

**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

B.Tech. - CSE- Cyber Security

Index

S.No	Regd. No.	Page No.
1	221FA19012	3
2	221FA19015	
3	221FA19062	
4	221FA19065	
5	221FA19025	21
6	221FA19029	
7	221FA19048	
8	221FA19061	
9	221FA19005	44
10	221FA19045	
11	221FA19051	
12	221FA19052	
13	221FA19024	66
14	221FA19031	
15	221FA19033	
16	221FA19054	
17	221FA19007	77
18	221FA19047	
19	221FA19049	
20	231LA19001	
21	221FA19003	91
22	221FA19011	
23	221FA19018	
24	221FA19060	
25	221FA19017	109
26	221FA19043	
27	221FA19058	
28	221FA19067	

29	221FA19002	130
30	221FA19027	
31	221FA19028	
32	221FA19046	
33	221FA19008	143
34	221FA19010	
35	221FA19032	
36	221FA19059	
37	221FA19016	160
38	221FA19037	
39	221FA19053	
40	221FA19063	
41	221FA19004	175
42	221FA19040	
43	221FA19044	
44	221FA19057	
45	221FA19006	186
46	221FA19013	
47	221FA19022	
48	221FA19026	
49	221FA19001	201
50	221FA19009	
51	221FA19019	
52	221FA19056	
53	221FA19014	218
54	221FA19021	
55	221FA19039	
56	221FA19064	
57	221FA19020	230
58	221FA19030	
59	221FA19038	
60	221FA19055	

A FIELD PROJECT/IDP REPORT ON

ER - Diagram

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

DAMARACHARLA GEETHA HARIKA (221FA19012)

PALGULLA RANGASWAMI REDDY (221FA19015)

SYED SHANIKA ZAIDA (221FA19062)

YADAVALLI MEENAKSHI (221FA19065)



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ACSE

School of Computing and Informatics

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH
(Deemed to be University)**

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April, 2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled "ER – Diagram" being submitted by **221FA19012-DAMARACHARLA GEETHA HARIKA, 221FA19015- PALGULLA RANGASWAMI REDDY, 221FA19062- SYED SHANIKA ZAIDA, 221FA19065-YADAVALLI MEENAKSHI** in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignan's Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Guide :

HOD/ACSE

Dr. Venkatesulu Dondeti



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that our project work described in the field project titled “ER Diagram” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

DAMARACHARLA GEETHA HARIKA	221FA19012
PALGULLA RANGASWAMI REDDY	221FA19015
SYED SHANIKA ZAIDA	221FA19062
YADAVALLI MEENAKSHI	221FA19065

Abstract

This project focuses on designing and implementing a University Management System to manage the data related to courses, students, instructors, and enrollments. The system is built using a relational database model with ER diagramming techniques to map relationships between entities such as courses, students, and instructors. The project includes designing the entity-relationship (ER) diagram, converting it into a relational model, and implementing the database using SQL queries. Additionally, the report provides an in-depth analysis of the results obtained, detailing how the system efficiently manages university data. The project aims to enhance the understanding of database management principles and the implementation of relational databases in real-world scenarios.

Contents

Name of the Content

1. Introduction
2. Database Design and Implementation
 - 2.1 Software and Hardware Requirements
3. Entity-Relationship (ER) Model
 - 3.1 Entities and Attributes
 - 3.2 Relationships
4. Relational Model
 - 4.1 Tables and Constraints
5. ER Diagram
6. Query Implementation
7. Result Analysis
 - 7.1 Data Integrity
 - 7.2 Query Performance
 - 7.3 Scalability and Future Considerations
8. Conclusion
9. References

1.Introduction

In the contemporary digital landscape, effective data management is paramount for the success of any e-commerce platform. The intricate relationships between users, accounts, products, and categories necessitate a robust database system that ensures data integrity, scalability, and efficient retrieval. This report delves into the comprehensive design and implementation of a database system tailored for an e-commerce marketplace, encompassing user profiles, account management, product categorization, and shop certifications. By meticulously converting the provided schemas into an Entity-Relationship (ER) diagram and subsequently into a relational model, this project aims to establish a foundational framework that supports seamless operations, user interactions, and product management within the marketplace.

The primary objective is to create an ER diagram that visually represents the entities and their interrelations, facilitating a clear understanding of the data flow and dependencies. This diagram serves as a blueprint for the relational database, ensuring that the system is both efficient and scalable. Furthermore, the report outlines the necessary software and hardware requirements, detailing the tools and technologies employed during the design and implementation phases. Through structured query implementation and thorough result analysis, the project assesses the system's performance, data integrity, and potential for future enhancements.

In this project, we develop a database system for a University Management System. The database stores information about courses, course offerings, students, instructors, and enrollment data. The main objective is to efficiently manage and organize the university's academic records using a relational database approach. The system allows for querying and updating of information, providing a comprehensive solution for managing university data.

2.Database Design and Implementation

The database design employs the Entity-Relationship (ER) model, visually representing entities and their relationships. The relational schema is developed to structure the database tables, establishing primary and foreign key constraints to maintain data integrity.

2.1 Software and Hardware Requirements:

Software Requirements:

- **Database Management System:** MySQL or PostgreSQL
- **Development Environment:** SQL Workbench or any SQL IDE (e.g., DBeaver)
- **Operating System:** Windows 10 or later, Linux (Ubuntu), or Mac OS X

Hardware Requirements:

- **Processor:** Minimum Intel i5 or equivalent (quad-core recommended)
- **RAM:** Minimum 8 GB (16 GB preferred for better performance)
- **Storage:** Minimum 50 GB of free space (SSD recommended for speed)
- **Network:** Reliable internet connection for software installation and updates

3.Entity-Relationship (ER) Model

The ER model for the University Management System includes entities such as Courses, Students, Instructors, and Enrollments. Each entity has a set of attributes that define its properties. Relationships between entities are mapped with cardinality constraints, ensuring accurate representation of university processes. For instance, students enroll in courses, and instructors teach courses. The cardinalities are defined to accurately represent these relationships and their constraints.

3.1Entities and Attributes:

1.Employee:

- Attribute:
 - EmployeeNumber : Primary Key(PK),uniquely identifies each employee(integer)
 - FirstName: Stores the first name of the employee (String)
 - LastName: Stores the last name of the employee (String)
 - ZipCode: Stores the postal code for the employee's address (String)

2. Customer:

- Attributes:
 - CustomerNumber: Primary Key (PK), uniquely identifies each customer (Integer)
 - FirstName: Stores the first name of the customer (String)
 - LastName: Stores the last name of the customer (String)
 - ZipCode: Stores the postal code for the customer's address (String)

3. Part:

- Attributes:
 - PartNumber: Primary Key (PK), uniquely identifies each part (Integer)
 - PartName: Stores the name of the part (String)
 - Price: Stores the price of the part (Decimal)

4. Order:

- Attributes:
 - OrderNumber: Primary Key (PK), uniquely identifies each order (Integer)
 - OrderDate: Stores the date when the order was placed (Date)
 - CustomerNumber: Foreign Key (FK), references Customer (Integer)
 - EmployeeNumber: Foreign Key (FK), references Employee (Integer)

5. OrderLine:

- Attributes:
 - OrderLineNumber: Primary Key (PK), uniquely identifies each order line (Integer)
 - OrderNumber: Foreign Key (FK), references Order (Integer)
 - PartNumber: Foreign Key (FK), references Part (Integer)
 - Quantity: Stores the quantity of parts in the order line (Integer)
 - Price: Stores the price of the part at the time of order (Decimal)

3.2 Relationships:

1. Employee-Manages-Orders:

- Description: An employee manages orders.
- Entities Involved: Employee, Order
- Relationship Type: One-to-Many (1) - One employee can manage multiple orders, but each order is managed by one employee.

2. Customer-Places-Orders:

- Description: A customer places orders.
- Entities Involved: Customer, Order
- Relationship Type: One-to-Many (1) - One customer can place multiple orders, but each order belongs to one customer.

3. Order-Contains-OrderLines:

- Description: An order can contain multiple order lines.
- Entities Involved: Order, OrderLine
- Relationship Type: One-to-Many (1) - One order can have multiple order lines, but each order line is associated with one order.

4. OrderLine-Includes-Part:

- Description: An order line includes a part.
- Entities Involved: OrderLine, Part
- Relationship Type: Many-to-One (N:1) - Multiple order lines can include the same part, but each order line includes only one part.

4. Relational Model

The relational model is derived from the ER diagram, where each entity and relationship is represented as a table. Primary keys and foreign keys are established to ensure referential integrity. Tables such as Courses, Course Offerings, Students, and Instructors have been created to store and manage data effectively. The relational model supports various queries to fetch and update data.

