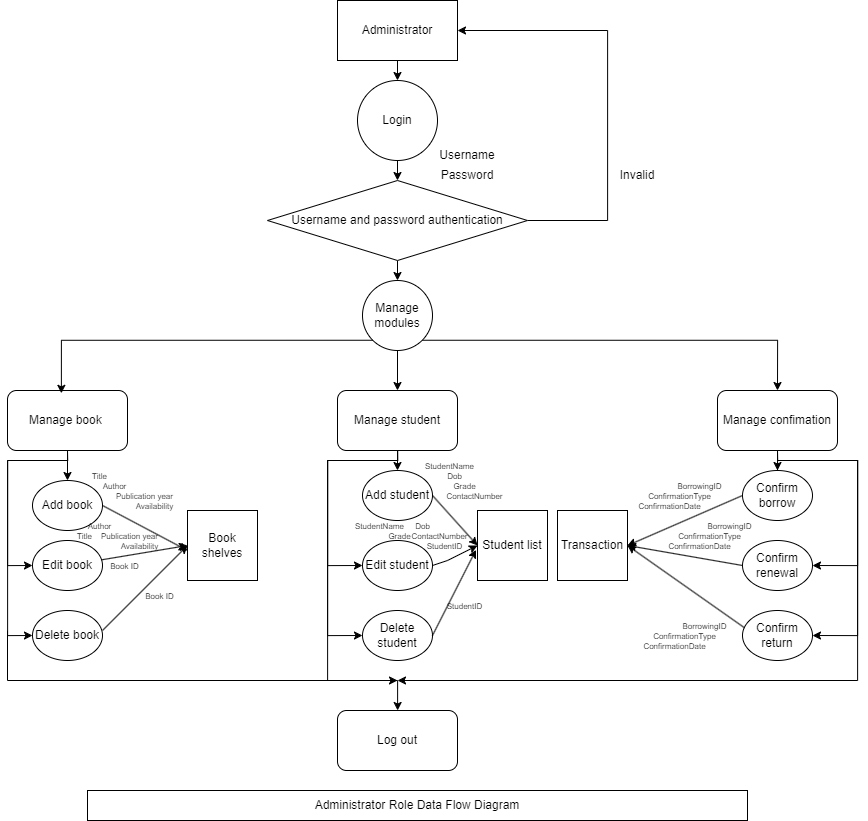
mysql-connector-j-8.0.27

1. Analysis  
   a. Data flow diagram  
   Based on the basic description of the library management system, there are administrators and students that interact with the system. The data flow diagram shows the interactions of both roles with the system including its modules and data exchanges.

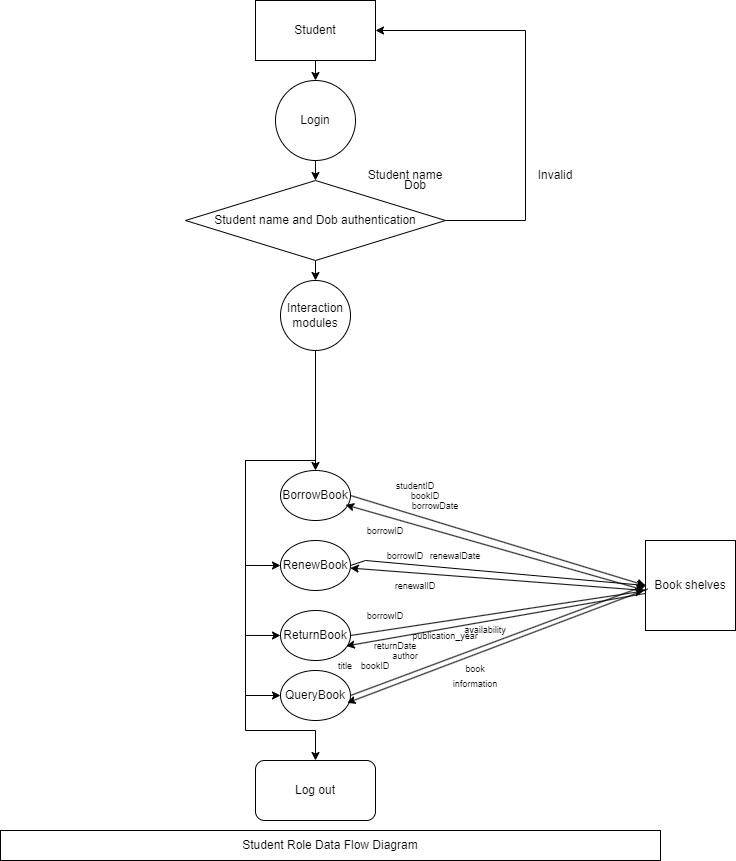
Student : borrow, renew, return, query books.

