

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

# CAPSTONE PROJECT REPORT

**PROJECT TITLE**

TRANSPORT MANAGEMENT SYSTEM WITH JAVA AND MYSQL

# REPORT SUBMITTED BY

192221111 HARIPRASAD.S

**REPORT SUBMITTED TO**

Dr. S. PADMAKALA

# COURSE CODE / COURSE NAME

CSA0968 / PROGRAMMING IN JAVA WITH COLLECTION INTERFACES

SLOT A

# DATE OF SUBMISSION

22.07.2024

# ABSTRACT

# This Transport Management System (TMS) project is designed to streamline and manage the logistics of transportation services. Developed using Java for application logic and MySQL for the database, the system leverages the Java Database Connectivity (JDBC) with the MySQL Connector for efficient database operations. Key functionalities include inserting, updating, deleting, and displaying transport records, ensuring real-time tracking and management of transportation data. The system's modular design covers various aspects such as user, vehicle, driver, route, and shipment management, making it a comprehensive solution for transportation logistics.

# The core of the TMS lies in its interaction with the MySQL database, managed through a well-defined schema consisting of tables for users, vehicles, drivers, routes, and shipments. The Java application provides a user-friendly interface for data input and retrieval, facilitating CRUD operations (Create, Read, Update, Delete). This integration not only ensures seamless data flow and real-time updates but also enhances operational efficiency, data accuracy, and decision-making capabilities in transportation management.

# INTRODUCTION

# In the fast-paced world of logistics and transportation, efficient management of transport operations is crucial. A Transport Management System (TMS) serves as a backbone for such operations by streamlining processes, enhancing visibility, and optimizing the use of resources. This system, implemented using Java and MySQL, aims to address the critical needs of transportation management by providing a reliable and scalable solution. The use of Java ensures a robust application logic, while MySQL offers a powerful and flexible database system. Together, they create a cohesive environment for managing transport records with ease and efficiency.

# The TMS integrates various modules such as user management, vehicle management, driver management, route optimization, and shipment tracking. Each module plays a vital role in ensuring the smooth operation of transportation services. By leveraging the capabilities of the MySQL Connector, the system facilitates secure and efficient database interactions, allowing for real-time data access and updates. This integration not only improves operational efficiency but also enhances decision-making capabilities through accurate and timely information.

# The development of this TMS is motivated by the need to address common challenges in the transportation industry, such as inefficient resource utilization, lack of real-time tracking, and data fragmentation. By providing a unified platform for managing transport operations, the TMS helps businesses overcome these challenges, leading to improved customer satisfaction, reduced operational costs, and enhanced overall performance.

# LITERATURE REVIEW

# The concept of Transport Management Systems has evolved significantly over the years, driven by advancements in technology and increasing demand for efficient logistics solutions. Early TMS implementations were often limited in scope, focusing primarily on basic tracking and scheduling functionalities. However, with the advent of modern programming languages and database systems, TMS solutions have become more sophisticated, offering comprehensive features that cover the entire spectrum of transport management.

# Several studies have highlighted the benefits of using Java for developing robust and scalable applications. Java's platform-independent nature, object-oriented features, and extensive libraries make it an ideal choice for developing complex systems like TMS. Additionally, Java's compatibility with various database systems through JDBC (Java Database Connectivity) allows for seamless integration with MySQL, a widely-used relational database known for its reliability, performance, and ease of use.

# The use of MySQL in TMS implementations has been extensively documented in the literature. MySQL provides a flexible and efficient database management system that supports complex queries, transactions, and data integrity. Its open-source nature and strong community support have made it a popular choice among developers for building scalable and high-performance applications. Studies have shown that MySQL's capabilities in handling large datasets and its support for various indexing and optimization techniques make it well-suited for TMS applications.