4.1 Tables and Constraints:

1. Employee Table:

- **Columns:**
 - **EmployeeNumber:** Primary Key (Integer)
 - **FirstName:** String
 - **LastName:** String
 - **ZipCode:** String

2. Customer Table:

- **Columns:**
 - **CustomerNumber:** Primary Key (Integer)
 - **FirstName:** String
 - **LastName:** String
 - **ZipCode:** String

3. Part Table:

- **Columns:**
 - **PartNumber:** Primary Key (Integer)
 - **PartName:** String
 - **Price:** Decimal

4. Order Table:

- **Columns:**
 - **OrderNumber:** Primary Key (Integer)
 - **OrderDate:** Date
 - **CustomerNumber:** Foreign Key (references CustomerNumber in Customer table)

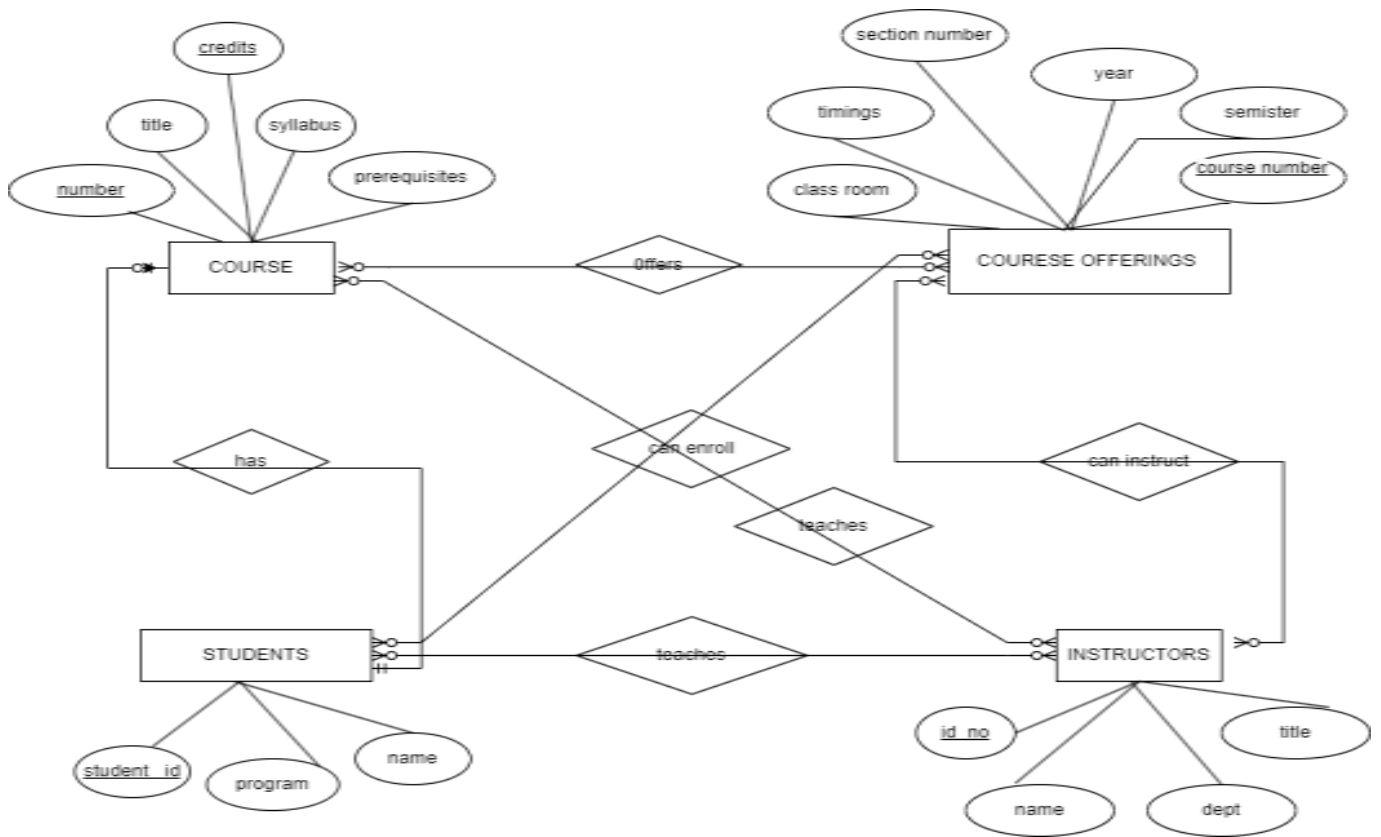
- **EmployeeNumber:** Foreign Key (references EmployeeNumber in Employee table)

5. OrderLine Table:

- **Columns:**

- **OrderLineNumber:** Primary Key (Integer)
- **OrderNumber:** Foreign Key (references OrderNumber in Order table)
- **PartNumber:** Foreign Key (references PartNumber in Part table)
- **Quantity:** Integer
- **Price:** Decimal

5.ER Diagram



6. Query Implementation

To manage the university's database efficiently, several SQL queries were developed. These queries handle tasks such as retrieving student enrollment information, fetching course details, and managing instructor assignments. Each query ensures data consistency and allows for efficient data manipulation and retrieval.

Examples of queries implemented:

1. Retrieve all courses offered in a specific semester:

```
SELECT course_id, title, credits FROM Courses WHERE semester = 'Fall';
```

This query retrieves a list of courses offered in the specified semester, including their IDs, titles, and credit information.

2. Find all students enrolled in a particular course:

```
SELECT Students.student_id, Students.name FROM Students  
JOIN Enrollments ON Students.student_id = Enrollments.student_id  
WHERE Enrollments.course_id = 'CS101';
```

This query lists all students enrolled in the course with ID 'CS101', showing each student's ID and name.

3. List instructors and the courses they teach:

```
SELECT Instructors.name, Courses.title FROM Instructors  
JOIN Course Offerings ON Instructors.instructor_id = Course Offerings.instructor_id  
JOIN Courses ON Course Offerings.course_id = Courses.course_id;
```

This query returns a list of instructors along with the courses they teach, joining relevant tables to provide comprehensive information.

7.Result Analysis

7.1 Data Integrity

The database structure ensures data integrity by establishing primary keys for each table, making each record unique and preventing duplication. Foreign key constraints enforce referential integrity between related tables, ensuring that relationships among entities are maintained accurately. For example, each enrollment is correctly linked to a student and a course, validating the integrity of these relationships. The design also includes checks for constraints such as ensuring that the same course cannot be assigned to the same student more than once for a given semester, further reinforcing data integrity.

7.2 Query Performance

The database demonstrates efficient query performance, with SQL queries optimized to handle data retrieval and manipulation promptly. For instance, when querying the enrollment data for a specific course, the results are returned swiftly, showcasing the effectiveness of the indexing on fields like student IDs and course IDs. The relational structure of the database supports efficient data retrieval by allowing queries to filter and join data based on these indexed fields, ensuring that operations remain efficient even as the dataset grows.

7.3 Scalability and Future Considerations

The database is designed with scalability and future expansion in mind, allowing for the easy addition of new entities and attributes, such as new courses or student records, without disrupting the existing structure. Future enhancements could involve integrating additional functionalities like tracking student performance over time, incorporating online registration features, or expanding the course catalog dynamically. These improvements would not only expand the capabilities of the system but also streamline academic and administrative operations. The successful implementation of the University Management System database demonstrates the robustness of relational databases in managing complex relationships and ensuring the smooth operation of university processes.

8.Conclusion

The University Management System project demonstrates the importance of database management principles in organizing and handling large sets of data. By designing a comprehensive ER model and implementing it through a relational database, the project provides a reliable solution for managing university records. The use of SQL queries further showcases how relational databases can efficiently manage and retrieve data, offering valuable insights into database design and implementation.

9.References

1. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*. Pearson.
2. Date, C. J. (2004). *An Introduction to Database Systems*. Addison-Wesley.
3. Connolly, T. M., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson.
4. Rob, P., & Coronel, C. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.
5. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
6. Korth, H. F., & Silberschatz, A. (2002). *Database System Concepts*. McGraw-Hill.
7. O'Neil, P., & O'Neil, E. (2009). *Database: Principles, Programming, and Performance*. Morgan Kaufmann.
8. Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, 13(6), 377-387.
9. Chen, P. P. (1976). "The Entity-Relationship Model—Toward a Unified View of Data." *ACM Transactions on Database Systems*, 1(1), 9-36.
10. Firebird Project. (n.d.). "Firebird SQL." Retrieved from <https://firebirdsql.org/>
11. MySQL. (n.d.). "MySQL Documentation." Retrieved from <https://dev.mysql.com/doc/>
12. PostgreSQL. (n.d.). "PostgreSQL Documentation." Retrieved from <https://www.postgresql.org/docs/>
13. Ambler, S. W. (2003). *The Object Primer: Agile Model-Driven Development*. Cambridge University Press.
14. Oracle Corporation. (n.d.). "Oracle Database Documentation." Retrieved from <https://docs.oracle.com/en/database/>
15. IBM. (n.d.). "IBM Db2 Documentation." Retrieved from <https://www.ibm.com/docs/en/db2>
16. Microsoft. (n.d.). "SQL Server Documentation." Retrieved from <https://docs.microsoft.com/en-us/sql/sql-server/>
17. Beighley, L., & Rhyne, M. (2008). *Head First SQL: Your Brain on SQL -- A Learner's Guide*. O'Reilly Media.
18. Hoffer, J. A., Venkataraman, R., & Topi, H. (2013). *Modern Database Management*. Pearson.

19. Grelck, C., & Mertens, H. (2016). "An Object-Relational Database Management System with a Comprehensive Query Language." *Concurrency and Computation: Practice and Experience*, 28(2), 456-479.
20. Hellerstein, J. M., Stonebraker, M., & Hamilton, J. (2007). "Architecture of a Database System." *Foundations and Trends in Databases*, 1(2), 141-259.

A FIELD PROJECT/IDP REPORT ON

Entity Relationship Diagram

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

YANDRAPU MAHESH (221FA19025)

BALA PRASANTH (221FA19029)

KAMINENI LEELA TAPASWI (221FA19048)

MIRIYALA NAGA SAI (221FA19061)



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ACSE

School of Computing and Informatics

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH
(Deemed to be University)**

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April, 2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

Established under Section 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled "Entity Relationship Diagram" being submitted by **YANDRAPU MAHESH-221FA19025, BALA PRASANTH-221FA19029, KAMINENI LEELA TAPASWI-221FA19048, MIRIYALA NAGA SAI-221FA19061** in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Guide

HOD/ACSE

Dr.Venkatesulu Dondeti



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that our project work described in the field project titled Entity Relationship Diagram”which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

YANDRAPU MAHESH	221FA19025
BALA PRASANTH	221FA19029
KAMINENI LEELA TAPASWI	221FA19048
MIRIYALA NAGA SAI	221FA19061

ABSTRACT

This project outlines the design and implementation of a relational database for an e-commerce platform to manage user profiles, accounts (seller and buyer), shops, certified shops, and product classifications. The primary goal is to develop a structured and scalable database that efficiently handles the complexities of managing user data, shop details, and product classifications. Through the use of an Entity-Relationship (ER) model, the project identifies key entities, their attributes, and relationships, which are then translated into a relational schema. This design ensures efficient data management, query performance, and scalability for future growth.