Administrator : add book, add student, edit book, edit student, delete book, delete student, confirm borrowing, confirm renewing, confirm returning.

Administrator:



Explanation:  
The first data flow diagram is for the administrator oriented role. Its functions are divided into managing books, students and confirmations. Before using any functions, administrators must first login into the system using a unique username and password. Administrators are only able to login if the authentication is valid. In the managing modules, administrators may add books, edit books and delete books. For the student management, administrators may add students, edit students and delete students. For the confirmation management, administrators may confirm borrowing, confirm renewal and confirm returns. Each action in the management needs certain data as illustrated in the data flow diagram. After finishing the necessary tasks, administrators are able to log out from the system.

Student:  


Explanation:  
The second data flow diagram is for the student oriented role. Its functions are divided into book borrowings, renew, return and query. Before using any functions, students must first login into the system using the student name and date of birth. Students are only able to login if the authentication is valid. In the borrowing book, students are able to borrow books by inputting their student ID, book ID, and the given borrowing date, the system then gives out a unique borrowing ID to track the borrowing history. In the renew book, students may renew their current borrowing activity by inputting the borrowID and the given renewal date, the system then gives out a renewal ID to track the renewal history. In the return book, students are returning books they are borrowing or after renewal, the returning date is then noted in the borrowing history. In the book query, students are able to find information about the books in the library by using the available information about the book, then the system gives the full information about the book including its book ID, title, author, publication year and availability. After finishing the interactions, students are able to log out from the system.

b. Data dictionary

The data dictionary explains the meanings behind the database of Tadika Mesra School Library Management System. P means primary key, whereas F means foreign key.

| Book table | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| P/F | Field name | Data type | Data format | Field size | Description | Example |
| P | book\_id | Integer | NNNNN |  | Unique number identification for each book, in this case it is expected to be numerals in the length of 5 digits | 10001 |
|  | title | Varchar |  | 80 | Title of the book, expected to be not empty | Cinderella |
|  | author | Varchar |  | 80 | Author of the book, expected to be not empty | Brothers Grimm |
|  | publication\_year | Integer | YYYY |  | Year of publication, expected to be numerals in the length of 4 digits | 2001 |
|  | availability\_status | Boolean | 1 or 0 |  | The availability status of the book, expected to be either 1 (available) or 0 (not available) | 1 |

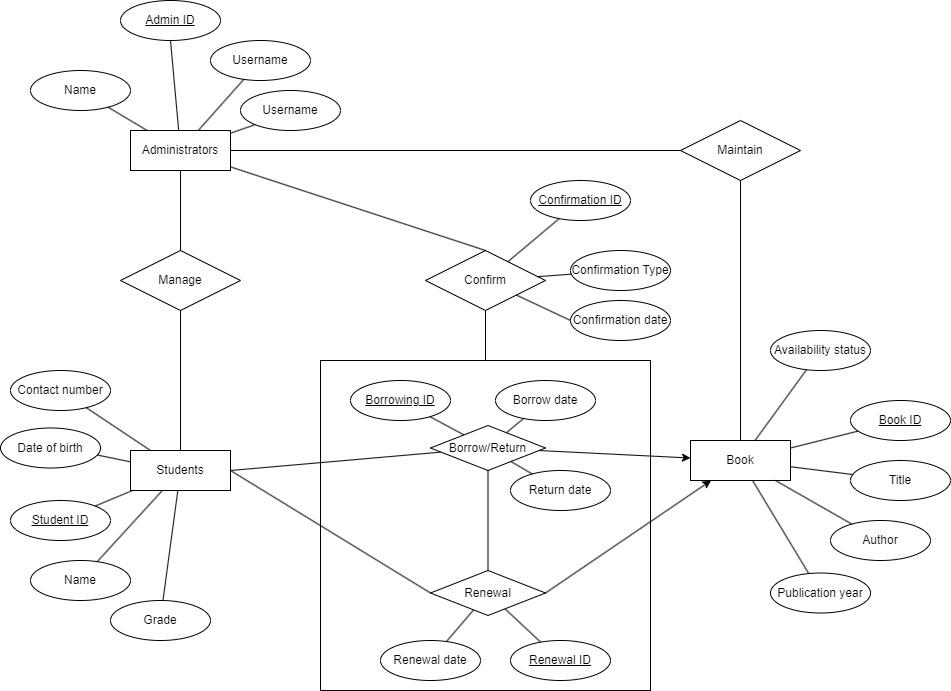
| Students table | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| P/F | Field name | Data type | Data format | Field size | Description | Example |
| P | student\_id | Integer | NNNNN |  | Unique number identification owned by each student, expected to be numerals in the length of 5 digits | 20241 |
|  | student\_name | Varchar |  | 80 | Full name of the student, expected to be not empty | Upin |
|  | grade | Varchar |  | 10 | School grade of the student, expected to be not empty | 10 |
|  | dob | Date | YYYY-MM-DD |  | Date of birth of the student | 2001-01-01 |
|  | contact\_number | Varchar | NNNNNNNN | 15 | Contact number of the student, expected to be numerals in the length of 8 digits | 08111222 |