# Furthermore, the integration of TMS with real-time data analytics and reporting tools has been explored in recent research. These integrations enable businesses to gain deeper insights into their transport operations, identify bottlenecks, and make data-driven decisions. The combination of Java, MySQL, and advanced analytics tools has proven to be effective in creating comprehensive TMS solutions that address the dynamic needs of the transportation industry.

# RESEARCH PLAN

The research plan for the Transport Management System encompasses five key phases. The first phase, Project Initiation and Planning, involves defining the project's scope, objectives, and identifying key stakeholders, along with establishing communication channels and developing a comprehensive project plan. In the Requirement Analysis and Design phase, user requirements are thoroughly analyzed, and the database schema and user interface are designed using UML diagrams and wireframes. The Development and Implementation phase focuses on implementing the database in MySQL and developing the application in Java, ensuring scalability and maintainability while integrating core functionalities like student data input, retrieval, and updates. The Testing and Refinement phase includes rigorous testing, such as unit tests, integration tests, and user acceptance testing, to identify and address any bugs and refine the system based on feedback. Finally, the Documentation, Deployment, and Feedback phase involves documenting the development process, preparing for deployment, and collecting user feedback for future enhancements.

| S.NO | DESCRIPTION | 17.07.2024  DAY-01 | 18.07.2024  DAY-02 | 19.07.2024  DAY-03 | 20.07.2024  DAY-04 | 21.07.2024  DAY-05 |
| --- | --- | --- | --- | --- | --- | --- |
| 1. | Project Initiation and Planning |  |  |  |  |  |
| 2. | Requirement Analysis and Design |  |  |  |  |  |
| 3. | Development and Implementation |  |  |  |  |  |
| 4. | Testing and Refinement |  |  |  |  |  |
| 5. | Documentation, Deployment, and Feedback |  |  |  |  |  |

**Fig. 1 Timeline chart**

# Day 1: Project Initiation and Planning (1 day)

# Define the Scope and Objectives: Establish the scope and objectives of the Transport Management System (TMS), focusing on optimizing transportation logistics, tracking vehicles, and managing shipments efficiently.

# Initial Research: Conduct initial research to gather insights into best practices for transport management systems, Java programming for backend development, and user interface design tailored for logistics applications.

# Identify Stakeholders: Identify key stakeholders, including logistics managers, drivers, and IT personnel, and establish effective communication channels.

# Develop Project Plan: Create a comprehensive project plan outlining tasks and milestones for subsequent stages of TMS development.

# Day 2: Requirement Analysis and Design (1 day)

# Requirement Analysis: Conduct a thorough requirement analysis, gathering user needs and essential functionalities for the TMS, including vehicle management, route optimization, and shipment tracking.

# Finalize Design: Finalize the design of the TMS, including database schema design, frontend interface specifications, and backend architecture using MySQL for data storage and Java for application logic.

# Define Requirements: Define software and hardware requirements, ensuring compatibility with Java development environments and MySQL database configurations.

# Day 3: Development and Implementation (2 days)

# Begin Coding: Start coding the TMS according to the finalized design and specifications, focusing on robust data handling, efficient query processing, and user-friendly interface development in Java.

# Implement Core Functionalities: Implement core functionalities such as vehicle data insertion, route management, and shipment status update operations.

# Ensure Integration: Ensure seamless integration between the Java application and MySQL database, adhering to JDBC standards for secure and efficient data access.

# Integrate Frameworks: Integrate necessary frameworks or libraries to enhance functionality and streamline development processes.

# Day 4: Testing and Refinement (1 day)

# Conduct Testing: Perform comprehensive testing of the TMS, including unit tests for individual components, integration tests to ensure seamless operation between modules, and user acceptance testing to validate system functionality.

# Resolve Issues: Identify and resolve any bugs or issues discovered during testing, ensuring the reliability, security, and performance of the TMS.

# Gather Feedback: Collect feedback from stakeholders and end-users to identify usability issues, performance bottlenecks, or additional features for refinement.