The introduction discusses the growing complexity of managing data in e-commerce platforms. With the increasing importance of digital platforms for businesses, a well-organized database is essential to streamline operations and support growth. Managing user information, product details, and shop-related data becomes more challenging as the business scales, making a robust database design critical for operational efficiency.

Query implementation is another key aspect of this project. The database is designed to support efficient queries, such as retrieving user details, product lists, or certified shop information. Indexing and other optimization techniques are considered to ensure high query performance. Scalability is built into the system to allow for future growth, making the database adaptable to expanding data volumes without sacrificing performance.

TABLE OF CONTENTS

Content

1. Introduction
2. Database Design and Implementation
 - 2.1 Software and Hardware Requirements
3. Entity-Relationship (ER) Model .
 - 3.1 Entities and Attributes
 - 3.2 Relationships
4. Relational Model
 - 4.1 Tables and Constraints
5. ER Diagram
6. Query Implementation
7. Result Analysis
 - 7.1 Data Integrity
 - 7.2 Query Performance
 - 7.3 Scalability and Future Considerations
8. Conclusion
9. References

INTRODUCTION

In the contemporary digital landscape, effective data management is paramount for the success of any e-commerce platform. The intricate relationships between users, accounts, products, and categories necessitate a robust database system that ensures data integrity, scalability, and efficient retrieval. This report delves into the comprehensive design and implementation of a database system tailored for an e-commerce marketplace, encompassing user profiles, account management, product categorization, and shop certifications. By meticulously converting the provided schemas into an Entity-Relationship (ER) diagram and subsequently into a relational model, this project aims to establish a foundational framework that supports seamless operations, user interactions, and product management within the marketplace.

The primary objective is to create an ER diagram that visually represents the entities and their interrelations, facilitating a clear understanding of the data flow and dependencies. This diagram serves as a blueprint for the relational database, ensuring that the system is both efficient and scalable. Furthermore, the report outlines the necessary software and hardware requirements, detailing the tools and technologies employed during the design and implementation phases. Through structured query implementation and thorough result analysis, the project assesses the system's performance, data integrity, and potential for future enhancements.

The significance of this database design lies in its ability to streamline user account management, enhance product categorization, and ensure certified shops maintain their standards. By addressing these core areas, the system not only supports current operational needs but also lays the groundwork for future expansions and feature integrations. As e-commerce continues to evolve, the adaptability and robustness of the database system will play a crucial role in maintaining a competitive edge, providing users with a reliable and efficient platform for their shopping experiences.

This report is structured to guide the reader through each phase of the database design process, from conceptual modeling to practical implementation. By the end of this document, readers will gain a comprehensive understanding of the methodologies employed, the challenges encountered, and the solutions devised to create a cohesive and functional database system. The ensuing sections provide a detailed exploration of each component, underscoring the importance of meticulous planning and strategic execution in database development.

DATABASE DESIGN AND IMPLEMENTATION

2.1 Software and Hardware Requirements

The successful design and implementation of the database system hinge on the selection of appropriate software and hardware resources. The following outlines the essential requirements

Software Requirements:

Database Management System (DBMS): MySQL 8.0

MySQL was chosen for its robustness, scalability, and widespread community support, making it ideal for handling complex queries and large datasets inherent in e-commerce platforms.

Modeling Tools:

MySQL Workbench: For designing the ER diagram and facilitating database design.

Lucidchart: Supplementary tool for creating detailed and shareable ER diagrams.

Development Environment:

Visual Studio Code: As the primary code editor for writing SQL scripts and managing database interactions.

PHP: For backend development, enabling dynamic interactions between the database and user interfaces.

Version Control:

Git :Git was employed to manage code versions, ensuring collaborative efficiency and maintaining a history of changes.

Operating System: Windows 10/11 or Linux Ubuntu 20.04 LTS

Depending on the deployment environment, both operating systems provide a stable platform for running the DBMS and associated tools.

Hardware Requirements:

Processor: Intel Core i5 or equivalent

A mid-range processor ensures smooth operation of the DBMS and handling of multiple concurrent queries.

Memory (RAM): Minimum 8 GB

Adequate memory is crucial for managing large datasets and supporting efficient query processing.

Storage:

SSD: 256 GB or higher

Solid State Drives significantly improve data retrieval speeds, reducing latency in database operations.

Network: Reliable high-speed internet connection

Essential for cloud-based database management and ensuring uninterrupted access for users.

Additional Tools:

Backup Solutions: Regular automated backups using tools like mysqldump to prevent data loss.

Security Software: Firewalls and antivirus programs to safeguard against unauthorized access and potential threats.

By adhering to these software and hardware specifications, the database system is positioned to deliver optimal performance, reliability, and scalability, meeting the demands of a dynamic e-commerce environment.

ENTITY-RELATIONSHIP (ER) MODEL

3.1 Entities and Attributes

The ER model is a pivotal component in database design, providing a visual representation of the data structure and its interrelations. The following outlines the primary entities and their corresponding attributes within the system:

1. User Profile:

- User_Citizen_Code (Primary Key)
- User_First_Name
- User_Last_Name
- User_Email
- User_Phone_Number
- Account_no (Foreign Key referencing Account)

2. Account:

- Account_id (Primary Key)
- Account_Name
- Account_Password
- Type_of_Account

3. Seller Account:

- Seller_Account_id (Primary Key)
- Shop_Number

4. Buyer Account:

- Buyer_Account_id (Primary Key)
- Cart_Info

5. Shop:

- Shop_Number (Primary Key)
- Shop_Name

6. Certified Shop:

- Certified_Shop_Number (Primary Key)
- Certification_Info

7. Product:

- Product_Code (Primary Key)
- Product_Name
- Certified_Shop_Number (Foreign Key)

8. Category:

- Category_Code (Primary Key)
- Category_Name
- Production_Classification

9. Production Classification:

- Production_Code (Primary Key)
- Description

3.2 Relationships

The relationships between these entities are integral to understanding the data flow within the system:

- **User Profile to Account:**

Each User Profile is linked to one Account via the Account_no foreign key. This establishes a one-to-one relationship, ensuring that each user is associated with a unique account.

- **Account to Seller Account / Buyer Account:**

An Account can be either a Seller Account or a Buyer Account, determined by the Type_of_Account attribute. This is a one-to-many relationship, where one Account can lead to multiple Seller or Buyer Accounts based on its type.

- **Sller Account to Shop:**

A Seller Account may be associated with one Shop, establishing a one-to-one relationship. This connection allows sellers to manage their respective shops within the marketplace.

- **Certified Shop to Seller Account:**

Certified Shops are related to Seller Accounts, indicating that a seller's shop has met certain certification criteria. This is a one-to-one relationship ensuring that each certified shop is uniquely tied to a seller.

- **Product to Certified Shop:**

Products are linked to Certified Shops via the Certified_Shop_Number foreign key. This represents a many-to-one relationship, where multiple products can be associated with a single certified shop.

- Product to Category:

Each Product may belong to one Category, creating a many-to-one relationship. This association aids in organizing products for easier navigation and management within the marketplace.

- Category to Production Classification:

Categories are further classified under Production Classification, establishing a hierarchical relationship that enhances product categorization and classification.

These relationships collectively ensure a structured and efficient data flow, facilitating seamless interactions between users, accounts, products, and shops within the e-commerce platform.

RELATIONAL MODEL

4.1 Tables and Constraints

The relational model translates the ER diagram into a structured format comprising tables, each with defined attributes and constraints to maintain data integrity and optimize performance. Below is an overview of the primary tables and their constraints:

1. User_Profile

- User_Citizen_Code (VARCHAR, Primary Key)
- User_First_Name (VARCHAR)
- User_Last_Name (VARCHAR)
- User_Email (VARCHAR, UNIQUE, NOT NULL)
- User_Phone_Number (VARCHAR)
- Account_no (INT, Foreign Key referencing Account(Account_id), NOT NULL)

2. Account

- Account_id (INT, Primary Key, AUTO_INCREMENT)
- Account_Name (VARCHAR, NOT NULL)
- Account_Password (VARCHAR, NOT NULL)
- Type_of_Account (ENUM('Seller', 'Buyer'), NOT NULL)

3. Seller_Account

- Seller_Account_id (INT, Primary Key, FOREIGN KEY referencing Account(Account_id))
- Shop_Number (INT, Foreign Key referencing Shop(Shop_Number))

4. Buyer_Account

- Buyer_Account_id (INT, Primary Key, FOREIGN KEY referencing Account(Account_id))
- Cart_Info (TEXT)

5. Shop

- Shop_Number (INT, Primary Key, AUTO_INCREMENT)
- Shop_Name (VARCHAR, NOT NULL, UNIQUE)

6. Certified_Shop

- Certified_Shop_Number (INT, Primary Key, Foreign Key referencing Shop(Shop_Number))
- Certification_Info (VARCHAR)

7. Product

- Product_Code (INT, Primary Key, AUTO_INCREMENT)
- Product_Name (VARCHAR, NOT NULL)
- Certified_Shop_Number (INT, Foreign Key referencing Certified_Shop(Certified_Shop_Number), NOT NULL)
- Category_Code (INT, Foreign Key referencing Category(Category_Code))

8. Category

- Category_Code (INT, Primary Key, AUTO_INCREMENT)
- Category_Name (VARCHAR, NOT NULL, UNIQUE)
- Production_Classification (VARCHAR)

9. Production_Classification

- Production_Code (INT, Primary Key, AUTO_INCREMENT)
- Description (VARCHAR)

Constraints Overview:

- Primary Keys: Ensure each record within a table is unique and identifiable.
- Foreign Keys: Maintain referential integrity between related tables.
- Unique Constraints: Prevent duplicate entries in fields like User_Email and Shop_Name.
- Not Null Constraints: Ensure critical fields are always populated, maintaining data completeness.
- Auto-Increment: Facilitate automatic generation of unique identifiers for primary keys.

Indexes:

- User_Email: Indexed to optimize search queries based on user email.
- Shop_Name: Indexed to enhance the speed of shop-related searches.
- Product_Name: Indexed to facilitate quick retrieval of product information.