| Administrators table | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| P/F | Field name | Data type | Data format | Field size | Description | Example |
| P | admin\_id | Integer | NNNNN |  | Unique number identification for each administrator, in this case it is expected to be numerals in the length of 5 digits | 30001 |
|  | admin\_name | Varchar |  | 80 | Full name of the administrator, expected to be not empty | Mawar |
|  | username | Varchar |  | 80 | Username of the administrator, expected to be not empty | mawar\_admin |
|  | password | Varchar |  | 10 | Password of the administrator, expected to be not empty | admin123 |

| Borrowings table | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| P/F | Field name | Data type | Data format | Field size | Description | Example |
| P | borrowing\_id | Integer | NNNNN |  | Unique number identification for each borrowing activity, in this case it is expected to be numerals in the length of 5 digits | 40001 |
| F | student\_id | Integer | NNNNN |  | Unique number identification owned by each student, expected to be numerals in the length of 5 digits | 20241 |
| F | book\_id | Integer | NNNNN |  | Unique number identification for each book, in this case it is expected to be numerals in the length of 5 digits | 10001 |
|  | borrow\_date | Date | YYYY-MM-DD |  | Date on the day of book borrowing | 2022-01-01 |
|  | return\_date | Date | YYYY-MM-DD |  | Date on the day of book returning | 2022-01-15 |

| Renewals table | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| P/F | Field name | Data type | Data format | Field size | Description | Example |
| P | renewal\_id | Integer | NNNNN |  | Unique number identification for each renewal activity, in this case it is expected to be numerals in the length of 5 digits | 50001 |
| F | borrowing\_id | Integer | NNNNN |  | Unique number identification for each borrowing activity, in this case it is expected to be numerals in the length of 5 digits | 40001 |
|  | renewal\_date | Date | YYYY-MM-DD |  | Date on the day of book renewal | 2022-01-10 |

| Confirmations table | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| P/F | Field name | Data type | Data format | Field size | Description | Example |
| P | confirmation\_id | Integer | NNNNN |  | Unique number identification for each confirmation activity, in this case it is expected to be numerals in the length of 5 digits | 60001 |
| F | borrowing\_id | Integer | NNNNN |  | Unique number identification for each borrowing activity, in this case it is expected to be numerals in the length of 5 digits | 40001 |
|  | confirmation\_type | Varchar |  | 20 | Describing the confirmation type, whether it is for borrowing, renewal or returning | Borrowing |
|  | confirmation\_date | Date | YYYY-MM-DD |  | Date on the day of confirmation | 2022-01-01 |