# Make Adjustments: Make necessary adjustments to the TMS based on feedback and testing results, aiming for a stable and user-friendly application.

# Day 5: Documentation, Deployment, and Feedback (1 day)

# Document Development: Document the development process comprehensively, including architectural decisions, implementation details, and testing methodologies used during TMS development.

# Prepare for Deployment: Prepare the TMS for deployment, ensuring proper configuration of the MySQL database and Java application server.

# Deploy to Testing Environment: Deploy the TMS to a testing environment for final validation, user acceptance, and quality assurance.

# Gather Final Feedback: Collect final feedback from stakeholders and end-users to evaluate the effectiveness of the TMS in meeting logistics management needs and addressing initial project objectives...

# METHODOLOGY

# Step 1: Install My Sql 8.0.37 Installer from the official website of oracle <https://dev.mysql.com/downloads/installer/> by selecting the windows operating system

# Step2:Install My Sql connector for java form the official website <https://dev.mysql.com/downloads/connector/j/> by selecting the option called Platform independent . This runs on all platforms including Windows.

# Step 3: Install Mysql by following the on-screen instructions and provide a strong password.

# Step 3: Extract the My mysql-connector-j-8.4.0 in the usual java programs folder where all the java program will be saved and delete the rar folder

# Step 4: Now type the java program with JDBC connectivity credentials and necessary CRUD functions like Insert, Update, Select and Delete and save it.

# Step 5: After installation open the Mysql command line client, enter the password and create a database by specifying the parameters.

# Step 6: open cmd and compile the program using compilation code.

# Step 7: Verify the changes in mysql command line client by typing “SELECT \* FROM Student;

**SQL CODE**

CREATE TABLE TransportManagement (

TransportID INT AUTO\_INCREMENT PRIMARY KEY,

PassengerID INT,

TransportType VARCHAR(255),

PickupLocation VARCHAR(255),

DropoffLocation VARCHAR(255),

ScheduledTime DATETIME,

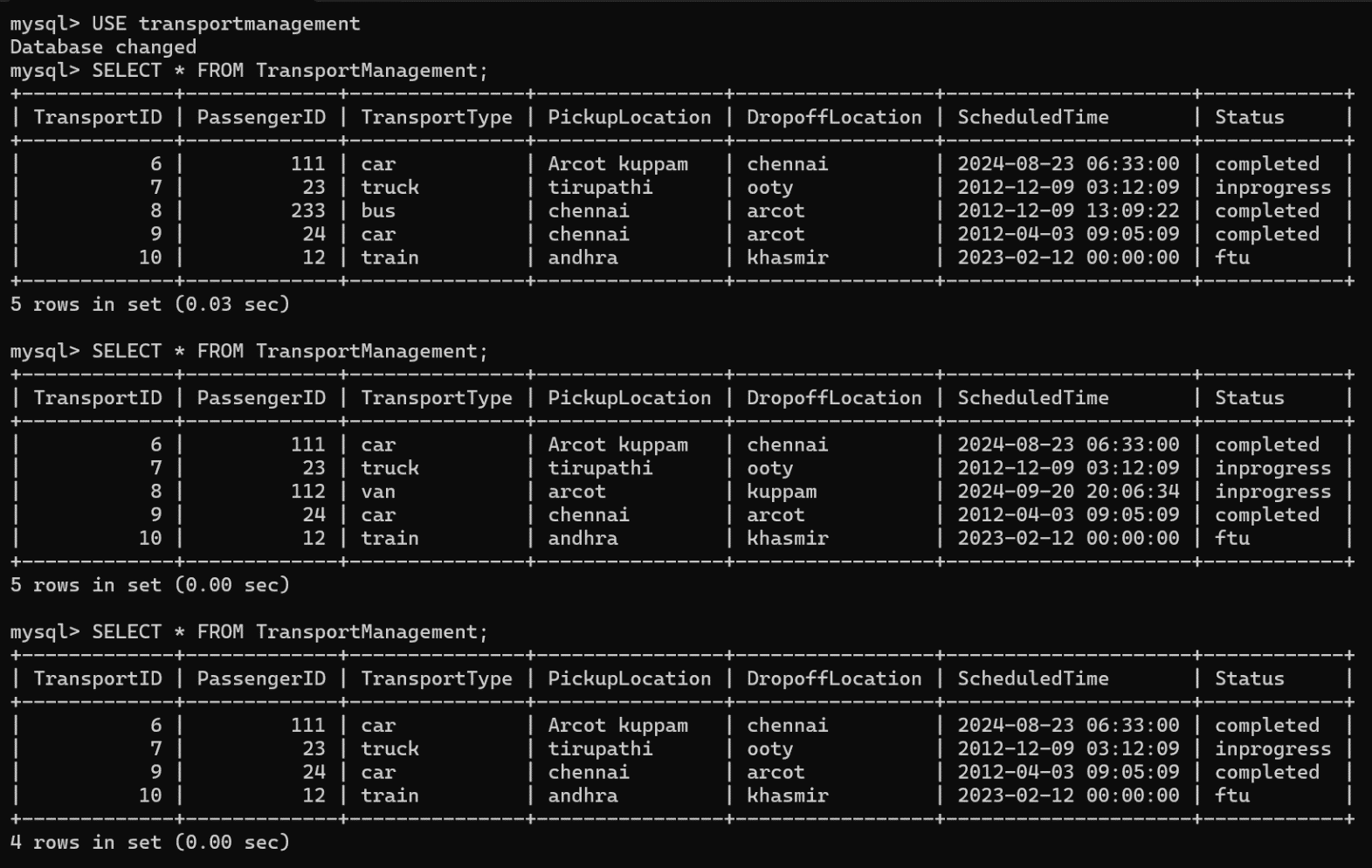
Status VARCHAR(255)