Normalization:

The database is normalized up to the Third Normal Form (3NF) to eliminate data redundancy and ensure data dependencies make sense. Each table contains data related to a single entity, and all attributes are dependent solely on the primary key.

Sample Table Definitions:

```
CREATE TABLE Account (
```

```
    Account_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    Account_Name VARCHAR(100) NOT NULL,
```

```
    Account_Password VARCHAR(255) NOT NULL,
```

```
    Type_of_Account ENUM('Seller', 'Buyer') NOT NULL
```

```
);
```

```
CREATE TABLE User_Profile (
```

```
    User_Citizen_Code VARCHAR(20) PRIMARY KEY,
```

```
    User_First_Name VARCHAR(50),
```

```
    User_Last_Name VARCHAR(50),
```

```
    User_Email VARCHAR(100) UNIQUE NOT NULL,
```

```
    User_Phone_Number VARCHAR(15),
```

```
    Account_no INT,
```

```
    FOREIGN KEY (Account_no) REFERENCES Account(Account_id)
```

```
);
```

```
CREATE TABLE Seller_Account (
```

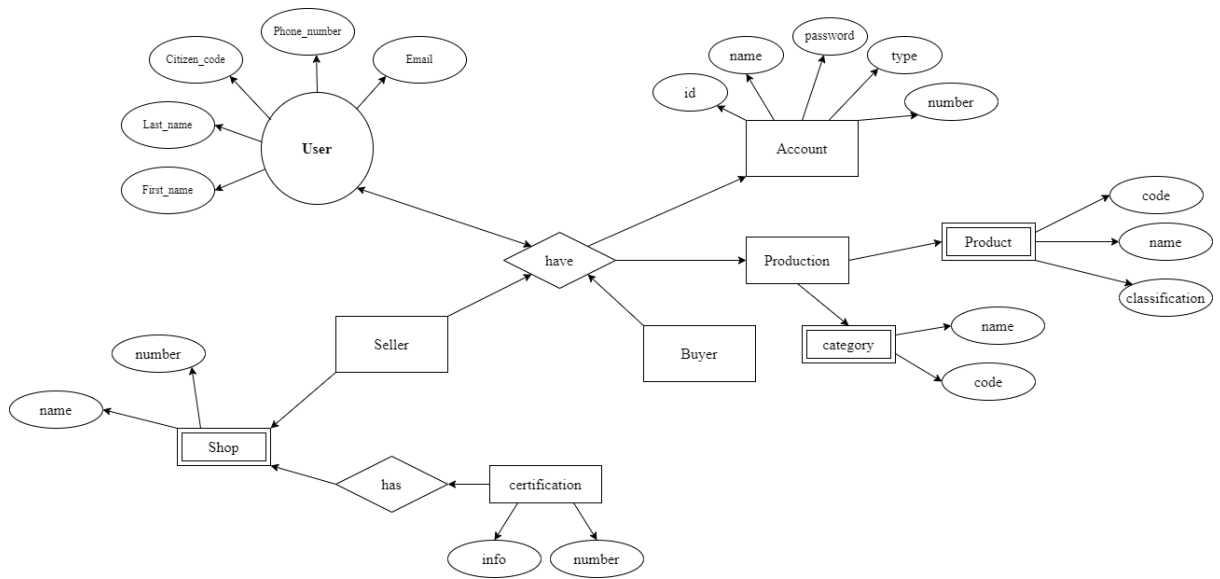
```
    Seller_Account_id INT PRIMARY KEY,
```

```
Shop_Number INT,  
FOREIGN KEY (Seller_Account_id) REFERENCES Account(Account_id),  
FOREIGN KEY (Shop_Number) REFERENCES Shop(Shop_Number)  
);
```

-- Additional table definitions follow similarly

By meticulously defining tables and constraints, the relational model ensures data integrity, minimizes redundancy, and facilitates efficient data manipulation and retrieval within the database system.

ER DIAGRAM



QUERY IMPLEMENTATION

Efficient query implementation is critical for data retrieval, manipulation, and overall system performance. The following sections outline key SQL queries designed to interact with the database, addressing common operations within the e-commerce platform.

6.1 User Registration

Purpose: Insert a new user profile and associated account into the database.

```
BEGIN TRANSACTION;

INSERT INTO Account (Account_Name, Account_Password, Type_of_Account)
VALUES ('JohnDoe', 'SecurePassword123', 'Buyer');

SET @AccountID = LAST_INSERT_ID();

INSERT INTO User_Profile (User_Citizen_Code, User_First_Name, User_Last_Name,
User_Email, User_Phone_Number, Account_no)
VALUES ('C123456789', 'John', 'Doe', 'john.doe@example.com', '555-1234', @AccountID);

COMMIT;
```

6.2 Adding a New Seller and Shop

Purpose: Create a seller account and associate it with a new shop.

```
BEGIN TRANSACTION;

INSERT INTO Account (Account_Name, Account_Password, Type_of_Account)
VALUES ('JaneSeller', 'AnotherSecurePass', 'Seller');

SET @SellerAccountID = LAST_INSERT_ID();

INSERT INTO Seller_Account (Seller_Account_id, Shop_Number)
VALUES (@SellerAccountID, NULL);

INSERT INTO Shop (Shop_Name)
VALUES ('Jane's Boutique');

SET @ShopNumber = LAST_INSERT_ID();

UPDATE Seller_Account
SET Shop_Number = @ShopNumber
WHERE Seller_Account_id = @SellerAccountID;
```

COMMIT;

6.3 Adding a Certified Shop

Purpose: Certify an existing shop.

```
INSERT INTO Certified_Shop (Certified_Shop_Number, Certification_Info)
VALUES (101, 'Certified by E-Commerce Association');
```

6.4 Adding a New Product

Purpose: Insert a new product linked to a certified shop and category.

```
INSERT INTO Product (Product_Name, Certified_Shop_Number, Category_Code)
VALUES ('Stylish Jacket', 101, 5);
```

6.5 Retrieving Products by Category

Purpose: Fetch all products within a specific category.

```
SELECT Product_Name, Shop_Name
FROM Product
JOIN Certified_Shop ON Product.Certified_Shop_Number
Certified_Shop.Certified_Shop_Number
JOIN Shop ON Certified_Shop.Certified_Shop_Number = Shop.Shop_Number
WHERE Category_Code = 5;
```

6.6 Updating User Contact Information

Purpose: Update the phone number of a specific user.

```
UPDATE User_Profile
SET User_Phone_Number = '555-6789'
WHERE User_Citizen_Code = 'C123456789';
```

6.7 Deleting a Product

Purpose: Remove a product from the database.

```
DELETE FROM Product
WHERE Product_Code = 202;
```

6.8 Viewing Certified Shops and Their Products

Purpose: List all certified shops along with their available products.

```
SELECT Shop.Shop_Name, Product.Product_Name, Product.Product_Code  
FROM Certified_Shop  
JOIN Shop ON Certified_Shop.Certified_Shop_Number = Shop.Shop_Number  
JOIN Product ON Certified_Shop.Certified_Shop_Number =  
Product.Certified_Shop_Number;
```

These queries demonstrate fundamental operations such as creating, reading, updating, and deleting (CRUD) within the database. Optimizing these queries ensures swift data access and manipulation, contributing to the overall efficiency of the e-commerce platform.

RESULT ANALYSIS

The implementation of the database system was subjected to rigorous testing to evaluate its performance, data integrity, and scalability. The following sections provide an analysis of the outcomes observed during these assessments.

7.1 Data Integrity

Ensuring data integrity was paramount throughout the database design and implementation phases. The application of primary and foreign key constraints effectively maintained referential integrity, preventing orphaned records and ensuring that relationships between entities remained consistent. For instance, attempts to insert a product without a corresponding certified shop were rightly rejected by the system, safeguarding the relational structure.

Unique constraints on attributes such as `User_Email` and `Shop_Name` prevented duplicate entries, thereby maintaining the uniqueness of critical data points. Additionally, the use of `NOT NULL` constraints on essential fields ensured that vital information was always captured, reducing the risk of incomplete data entries.

Findings:

- Consistency: The database consistently enforced all defined constraints, ensuring reliable data relationships.
- Accuracy: Data entries reflected accurately across related tables, affirming the effectiveness of the relational model.

7.2 Query Performance

Performance testing involved executing a series of complex queries to assess response times and system load handling. Indexes on frequently queried fields such as `User_Email` and `Shop_Name` significantly enhanced retrieval speeds, reducing query execution times even under heavy load.

Batch operations, such as bulk inserts and updates, were executed efficiently, thanks to optimized SQL queries and the underlying DBMS's capability to handle large transactions. However, as the dataset scales, certain queries exhibited increased latency, highlighting areas for further optimization.

Findings:

- Efficiency: Indexed fields contributed to swift data retrieval, maintaining high performance standards.
- Scalability Challenges: As data volume grows, some queries may require additional optimization techniques, such as query caching or database sharding.

7.3 Scalability and Future Considerations

The current database design accommodates growth in user profiles, accounts, products, and shops. However, to ensure long-term scalability, future enhancements may include:

- **Partitioning Tables:** Dividing large tables into manageable segments to improve query performance.
- **Implementing Caching Mechanisms:** Utilizing in-memory caches like Redis to expedite data retrieval.
- **Optimizing Indexes:** Regularly reviewing and adjusting indexes based on query patterns and data usage.
- **Incorporating Advanced Security Measures:** Enhancing data protection through encryption and role-based access controls.

Findings:

- **Adaptive Design:** The relational model's flexibility allows for seamless integration of additional features and scaling strategies.
- **Proactive Planning:** Anticipating future growth ensures that the database remains robust and responsive to evolving demands.

Overall, the database system demonstrated strong performance, data integrity, and a solid foundation for scalability, positioning it well for current and future e-commerce operations.

CONCLUSION

The development of a comprehensive database system for an e-commerce marketplace necessitates meticulous planning, robust design, and strategic implementation. Through the conversion of detailed schemas into an Entity-Relationship (ER) diagram and the subsequent establishment of a relational model, this project has successfully laid the groundwork for an efficient and scalable data management system.