1. Concept design: E-R diagram  
   
2. Logical design  
   a. Relational schema

* books (book\_id, title, author, publication\_year, availability\_status)  
  book\_id is the PRIMARY KEY
* students (student\_id, student\_name, grade, dob, contact\_number )  
  student\_id is the PRIMARY KEY
* administrators (admin\_id, admin\_name, username, password)

admin\_id is the PRIMARY KEY

* borrowings (borrowing\_id, student\_id INT, book\_id INT, borrow\_date DATE, return\_date DATE)

borrowing\_id is the PRIMARY KEY

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (book\_id) REFERENCES books(book\_id)

* renewals (renewal\_id, borrowing\_id, renewal\_date)

renewal\_id is the PRIMARY KEY

FOREIGN KEY (borrowing\_id) REFERENCES borrowings(borrowing\_id)

* confirmations (confirmation\_id, borrowing\_id, confirmation\_type, confirmation\_date)

confirmation\_id is the PRIMARY KEY

FOREIGN KEY (borrowing\_id) REFERENCES borrowings(borrowing\_id)

Based on the relational schema, these are the function dependencies:  
book\_id → title, author, publication\_year, availability\_status

student\_id → student\_name, grade, dob, contact\_number

admin\_id → admin\_name, username, password

borrowing\_id → student\_id, book\_id, borrow\_date, return\_date

renewal\_id → borrowing\_id, renewal\_date

confirmation\_id → borrowing\_id, confirmation\_type, confirmation\_date  
  
From the relational schema and the function dependencies, it can be seen that the database design fulfills the 1NF and 2NF as each tables have its own primary keys and there are no partial dependencies. It fulfills 3NF because there are no transitive dependencies as a transitive dependency occurs when an attribute depends on a non-primary key attribute. After the normalization, the database design is further optimized using view creation, trigger creation, index designing and stored procedures creation which are discussed in the next section.

b. Trigger, view, and stored procedures

* Trigger

| Trigger name | Event | Purpose | Implementation |
| --- | --- | --- | --- |
| update\_availability\_status | Triggered after an insertion on the borrowings table. | Updates the availability\_status of the borrowed book to 0 (not available) in the books table. | Uses the NEW keyword to refer to the newly inserted row in the borrowings table and updates the corresponding book's availability status. |
| update\_return\_date | Triggered before an update on the borrowings table. | Updates the availability\_status of the returned book to 1 (available) in the books table. | Checks if the return\_date in the NEW row is not NULL and updates the availability status of the corresponding book. |
| prevent\_overdue\_borrowing | Triggered before an insertion on the borrowings table. | Prevents the insertion of a borrowing record if the book is overdue (borrow date is earlier than the current date) and has not been returned. | Uses an IF condition to check if the borrow\_date is earlier than the current date (NOW()) and the book has not been returned (return\_date IS NULL). If the condition is met, it signals an error. |
| prevent\_student\_deletion | Triggered before a deletion on the students table. | Prevents the deletion of a student who has active borrowings (books not returned). | Checks if there are any active borrowings for the student (where return\_date is NULL) using an EXISTS subquery. If active borrowings are found, it signals an error. |

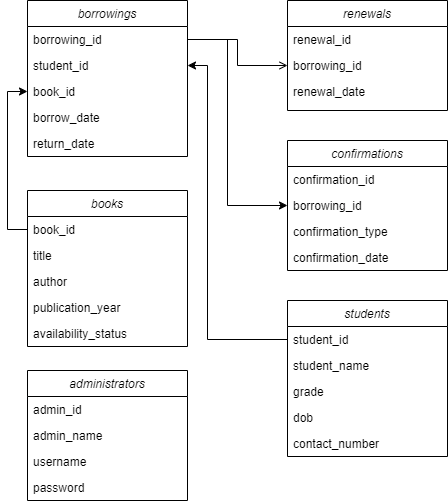
* View

| View name | Purpose |
| --- | --- |
| students\_grade\_12 | This view includes all columns from the students table for students in grade 12. |
| students\_grade\_11 | This view includes all columns from the students table for students in grade 11. |
| students\_grade\_10 | This view includes all columns from the students table for students in grade 10. |
| students\_grade\_9 | This view includes all columns from the students table for students in grade 9. |
| students\_grade\_8 | This view includes all columns from the students table for students in grade 8. |
| students\_grade\_7 | This view includes all columns from the students table for students in grade 7. |
| sd\_curr\_borr | This view includes distinct student records from the students table who are currently borrowing books (where the return date is NULL). |
| sd\_haveborr | This view includes distinct student records from the students table who have at least borrowed one book (regardless of return date). |