);

# COMPILATION CODE:

# mysql >USE transportmanagement

# mysql >SELECT \* FROM TransportManagement;



# JAVA CODE

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Scanner;

public class Transport {

private static final String URL = "jdbc:mysql://localhost:3306/TransportManagement";

private static final String USERNAME = "root";

private static final String PASSWORD = "H@ri";

public static void main(String[] args) {

try {

// Load MySQL JDBC Driver

Class.forName("com.mysql.cj.jdbc.Driver");

// Initialize scanner for user input

Scanner scanner = new Scanner(System.in);

boolean exit = false;

while (!exit) {

System.out.println("Choose an option:");

System.out.println("1. Insert Transport Record");

System.out.println("2. Update Transport Record");

System.out.println("3. Delete Transport Record");

System.out.println("4. Display All Transport Records");

System.out.println("5. Exit");

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

// Insert transport record

System.out.println("Enter passenger ID:");

int passengerID = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.println("Enter transport type:");

String transportType = scanner.nextLine();

System.out.println("Enter pickup location:");

String pickupLocation = scanner.nextLine();

System.out.println("Enter dropoff location:");

String dropoffLocation = scanner.nextLine();

System.out.println("Enter scheduled time (YYYY-MM-DD HH:MM:SS):");

String scheduledTime = scanner.nextLine();

System.out.println("Enter status:");

String status = scanner.nextLine();

insertTransport(passengerID, transportType, pickupLocation, dropoffLocation, scheduledTime, status);

break;

case 2:

// Update transport record

System.out.println("Enter transport ID to update:");

int updateId = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.println("Enter new passenger ID:");

int newPassengerID = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.println("Enter new transport type:");

String newTransportType = scanner.nextLine();

System.out.println("Enter new pickup location:");

String newPickupLocation = scanner.nextLine();

System.out.println("Enter new dropoff location:");

String newDropoffLocation = scanner.nextLine();

System.out.println("Enter new scheduled time (YYYY-MM-DD HH:MM:SS):");

String newScheduledTime = scanner.nextLine();