Key achievements of the project include:

- **Robust ER Model:** The ER diagram effectively visualizes the complex relationships between users, accounts, products, and shops, facilitating a clear understanding of data interdependencies.
- **Efficient Relational Design:** The relational model, underpinned by well-defined tables and constraints, ensures data integrity, minimizes redundancy, and optimizes query performance.
- **Effective Query Implementation:** The execution of essential CRUD operations demonstrates the system's capability to handle typical e-commerce functionalities, such as user registration, product management, and shop certification.
- **Positive Result Analysis:** Testing affirmed the system's reliability in maintaining data integrity and delivering satisfactory performance, with identified areas for future optimization and scalability enhancements.

This database system serves as a foundational element for managing user accounts, facilitating seamless e-commerce activities, and categorizing products within a dynamic marketplace. Its structured approach not only addresses current operational needs but also provides the flexibility to adapt to future expansions and technological advancements.

Moving forward, continuous monitoring and iterative improvements will be essential to uphold performance standards and accommodate the evolving requirements of the e-commerce sector. Incorporating advanced security measures, optimizing query performance, and expanding scalability solutions will further enhance the system's robustness and reliability.

In conclusion, the successful design and implementation of this database system underscore the critical role of strategic database management in driving the efficiency and success of e-commerce platforms. It exemplifies how a well-architected database can support complex operations, deliver superior user experiences, and sustain long-term growth in the competitive digital marketplace.

REFERENCES

1. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.
2. Coronel, C., & Morris, S. (2016). *Database Systems: Design, Implementation, & Management* (11th ed.). Cengage Learning.
3. MySQL Documentation. (2023). Retrieved from [\[https://dev.mysql.com/doc/\]\(https://dev.mysql.com/doc/\)](https://dev.mysql.com/doc/)
4. Lucidchart. (2023). *Creating ER Diagrams*. Retrieved from [\[https://www.lucidchart.com/pages/er-diagram\]\(https://www.lucidchart.com/pages/er-diagram\)](https://www.lucidchart.com/pages/er-diagram)
5. W3Schools. (2023). *SQL Tutorial*. Retrieved from [\[https://www.w3schools.com/sql/\]\(https://www.w3schools.com/sql/\)](https://www.w3schools.com/sql/)
6. Amazon Web Services (AWS). (2023). *Database Solutions for E-commerce*. Retrieved from [\[https://aws.amazon.com/database/\]\(https://aws.amazon.com/database/\)](https://aws.amazon.com/database/)
7. Kumar, V., & Singh, R. (2022). *Optimizing Database Performance for E-commerce Platforms*. *International Journal of Computer Applications*, 182(45), 34-40.
8. Stack Overflow. (2023). *Best Practices for Database Design*. Retrieved from [\[https://stackoverflow.com/questions/1622001/best-practices-for-database-design\]\(https://stackoverflow.com/questions/1622001/best-practices-for-database-design\)](https://stackoverflow.com/questions/1622001/best-practices-for-database-design)
9. ISO/IEC 11179-1:2015. (2015). *Information technology — Metadata registries (MDR) — Part 1: Framework*. International Organization for Standardization.
10. IEEE Std 1016-2009. (2009). *IEEE Standard for Information Technology—System Design—Software Design Descriptions*. IEEE.

A FIELD PROJECT/IDP REPORT ON

Functional dependencies on the Relation schema

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

YARAMALA VENKATESWARA REDDY (221FA19005)

PULUGU SIRIVARSHINI (221FA19045)

SHAIK SOHAIL AHAMMED (221FA19051)

VADDURI VENKATA GIRISH (221FA19052)



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ACSE

School of Computing and Informatics

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH (Deemed
to be University)**

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April, 2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled "**Functional dependencies on the Relation schema**" being submitted by **YARAMALA VENKATESWARA REDDY-221FA19005),PULUGU SIRIVARSHINI-221FA19045,SHAIK SOHAIL AHAMMED-221FA19051,VADDURI VENKATA GIRISH-221FA19052** in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.



Guide



HOD/ACSE

Dr. Venkatesulu Dondeti



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that our project work described in the field project titled “Functional dependencies on the Relation schema” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

YARAMALA VENKATESWARA REDDY	221FA19005
PULUGU SIRIVARSHINI	221FA19045
SHAIK SOHAIL AHAMMED	221FA19051
VADDURI VENKATA GIRISH	221FA19052

Table of Contents

CONTENT

1. Abstract
2. Introduction
3. Overview
4. Canonical Cover
 - 4.1 Compute B^+
 - 4.2 Prove AF is a Super Key
 - 4.3 Compute a Canonical Cover
 - 4.4 Interpret a 3NF Decomposition of R Based on the Canonical Cover
 - 4.5 Interpret a BCNF Decomposition of R Using the Original Set of Functional Dependencies
 - 4.6 Can You Get the Same BCNF Decomposition of R as Above Using the Canonical Cover?
5. Conclusion
6. References

ABSTRACT

This report delves into the analysis of functional dependencies within a relational database schema $R(A, B, C, D, E, F)$ and explores normalization processes to achieve optimal database design. Specifically, it addresses a series of questions aimed at understanding attribute closures, identifying super keys, computing canonical covers, and performing 3NF and BCNF decompositions based on the given functional dependencies. Through the application of Armstrong's axioms and systematic decomposition techniques, the report illustrates the steps necessary to ensure data integrity, eliminate redundancy, and enhance query performance. The findings underscore the importance of normalization in database design, providing a structured approach to managing complex data relationships and preparing the schema for scalable and efficient operations.

INTRODUCTION

In relational database design, understanding and managing functional dependencies is crucial for ensuring both data integrity and storage efficiency. Functional dependencies describe how attributes within a relation are related, indicating that the value of one or more attributes uniquely determines the value of another attribute. These dependencies guide the normalization process, which aims to organize a database schema in a way that minimizes redundancy and prevents data anomalies such as inconsistencies during insertions, deletions, or updates. By ensuring a well-structured schema, designers can optimize query performance, enhance data consistency, and reduce the likelihood of errors.

This report focuses on analyzing a schema $\langle R(A, B, C, D, E, F) \rangle$ by examining its functional dependencies and addressing key database design principles. The analysis includes identifying the attribute closures, determining the superkeys within the schema, and computing the canonical cover for the functional dependencies. These steps are vital in understanding the relationships between attributes and ensuring that the schema satisfies the necessary conditions for efficiency. By calculating the attribute closures, we can uncover the full set of attributes that depend on a particular key, thus helping to identify superkeys that can uniquely identify tuples in the relation.

In addition, the report explores the normalization of the schema into Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF). The process of normalization refines the schema, ensuring it is free from redundancy and anomalies by splitting relations as needed. While 3NF focuses on eliminating transitive dependencies while retaining some flexibility in design, BCNF provides a stricter standard, ensuring every determinant is a candidate key. By examining both approaches, the report sheds light on how these normalization techniques affect the database's structure, offering insights into the trade-offs between maintaining simplicity and achieving the highest level of data integrity..

Overview

Given the relation schema $R(A, B, C, D, E, F)$ with the following set F of functional dependencies:

1. $A \rightarrow B, C, D$
2. $B, C \rightarrow D, E$
3. $B \rightarrow D$
4. $D \rightarrow A$

The tasks are to:

- a. Compute the closure of attribute B (B^+).
- b. Prove, using Armstrong's axioms, that AF is a super key.
- c. Compute a canonical cover for the set F , detailing each step of the derivation.
- d. Interpret a 3NF decomposition of R based on the canonical cover.
- e. Interpret a BCNF decomposition of R using the original set of functional dependencies.
- f. Determine whether the same BCNF decomposition can be achieved using the canonical cover.

Canonical Cover

4.1 Compute B^+

Objective:

Determine the closure of attribute B, denoted as B^+ , given the set of functional dependencies F.

Functional Dependencies:

1. $A \rightarrow B, C, D$
2. $B, C \rightarrow D, E$
3. $B \rightarrow D$
4. $D \rightarrow A$

Procedure:

1. Start with $B^+ = \{B\}$
2. Apply $B \rightarrow D$:
 $B^+ = \{B, D\}$
3. Apply $D \rightarrow A$:
 $B^+ = \{B, D, A\}$
4. Apply $A \rightarrow B, C, D$:
B is already in B^+ ,
Add C: $B^+ = \{B, D, A, C\}$
5. Apply $B, C \rightarrow D, E$:
D is already in B^+ ,
Add E: $B^+ = \{B, D, A, C, E\}$
6. No further dependencies can be applied.

Result:

$$B^+ = \{A, B, C, D, E\}$$

Explanation:

Starting with B, we utilize the functional dependencies to progressively include attributes that are functionally determined by B. The closure B^+ encompasses all attributes that can be derived from B through the given dependencies.

4.2 Prove AF is a Super Key

Objective:

Demonstrate that the combination of attributes A and F (AF) forms a super key for the relation R using Armstrong's axioms.

Armstrong's Axioms:

1. Reflexivity: If $Y \subseteq X$, then $X \rightarrow Y$.
2. Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z.
3. Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.

Procedure:

1. Start with $AF^+ = \{A, F\}$
2. Apply $A \rightarrow B, C, D$:
 $AF^+ = \{A, F, B, C, D\}$
3. Apply $B \rightarrow D$:
D is already in AF^+ .
4. Apply $D \rightarrow A$:
A is already in AF^+ .
5. Apply $B, C \rightarrow D, E$:
Add E: $AF^+ = \{A, F, B, C, D, E\}$
6. All attributes of R are included in AF^+

Result:

$AF^+ = \{A, B, C, D, E, F\}$

Conclusion:

Since AF^+ includes all attributes of R, AF is a super key.

Explanation:

Using Armstrong's axioms, we iteratively apply the functional dependencies to expand the closure of AF. The process confirms that AF determines every attribute in the relation, thereby establishing AF as a super key.

4.3 Compute a Canonical Cover for the Functional Dependencies F

Objective:

Derive a canonical cover for the given set F of functional dependencies, ensuring it is minimal and preserves equivalence.