* Stored procedures

| Procedure name | Parameters | Purpose |
| --- | --- | --- |
| BorrowBook | -studentId  -bookId  -borrowDate | This procedure is used to borrow a book. It checks if the book is available, and if so, inserts a record into the borrowings table and updates the availability status of the book. |
| RenewBook | -borrowingId  -renewalDate | This procedure is used to renew a book. It checks if the book is renewable (not yet returned), and if so, inserts a record into the renewals table. |
| ReturnBook | -borrowingId  -returnDate | This procedure is used to return a book. It checks if the book is borrowed (not yet returned), and if so, updates the return date in the borrowings table and updates the availability status of the book. |
| QueryBooks |  | This procedure is used to query and retrieve all records from the books table. |
| AddBook | -title  -author  -publicationYear  -availabilityStatus | This procedure is used to add a new book to the books table. |
| AddStudent | -studentName  -grade  -dob  -contactNumber | This procedure is used to add a new student to the students table. |
| EditBook | -bookId  -newTitle  -newAuthor  -newPublicationYear  -newAvailabilityStatus | This procedure is used to edit the information of an existing book in the books table. |
| EditStudent | -studentId  -newStudentName  -newGrade  -newDob  -newContactNumber | This procedure is used to edit the information of an existing student in the students table. |
| DeleteBook | bookId | This procedure is used to delete an existing book from the books table. |
| DeleteStudent | studentId | This procedure is used to delete an existing student from the students table. It includes a trigger to prevent deletion if the student has active borrowings. |
| ConfirmBorrowing | -borrowingId  -confirmationType  -confirmationDate | This procedure is used to confirm a borrowing event by inserting a record into the confirmations table. |
| ConfirmRenewing | -borrowingId  -confirmationType  -confirmationDate | This procedure is used to confirm a renewal event by inserting a record into the confirmations table. |
| ConfirmReturning | -borrowingId  -confirmationType  -confirmationDate | This procedure is used to confirm a returning event by inserting a record into the confirmations table. |

1. Table Design



These tables are interlinked through foreign keys, establishing relationships and enforcing data integrity:

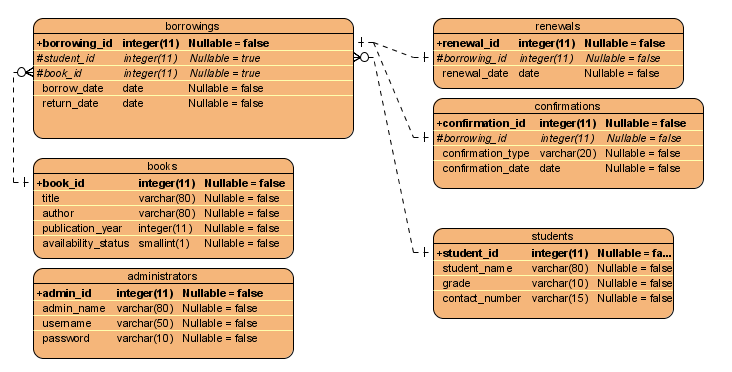
- The `student\_id` in the Borrowings table is a foreign key referencing the `student\_id` in the Students table, ensuring that a borrowing record can only exist for a student present in the Students table.

- The `book\_id` in the Borrowings table is a foreign key referencing the `book\_id` in the Books table, showing that a borrowing record can only exist for a book listed in the Books table.

- The `borrowing\_id` in the Renewals table is a foreign key pointing to the `borrowing\_id` in the Borrowings table, indicating that a renewal record can only exist for a borrowing present in the Borrowings table.

- Similarly, the `borrowing\_id` in the Confirmations table is a foreign key linking to the `borrowing\_id` in the Borrowings table, showing that a confirmation record can only exist for a borrowing in the Borrowings table.

1. Physical design



Relationships between tables:

* The student\_id in the Borrowings table references the student\_id in the Students table, ensuring that a borrowing record can only exist for a student who exists in the Students table.
* Similarly, the book\_id in the Borrowings table references the book\_id in the Books table, ensuring that a borrowing record can only exist for a book that exists in the Books table.
* Each record in the Renewals table is linked to a specific borrowing record in the Borrowings table through the borrowing\_id foreign key.
* Likewise, each confirmation record in the Confirmations table is linked to a specific borrowing record in the Borrowings table through the borrowing\_id foreign key.