System.out.println("Enter new status:");

String newStatus = scanner.nextLine();

updateTransport(updateId, newPassengerID, newTransportType, newPickupLocation, newDropoffLocation, newScheduledTime, newStatus);

break;

case 3:

// Delete transport record

System.out.println("Enter transport ID to delete:");

int deleteId = scanner.nextInt();

deleteTransport(deleteId);

break;

case 4:

// Display all transport records

selectTransportRecords();

break;

case 5:

// Exit

exit = true;

break;

default:

System.out.println("Invalid option. Please try again.");

break;

}

}

scanner.close();

} catch (ClassNotFoundException e) {

System.out.println("MySQL JDBC Driver not found.");

e.printStackTrace();

}

}

// Insert a new transport record

public static void insertTransport(int passengerID, String transportType, String pickupLocation, String dropoffLocation, String scheduledTime, String status) {

String query = "INSERT INTO TransportManagement (PassengerID, TransportType, PickupLocation, DropoffLocation, ScheduledTime, Status) VALUES (?, ?, ?, ?, ?, ?)";

try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setInt(1, passengerID);

preparedStatement.setString(2, transportType);

preparedStatement.setString(3, pickupLocation);

preparedStatement.setString(4, dropoffLocation);

preparedStatement.setString(5, scheduledTime);

preparedStatement.setString(6, status);

preparedStatement.executeUpdate();

System.out.println("Transport record inserted successfully.");

} catch (SQLException e) {

System.out.println("Insert operation failed!");

e.printStackTrace();

}

}

// Select and display all transport records

public static void selectTransportRecords() {

String query = "SELECT \* FROM TransportManagement";

try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query)) {

System.out.println("Current Transport Records:");

while (resultSet.next()) {

int transportID = resultSet.getInt("TransportID");

int passengerID = resultSet.getInt("PassengerID");

String transportType = resultSet.getString("TransportType");

String pickupLocation = resultSet.getString("PickupLocation");

String dropoffLocation = resultSet.getString("DropoffLocation");

String scheduledTime = resultSet.getString("ScheduledTime");

String status = resultSet.getString("Status");

System.out.println("TransportID: " + transportID + ", PassengerID: " + passengerID + ", Transport Type: " + transportType + ", Pickup Location: " + pickupLocation + ", Dropoff Location: " + dropoffLocation + ", Scheduled Time: " + scheduledTime + ", Status: " + status);

}

} catch (SQLException e) {

System.out.println("Select operation failed!");

e.printStackTrace();

}

}

// Update a transport record

public static void updateTransport(int transportID, int passengerID, String transportType, String pickupLocation, String dropoffLocation, String scheduledTime, String status) {

String query = "UPDATE TransportManagement SET PassengerID = ?, TransportType = ?, PickupLocation = ?, DropoffLocation = ?, ScheduledTime = ?, Status = ? WHERE TransportID = ?";

try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setInt(1, passengerID);

preparedStatement.setString(2, transportType);

preparedStatement.setString(3, pickupLocation);

preparedStatement.setString(4, dropoffLocation);

preparedStatement.setString(5, scheduledTime);

preparedStatement.setString(6, status);

preparedStatement.setInt(7, transportID);

preparedStatement.executeUpdate();

System.out.println("Transport record updated successfully.");

} catch (SQLException e) {

System.out.println("Update operation failed!");

e.printStackTrace();

}

}

// Delete a transport record

public static void deleteTransport(int transportID) {

String query = "DELETE FROM TransportManagement WHERE TransportID = ?";

try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

PreparedStatement preparedStatement = connection.prepareStatement(query)) {

preparedStatement.setInt(1, transportID);

preparedStatement.executeUpdate();

System.out.println("Transport record deleted successfully.");

} catch (SQLException e) {

System.out.println("Delete operation failed!");

e.printStackTrace();

}

}

}

# COMPILATION CODE

# C:\javas>javac -cp .;c:\javas\mysql-connector-j-8.4.0\mysql-connector-j-8.4.0.jar Transport.java

# C:\javas>java -cp .;c:\javas\mysql-connector-j-8.4.0\mysql-connector-j-8.4.0.jar Transport

# 

# CONCLUSION

# This Transport Management System (TMS) implemented in Java with MySQL as the database provides a comprehensive and efficient way to manage transport records. The project covers the essential CRUD (Create, Read, Update, Delete) operations, allowing for the insertion, updating, deletion, and selection of transport records in a structured and user-friendly manner.