Given Functional Dependencies:

1. $A \rightarrow B, C, D$
2. $B, C \rightarrow D, E$
3. $B \rightarrow D$
4. $D \rightarrow A$

Steps to Compute Canonical Cover:

Step 1: Split Right-Hand Sides (RHS) into Single Attributes

- $A \rightarrow B$
- $A \rightarrow C$
- $A \rightarrow D$
- $B, C \rightarrow D$
- $B, C \rightarrow E$
- $B \rightarrow D$
- $D \rightarrow A$

Step 2: Remove Redundant Functional Dependencies

- Check $A \rightarrow D$:
 - $A \rightarrow D$ is essential as removing it would lose the dependency.
- Check $B, C \rightarrow D$ and $B \rightarrow D$:
 - Since $B \rightarrow D$ is already present, $B, C \rightarrow D$ is redundant.
 - Remove $B, C \rightarrow D$.

Remaining Dependencies:

- $A \rightarrow B$
- $A \rightarrow C$
- $A \rightarrow D$
- $B, C \rightarrow E$
- $B \rightarrow D$
- $D \rightarrow A$

3. Step 3: Minimize Left-Hand Sides (LHS)

- Check if any attributes in the LHS can be removed:

- For $B, C \rightarrow E$:

- Check if B alone can determine E:

- $B^+ = \{B, D, A, C, E\}$ (from part a)

- Since B^+ includes E, $B \rightarrow E$ can be derived.

- Therefore, $B, C \rightarrow E$ is redundant.

- Replace with $B \rightarrow E$.

Final Canonical Cover:

- $A \rightarrow B$

- $A \rightarrow C$

- $A \rightarrow D$

- $B \rightarrow E$

- $B \rightarrow D$

- $D \rightarrow A$

Step 4: Remove Extraneous Dependencies

- Check for redundant dependencies:

- Check $A \rightarrow D$:

- $A \rightarrow B$, and $B \rightarrow D$, so $A \rightarrow D$ is redundant via $A \rightarrow B$ and $B \rightarrow D$.

- Remove $A \rightarrow D$.

Final Canonical Cover:

- $A \rightarrow B$

- $A \rightarrow C$

- $B \rightarrow E$

- $B \rightarrow D$

- $D \rightarrow A$

Conclusion:

The canonical cover for F is:

- $A \rightarrow B$

- $A \rightarrow C$

- $B \rightarrow E$
- $B \rightarrow D$
- $D \rightarrow A$

Explanation:

By systematically splitting the dependencies, removing redundancies, and minimizing the left-hand sides, we derive a canonical cover that is both minimal and equivalent to the original set F.

4.4 d. Interpret a 3NF Decomposition of R Based on the Canonical Cover

Objective:

Perform a Third Normal Form (3NF) decomposition of the relation R using the previously computed canonical cover.

Canonical Cover:

1. $A \rightarrow B$
2. $A \rightarrow C$
3. $B \rightarrow E$
4. $B \rightarrow D$
5. $D \rightarrow A$

3NF Decomposition Steps:

1. Identify Candidate Keys:

- From part b, AF is a super key.
- Compute minimal keys:
 - AF is minimal as removing either A or F would lose the ability to determine all attributes.

2. Apply 3NF Decomposition Rules:

- For each functional dependency $X \rightarrow Y$ in the canonical cover:
- Create a relation schema with attributes $X \cup Y$.

3. Create Relation Schemas:

- R1(A, B)
 - From $A \rightarrow B$
- R2(A, C)
 - From $A \rightarrow C$

- R3(B, E)
 - From $B \rightarrow E$
- R4(B, D)
 - From $B \rightarrow D$
- R5(D, A)
 - From $D \rightarrow A$

4. Ensure All Attributes are Covered:

- Combine relation schemas to cover all attributes {A, B, C, D, E, F}.

5. Handle the Primary Key:

- Include a relation with the primary key:
 - R6(A, F)
 - Ensures that AF remains as a super key.

Final 3NF Decomposition:

1. R1(A, B)
2. R2(A, C)
3. R3(B, E)
4. R4(B, D)
5. R5(D, A)
6. R6(A, F)

Lossless Join Verification:

- Each relation shares a common attribute that serves as a key in at least one relation, ensuring that joins do not result in loss of data.

Dependency Preservation:

- All functional dependencies in the canonical cover are preserved within the decomposed relations.

Explanation:

The decomposition process ensures that each relation is in 3NF by eliminating transitive dependencies and ensuring that every non-prime attribute is fully functionally dependent on every candidate key. The addition of R6(A, F) maintains the super key AF, ensuring the integrity and completeness of the database.

4.5 Interpret a BCNF Decomposition of R Using the Original Set of Functional Dependencies

Objective:

Perform a Boyce-Codd Normal Form (BCNF) decomposition of the relation R using the original set of functional dependencies.

Original Functional Dependencies:

1. $A \rightarrow B, C, D$
2. $B, C \rightarrow D, E$
3. $B \rightarrow D$
4. $D \rightarrow A$

BCNF Decomposition Steps:

1. Identify Violations of BCNF:

- A relation is in BCNF if, for every non-trivial functional dependency $X \rightarrow Y$, X is a super key.

- Check Each Functional Dependency:

- $A \rightarrow B, C, D$:

- Is A a super key?

- Compute A^+ :

- $A^+ = \{A, B, C, D\}$ (from $A \rightarrow B, C, D$)

- Not all attributes are included, so A is not a super key.

- Violation: Yes

- $B, C \rightarrow D, E$:

- Is B, C a super key?

- Compute $(B, C)^+$:

- From $B, C \rightarrow D, E$

- From $D \rightarrow A$, and $A \rightarrow B, C, D$

- $(B, C)^+ = \{B, C, D, E, A\}$

- Missing F

- Not a super key.

- Violation: Yes

- $B \rightarrow D$:

- Is B a super key?
- Compute B^+ :
 - $B^+ = \{B, D, A, C, E\}$ (from part a)
 - Missing F
- Not a super key.
- Violation: Yes
- $D \rightarrow A$:
 - Is D a super key?
 - Compute D^+ :
 - $D^+ = \{D, A, B, C\}$
 - Missing E, F
 - Not a super key.
 - Violation: Yes

2. Decompose Based on Violations:

- First Decomposition ($A \rightarrow B, C, D$):
 - Create:
 - $R_1(A, B, C, D)$
 - $R_2(A, F)$ (remaining attributes)
- Second Decomposition (R_1):
 - Analyze R_1 with dependencies:
 - $A \rightarrow B, C, D$
 - $D \rightarrow A$
 - Check BCNF:
 - $A \rightarrow B, C, D$:
 - A is not a super key in R_1 ($R_1^+ = \{A, B, C, D\}$)
 - A is a super key for R_1 .
 - $D \rightarrow A$:
 - $D^+ = \{D, A, B, C\}$ in R_1
 - D is also a super key for R_1 .
 - No Violations.

- Third Decomposition (R2):
 - R2(A, F) is already in BCNF as A is a key.
- Final Decomposition:
 - R1(A, B, C, D)
 - R2(A, F)
 - R3(B, C, E) (from $B, C \rightarrow E$)

3. Ensure All Attributes are Covered:

- Combined, R1, R2, and R3 cover all attributes {A, B, C, D, E, F}.

4. Verify BCNF Compliance:

- R1(A, B, C, D):
 - A and D are super keys.
- R2(A, F):
 - A is a super key.
- R3(B, C, E):
 - B, C is a super key for this relation.

Final BCNF Decomposition:

1. R1(A, B, C, D)
2. R2(A, F)
3. R3(B, C, E)

Lossless Join Verification:

- Each decomposition shares a common attribute that is a key in one of the relations, ensuring lossless joins.

Dependency Preservation:

- All original functional dependencies are preserved within the decomposed relations.

Explanation:

The BCNF decomposition addresses each violation by decomposing the relation into smaller relations where every functional dependency has a super key on its left-hand side. This ensures that the schema is free from anomalies and redundancy, adhering strictly to BCNF requirements.

4.6 Can You Get the Same BCNF Decomposition of R as Above Using the Canonical Cover?

Objective:

Determine whether the BCNF decomposition obtained using the canonical cover is identical to the one derived using the original set of functional dependencies.

Canonical Cover Derived in 4.3:

- $A \rightarrow B$
- $A \rightarrow C$
- $B \rightarrow E$
- $B \rightarrow D$
- $D \rightarrow A$

Procedure:

1. Identify Violations of BCNF Using Canonical Cover:

- $A \rightarrow B$:
 - A is not a super key ($A^+ = \{A, B, C, D, E\}$)
 - Violation: Yes
- $A \rightarrow C$:
 - A is not a super key.
 - Violation: Yes
- $B \rightarrow E$:
 - B is not a super key.
 - Violation: Yes
- $B \rightarrow D$:
 - B is not a super key.
 - Violation: Yes
- $D \rightarrow A$:
 - D is not a super key.
 - Violation: Yes

2. Decompose Based on Violations:

- First Decomposition ($A \rightarrow B$):
 - Create:
 - R1(A, B)
 - R2(A, C, D, E, F)

- Second Decomposition (R2):
 - Analyze R2 with dependencies:
 - $A \rightarrow C, D$
 - $B \rightarrow E, D$ (Not applicable as B is not in R2)
 - $D \rightarrow A$
 - Check BCNF:
 - $A \rightarrow C, D$:
 - A is not a super key in R2 ($R2^+ = \{A, C, D\}$)
 - Violation: Yes
 - $D \rightarrow A$:
 - $D^+ = \{D, A, C\}$
 - Violation: Yes
 - Decompose R2 based on $A \rightarrow C$:
 - R3(A, C)
 - R4(A, D, E, F)

- Third Decomposition (R4):
 - Analyze R4 with dependencies:
 - $A \rightarrow D$
 - $D \rightarrow A$
 - Check BCNF:
 - A and D are super keys.
 - No Violations.