1. Encountered Problems
2. Deciding the 3NF fulfillment

After illustrating the data flow diagram and E-R diagram, we tried to analyze the relational schemas and decide whether the 3NF is fulfilled or not. At first, we added another table of returning books. But after analyzing the schemas, we found out it does not fulfill the 3NF and it is not efficient.

1. Deciding the logic of triggers

We made many syntax errors and logic errors when creating the triggers, especially prevent\_overdue\_borrowing. At first the logic looks like:  
DELIMITER //

CREATE TRIGGER prevent\_overdue\_borrowing

BEFORE INSERT ON borrowings

FOR EACH ROW

BEGIN

IF 1 = 1 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot borrow overdue books';

END IF;

END;

//

DELIMITER ;  
It keeps showing true for every borrowing, creating a loop.

1. Data constraints design

In the availability status, we used numerals to show the available copies of the book at first. But after simulating the transactions. We found out that for each copy, we must make a unique book identification, creating a more complex library system, needing more triggers.

1. Connecting Java to MySQL Database

In the project, an outdated version of the MySQL connector was added, resulting in Java programs failing to establish a connection despite the addition of the MySQL connector JAR.

1. Efficiently Determining Foreign Keys

Challenges were encountered in efficiently determining foreign keys for each table. Errors in placing foreign keys according to their functions led to an inefficient and redundant database.

1. Solutions
2. Deciding the 3NF fulfillment

To solve the 3NF problem, we changed the overall design, adding confirmation tables instead of returning tables. After a return action is executed, the returning date will be directly inserted into the confirmation tables.

1. Deciding the logic of triggers

The 1=1 keeps giving true for every input, after detecting the bug. We change the logic to use NOW() to detect the time, resulting in a correct trigger execution.  
IF NEW.borrow\_date < NOW() AND NEW.return\_date IS NULL

1. Data constraints design

Knowing that the library management system with copies availability is complex, we switched the copies into availability status, assuming that each book only has one copy, simplifying the management system and the trigger creations.

1. Connecting Java to MySQL Database

We updated the MySQL connector to ensure the use of the latest version, guaranteeing optimal compatibility between MySQL database and Java.

1. Efficiently Determining Foreign Keys

To solve the problem, it is important to carefully identify relationships between tables, distinguishing between one-to-one, one-to-many, or many-to-many relationships. Through the precise application of these relationships, we obtained efficient tables with minimal redundancy.

1. Conclusion

In conclusion, our project on the Tadika Mesra School Library Management System produces a secure and practical system that meets the needs of both students and administrators. We created a system that combines two main roles, the student and the administrator. It makes it easy for students to do things like borrow, renew, return, and ask questions about books, while giving administrators the tools they need to manage both books and students effectively.

The database design, which is shown in a well-structured relational model, shows that we follow the normalization rules (1NF, 2NF, and 3NF). The idea of the system is well shown in the E-R diagram, which also makes it easy to see how entities relate to each other. Our application goes beyond the theoretical level by using optimization techniques like views, triggers, and stored procedures to make the design of logic better. Adding indexes to the actual design of a database makes sure that it runs quickly.

Using MySQL Workbench and the computer language JAVA, we not only achieve the main goal of the project, which was to simulate how libraries work in the real world, but we also make a platform for learning. Our design process, which includes requirement analysis, data flow diagrams, conceptualization, and physical design, shows how theory ideas can be used in real world application. At the end of our project, we are able to create the Tadika Mesra School Library Management System that provides a comprehensive, user-friendly, and educational tool for efficient library administration.

***Appendix***

1. Configuration and help document

help.txt

1. Source code:

src.zip

1. Database backup file

tm\_library\_administrators.sql

tm\_library\_books.sql

tm\_library\_borrowings.sql

tm\_library\_confirmations.sql

tm\_library\_renewals.sql

tm\_library\_routines.sql

tm\_library\_students.sql

1. Video demonstration

tm\_library JAVA demo.mov

tm\_library MYSQL demo.MP4

1. Installation package

mysql-connector-j-8.0.27.jar