# Key aspects of the system include:

# 1.User Interaction: The console-based user interface allows users to choose different operations such as inserting, updating, deleting, and displaying transport records. This interaction is facilitated through a menu-driven approach, making it intuitive and straightforward to use.

# 2.Database Connectivity: By leveraging JDBC with MySQL Connector, the system establishes a reliable connection to the MySQL database. This connection is essential for performing various database operations efficiently.

# 3.Modular Design: The system's functionality is encapsulated within different methods (e.g., `insertTransport`, `updateTransport`, `deleteTransport`, `selectTransportRecords`), promoting code reuse and maintainability.

# 4.Data Integrity: With the `TransportManagement` table schema, the system ensures that each transport record is uniquely identified by a `TransportID`. The schema also defines the necessary fields such as `PassengerID`, `TransportType`, `PickupLocation`, `DropoffLocation`, `ScheduledTime`, and `Status`, ensuring data consistency and integrity.

# 5.Exception Handling: The system incorporates exception handling mechanisms to manage potential SQL exceptions, enhancing the robustness and reliability of the application.

# Future Enhancements

# 1. User Authentication: Implementing user authentication and role-based access control to enhance security.

# 2.Graphical User Interface (GUI): Developing a GUI using JavaFX or a web-based interface using JSP/Servlets for a more user-friendly experience.

# 3.Advanced Reporting: Adding advanced reporting and analytics features to provide deeper insights into transport operations.

# 4.Integration with GPS: Integrating GPS functionality for real-time tracking of vehicles and shipments.

# REFERENCES

1.Transport Management Systems (TMS): A Complete Guide" by F. Glaves, Logistics Management. [Logistics Management](<https://www.logisticsmgmt.com>)

2. Java Persistence with Hibernate" by Christian Bauer and Gavin King, Manning Publications. [Manning Publications](<https://www.manning.com/books/java-persistence-with-hibernate>)

3. "Pro Java Persistence with Hibernate"by Christian Bauer and Gavin King, Apress. [Apress](<https://www.apress.com/book/9781430224240>)

4. "MySQL 8.0 Reference Manual" by Oracle Corporation, Oracle. [MySQL Reference Manual](<https://dev.mysql.com/doc/refman/8.0/en/>)

5. "Java: The Complete Reference" by Herbert Schildt, McGraw-Hill Education. [McGraw-Hill Education](<https://www.mhprofessional.com/>)

6. "Head First Java" by Kathy Sierra and Bert Bates, O'Reilly Media. [O'Reilly Media](<https://www.oreilly.com/library/view/head-first-java/0596009208/>)

7. "Java Database Connectivity (JDBC) Tutorial" by Oracle Corporation, Oracle. [JDBC Tutorial](<https://docs.oracle.com/javase/tutorial/jdbc/>)

8. "Designing Data-Intensive Applications" by Martin Kleppmann, O'Reilly Media. [O'Reilly Media](<https://www.oreilly.com/library/view/designing-data-intensive/9781491903895/>)

9. "Spring in Action" by Craig Walls, Manning Publications. [Manning Publications](<https://www.manning.com/books/spring-in-action-fifth-edition>)

10. "Database System Concepts" by Abraham Silberschatz, Henry Korth, and S. Sudarshan, McGraw-Hill Education. [McGraw-Hill Education](<https://www.mhprofessional.com/>)