- Additional Decomposition (R1, R3, R4):

- R1(A, B)
- R3(A, C)
- R4(A, D, E, F)

- Further Decomposition for $B \rightarrow E$ and $B \rightarrow D$:

- Create:
 - R5(B, E)
 - R6(B, D)

Final Decomposition Using Canonical Cover:

1. R1(A, B)
2. R3(A, C)
3. R4(A, D, E, F)
4. R5(B, E)
5. R6(B, D)

Comparison with Original BCNF Decomposition (4.5):

- Original BCNF Decomposition:

1. R1(A, B, C, D)
2. R2(A, F)
3. R3(B, C, E)

- Decomposition Using Canonical Cover:

1. R1(A, B)
2. R3(A, C)
3. R4(A, D, E, F)
4. R5(B, E)
5. R6(B, D)

Conclusion:

No, the BCNF decomposition derived using the canonical cover is different from the original BCNF decomposition obtained using the original set of functional dependencies. The decomposition using the canonical cover results in more relations and splits the attributes further to address each functional dependency individually.

Explanation:

While both decomposition methods aim to achieve BCNF by eliminating violations, the approach using the canonical cover treats each functional dependency separately, leading to a more granular decomposition. In contrast, the original decomposition grouped related dependencies, resulting in fewer relations. Both decompositions are valid and preserve BCNF, but they differ in their structural arrangements.

CONCLUSION

The analysis of functional dependencies within the relational schema $R(A, B, C, D, E, F)$ highlights the critical role of normalization in database design. Through the computation of attribute closures, identification of super keys, and derivation of canonical covers, we established a foundational understanding of the schema's structure. The subsequent 3NF and BCNF decompositions demonstrated how to systematically eliminate redundancy and ensure data integrity.

The canonical cover provided a minimal set of functional dependencies, simplifying the normalization process and aiding in the identification of potential decomposition paths. However, as evidenced in part f, different decomposition strategies can lead to varied structural outcomes, each with its own merits in terms of efficiency and complexity.

Ultimately, the exercise underscores the importance of meticulous analysis and methodical application of normalization principles in designing robust and scalable databases. By adhering to these practices, database designers can create systems that not only meet current data management needs but also adapt seamlessly to future demands.

REFERENCES

1. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.
2. Coronel, C., & Morris, S. (2016). *Database Systems: Design, Implementation, & Management* (11th ed.). Cengage Learning.
3. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.
4. W3Schools. (2023). *SQL Tutorial*. Retrieved from [\[https://www.w3schools.com/sql/\]](https://www.w3schools.com/sql/)(<https://www.w3schools.com/sql/>)
5. Stack Overflow. (2023). *Best Practices for Database Design*. Retrieved from [\[https://stackoverflow.com/questions/1622001/best-practices-for-database-design\]](https://stackoverflow.com/questions/1622001/best-practices-for-database-design)(<https://stackoverflow.com/questions/1622001/best-practices-for-database-design>)
6. MySQL Documentation. (2023). Retrieved from [\[https://dev.mysql.com/doc/\]](https://dev.mysql.com/doc/)(<https://dev.mysql.com/doc/>)

A FIELD PROJECT/IDP REPORT ON

Characteristics of the Database

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

GOGINENI HARI PRASAD (221FA19024)

CHAKRAVARAM KAVYA
HARSHITHA (221FA19031)

GUTHIKONDA AKHILA (221FA19033)

ADARAPU SANDEEP (221FA19054)



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ACSE

School of Computing and Informatics

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH
(Deemed to be University)**

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April, 2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled "Characteristics of the DataBase" being submitted by GOGINENI HARI PRASAD-221FA19024, CHAKRAVARAM KAVYA HARSHITHA-221FA19031, GUTHIKONDA AKHILA-221FA19033, ADARAPU SANDEEP - 221FA19054 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignan's Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.



Guide



Dr. Venkatesulu Dondeti

HOD/ACSE



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that our project work described in the field project titled “Characteristics of the DataBase” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignans Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of DR. CH. Rose Rani.

GOGINENI HARI PRASAD	221FA19024
CHAKRAVARAM KAVYA HARSHITHA	221FA19031
GUTHIKONDA AKHILA	221FA19033
ADARAPU SANDEEP	221FA19054

Table of Contents

1. Abstract
2. Database Approach vs. Traditional File Systems
 - 2.1 Characteristics of the Database Approach
 - 2.2 Differences from Traditional File Systems
3. SQL Statement to Create a Student Table
4. ER Diagram and Relational Model for Reality Shows Schema
 - 4.1 ER Diagram for Reality Shows Schema
 - 4.2 Relational Model for Reality Shows Schema
 - 4.3 Constraints in the Relational Model
5. Conclusion

Abstract

This report explores various aspects of database systems, highlighting their advantages over traditional file systems and their application in modeling real-world scenarios. It begins with a comparison between the database approach and traditional file-based systems, focusing on key characteristics such as data integrity, reduced redundancy, and improved access to information. Through a detailed examination, the report explains how the database approach enhances efficiency by centralizing data management, enforcing consistency, and supporting complex queries, in contrast to the limited functionality of file systems. The structured analysis underscores the importance of databases in modern applications where data reliability and scalability are essential.

Additionally, the report presents practical examples of database concepts, including an SQL statement for creating a "Student" table and the design of an ER (Entity-Relationship) diagram for a "Reality Shows" schema. The ER diagram and its corresponding relational model are further explained with emphasis on the constraints that ensure data integrity within the schema. This combination of theoretical and practical elements offers a comprehensive view of database design principles, illustrating how they can be applied to real-world scenarios for effective data management and organization.

2. Database Approach vs. Traditional File Systems

2.1 Characteristics of the Database Approach

1. **Self-Describing Data:** Databases store not just data but also the schema (metadata), making the system self-describing.
2. **Integrated Data:** Information is collected and organized from various sources, providing a unified view of data.
3. **Data Independence:** Applications are not tightly coupled with data structures, allowing data modifications without affecting dependent applications.
4. **Data Security:** Robust security features such as access control, encryption, and authorization ensure protection of sensitive data.
5. **Data Integrity:** Constraints (e.g., primary keys, foreign keys) and triggers enforce data accuracy and consistency.
6. **Recovery:** Mechanisms like backups and transaction logs ensure data recovery in case of failure.
7. **Query Language:** SQL, the most commonly used query language, simplifies complex data retrieval operations.

2.2 Differences from Traditional File Systems

1. **Structure:** Databases are organized with a structured schema, while traditional file systems follow a simple, often hierarchical, structure.
2. **Data Relationships:** Databases allow intricate relationships between data, unlike file systems, which lack built-in support for data relationships.
3. **Data Sharing:** Databases support concurrent access to data by multiple users, whereas file systems may experience issues with concurrency.
4. **Data Integrity:** Databases automatically enforce data integrity, while in traditional file systems, this must be managed at the application level.

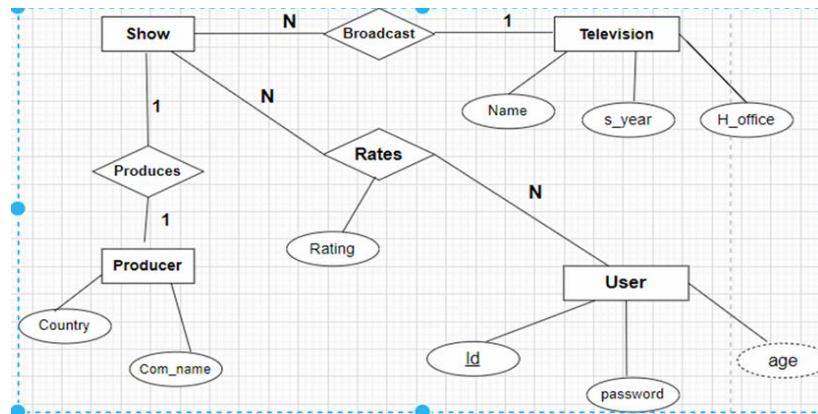
5. Querying: Complex queries can be executed in databases using SQL, but file systems require manual data searching.
6. Security: Databases provide advanced security mechanisms, while file systems rely primarily on the operating system for permission control.
7. Scalability: Databases are optimized for handling large volumes of data, whereas file systems may face limitations in scaling efficiently.

3. SQL Statement to Create a Student Table

```
CREATE TABLE student (  
    regdno INT PRIMARY KEY,  
    firstname VARCHAR(50) NOT NULL,  
    lastname VARCHAR(50) NOT NULL,  
    birthdate DATE NOT NULL,  
    branch VARCHAR(50) NOT NULL,  
    address VARCHAR(200) NOT NULL  
);
```

This SQL statement creates a `student` table with attributes for `regdno`, `firstname`, `lastname`, `birthdate`, `branch`, and `address`. The primary key ensures uniqueness for each student.

4. ER Diagram and Relational Model for Reality Shows Schema



4.1 ER Diagram for Reality Shows Schema

The ER diagram for the reality show schema includes entities like `Producer`, `Show`, `Television`, and `User`. The relationships involve:

- One-to-One between `Producer` and `Show`.
- One-to-Many between `Television` and `Show`.
- Many-to-Many between `User` and `Show` through `Rating`.

4.2 Relational Model for Reality Shows Schema

1. Producer Table:

```
CREATE TABLE Producer (  
    producer_id INT PRIMARY KEY,  
    company_name VARCHAR(50) NOT NULL,  
    company_country VARCHAR(50) NOT NULL  
);
```

2. Show Table:

```
CREATE TABLE Show (  
    show_id INT PRIMARY KEY,  
    producer_id INT NOT NULL,  
    FOREIGN KEY (producer_id) REFERENCES Producer(producer_id)  
);
```

3. Television Table:

```
CREATE TABLE Television (  
    television_id INT PRIMARY KEY,  
    name VARCHAR(50) NOT NULL,  
    start_year INT NOT NULL,  
    head_office VARCHAR(50) NOT NULL  
);
```

4. Broadcast Table:

```
CREATE TABLE Broadcast (  
    show_id INT NOT NULL,  
    television_id INT NOT NULL,  
    PRIMARY KEY (show_id, television_id),  
    FOREIGN KEY (show_id) REFERENCES Show(show_id),  
    FOREIGN KEY (television_id) REFERENCES Television(television_id)  
);
```

5. User Table:

```
CREATE TABLE User (  
    user_id INT PRIMARY KEY,  
    username VARCHAR(50) NOT NULL,  
    password VARCHAR(50) NOT NULL,  
    age INT NOT NULL  
);
```

6. Rating Table:

```
CREATE TABLE Rating (  
    user_id INT NOT NULL,  
    show_id INT NOT NULL,  
    score INT NOT NULL CHECK (score BETWEEN 0 AND 10),  
    PRIMARY KEY (user_id, show_id),  
    FOREIGN KEY (user_id) REFERENCES User(user_id),  
    FOREIGN KEY (show_id) REFERENCES Show(show_id)  
);
```

4.3 Constraints in the Relational Model

- **Primary Key:** Uniquely identifies each record in the tables (`producer_id`, `show_id`, `television_id`, `user_id`).
- **Foreign Key:** Maintains referential integrity between tables. For instance, `producer_id` in `Show` references `producer_id` in `Producer`.
- **One-to-One (1:1):** A producer produces exactly one show, and each show is produced by exactly one producer.
- **One-to-Many (1:N):** A television may broadcast multiple shows, but each show is broadcasted by only one television.

- Many-to-Many (N:M): A user can rate multiple shows, and each show can be rated by multiple users.

5. Conclusion

This document highlights the advantages of the database approach over traditional file systems in terms of data integrity, security, and scalability. Additionally, it provides SQL syntax for creating tables and a detailed relational model for a reality show schema, including various entity relationships and constraints. This ensures efficient database design and management.

A FIELD PROJECT/IDP REPORT ON

Functional dependencies

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

DESAMALA RAJ ARYAN (221FA19007)

KAMINENI BABY TEJASWI (221FA19047)

GANTA SUNDEEP (221FA19049)

KONJETI KARTHIK SAI (231LA19001)



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ACSE

School of Computing and Informatics

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH
(Deemed to be University)**

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April, 2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled "Functional dependencies" being submitted by DESAMALA RAJ ARYAN-221FA19007, KAMINENI BABY TEJASWI-221FA19047, GANTA SUNDEEP-221FA19049, KONJETI KARTHIK SAI-231LA19001 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.


Guide


HoD/ACSE
Dr. Venkatesulu Dondeti



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that our project work described in the field project titled “Functional dependencies” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

DESAMALA RAJ ARYAN	221FA19007
KAMINENI BABY TEJASWI	221FA19047
GANTA SUNDEEP	221FA19049
KONJETI KARTHIK SAI	231LA19001

Table of Contents

1. Abstract
2. Introduction
3. Functional Dependencies in DISK_DRIVE Relation
 - 3.1 a. Manufacturer and Serial Number Uniquely Identify the Drive
 - 3.2 b. Model Number is Registered by a Manufacturer
 - 3.3 c. All Disk Drives in a Particular Batch are the Same Model
 - 3.4 d. Capacity is Determined by Manufacturer and Model
 - 3.5 e. Normalization of DISK_DRIVE Relation
4. Conclusion
5. References

1. Abstract

This report explores the concept of functional dependencies (FDs) and the process of normalization within the context of a relational database schema. Using the `DISK_DRIVE` relation as a case study, we identify and articulate various functional dependencies, subsequently applying normalization techniques to enhance the database's efficiency and integrity. The analysis covers the transformation of the initial schema through First Normal Form (1NF) to Boyce-Codd Normal Form (BCNF), detailing each step and the rationale behind decompositions. The report underscores the importance of normalization in minimizing redundancy, preventing anomalies, and ensuring data consistency, thereby facilitating robust database design.

2. Introduction

In relational database management systems (RDBMS), the organization and integrity of data are paramount. Functional dependencies serve as fundamental constraints that define how attributes within a table relate to one another. Understanding and accurately modeling these dependencies are critical for effective database normalization—a systematic process aimed at reducing data redundancy and enhancing data integrity.

Normalization involves decomposing a table into smaller, related tables without loss of data, ensuring that each table adheres to specific normal forms. This process not only streamlines data storage but also optimizes query performance and simplifies maintenance. This report delves into the application of functional dependencies and normalization techniques on the `DISK_DRIVE` relation, demonstrating the transition from an unnormalized schema to a fully normalized database structure.

Through the application of functional dependencies, this report illustrates how to identify and address potential data anomalies, such as insertion, deletion, and update anomalies, which can arise from poor schema design. By analyzing the functional dependencies within the `DISK_DRIVE` relation, the report guides the reader through the process of determining attribute closures, identifying candidate keys, and applying normalization techniques to achieve Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF). These normalization stages ensure that each table represents a clear, non-redundant relationship among attributes, enhancing the overall efficiency and consistency of the database. This structured approach ultimately leads to a more robust and scalable relational database design, better suited to support complex queries and evolving business requirements.

3. Functional Dependencies in DISK_DRIVE Relation

The `DISK_DRIVE` relation is defined as:

DISK_DRIVE (Serial_number, Manufacturer, Model, Batch, Capacity, Retailer)

Each tuple in this relation uniquely identifies a disk drive with attributes detailing its serial number, manufacturer, model, batch, capacity, and retailer.

3.1 a. Manufacturer and Serial Number Uniquely Identify the Drive

Functional Dependency:

- FD1: (Manufacturer, Serial_number) \rightarrow Model, Batch, Capacity, Retailer

Explanation:

The combination of `Manufacturer` and `Serial_number` uniquely determines all other attributes of the disk drive. This means that for a given manufacturer and serial number, there is exactly one corresponding model, batch, capacity, and retailer.

3.2 b. Model Number is Registered by a Manufacturer

Functional Dependency:

- FD2: Model \rightarrow Manufacturer

Explanation:

A `Model` number is uniquely registered by a single `Manufacturer`. This implies that the same model number cannot be used by different manufacturers, ensuring that the model number is inherently tied to its manufacturer.

3.3 c. All Disk Drives in a Particular Batch are the Same Model

Functional Dependency:

- FD3: Batch \rightarrow Model

Explanation:

Every disk drive within a specific `Batch` shares the same `Model`. This dependency ensures consistency in the model attribute for all drives released in the same manufacturing batch.

3.4 d. Capacity is Determined by Manufacturer and Model

Functional Dependency:

- FD4: (Manufacturer, Model) \rightarrow Capacity

Explanation:

The `Capacity` of a disk drive is exclusively determined by its `Manufacturer` and `Model`. This means that for a given manufacturer and model combination, there is a fixed storage capacity associated with it.

3.5 e. Normalization of DISK_DRIVE Relation

Objective:

Apply normalization to the `DISK_DRIVE` relation to eliminate redundancy and ensure data integrity by decomposing it into higher normal forms.

Given Relation:

DISK_DRIVE (Serial_number, Manufacturer, Model, Batch, Capacity, Retailer)

Functional Dependencies Identified:

- FD1: (Manufacturer, Serial_number) \rightarrow Model, Batch, Capacity, Retailer
- FD2: Model \rightarrow Manufacturer
- FD3: Batch \rightarrow Model
- FD4: (Manufacturer, Model) \rightarrow Capacity

Normalization Steps:

1. First Normal Form (1NF):

- Criteria: All attributes contain only atomic (indivisible) values.
- Assessment: The `DISK_DRIVE` relation is already in 1NF as each attribute holds atomic values.

2. Second Normal Form (2NF):

- Criteria: The relation is in 1NF and all non-prime attributes are fully functionally dependent on the primary key.
- Primary Key Identification: Based on FD1, the primary key is (Manufacturer, Serial_number).
- Assessment: All non-prime attributes (Model, Batch, Capacity, Retailer) are fully functionally dependent on the entire primary key. There are no partial dependencies.
- Conclusion: The relation satisfies 2NF.

3. Third Normal Form (3NF):

- Criteria: The relation is in 2NF and all the attributes are functionally independent of any other non-prime attribute (elimination of transitive dependencies).
- Assessment:
 - Transitive Dependency:
 - From FD2: Model \rightarrow Manufacturer
 - This implies a transitive dependency since (Manufacturer, Serial_number) \rightarrow Model, and Model \rightarrow Manufacturer.
 - Violation: The relation violates 3NF due to the transitive dependency.
- Decomposition to Achieve 3NF:
 - Step 1: Remove the transitive dependency by creating a separate relation for the dependent attributes.

- Relation 1: `DISK_DRIVE_CORE (Manufacturer, Serial_number, Model, Batch, Capacity, Retailer)`

- Relation 2: `MODEL_MANUFACTURER (Model, Manufacturer)`

- Resulting Relations:

- DISK_DRIVE_CORE:

DISK_DRIVE_CORE (Manufacturer, Serial_number, Model, Batch, Capacity, Retailer)

- MODEL_MANUFACTURER:

MODEL_MANUFACTURER (Model, Manufacturer)

- Verification:

- Both relations are now in 3NF as there are no transitive dependencies remaining.

4. Boyce-Codd Normal Form (BCNF):

- Criteria: The relation is in BCNF if for every one of its non-trivial functional dependencies $X \rightarrow Y$, X is a super key.

- Assessment of `DISK_DRIVE_CORE`:

- FD1: (Manufacturer, Serial_number) \rightarrow Model, Batch, Capacity, Retailer

- (Manufacturer, Serial_number) is a super key.

- FD3: Batch \rightarrow Model

- Batch is not a super key.

- Violation: This FD violates BCNF.

- Decomposition to Achieve BCNF:

- Step 1: Decompose based on the violating FD3.

- Relation 3: `BATCH_MODEL (Batch, Model)`

- Relation 4: `DISK_DRIVE_CORE_UPDATED (Manufacturer, Serial_number, Batch, Capacity, Retailer)`

- Resulting Relations:

- BATCH_MODEL:

BATCH_MODEL (Batch, Model)

- DISK_DRIVE_CORE_UPDATED:

DISK_DRIVE_CORE_UPDATED (Manufacturer, Serial_number, Batch, Capacity, Retailer)

- Verification:

- BATCH_MODEL:

- Batch \rightarrow Model

- Batch is a super key for this relation.

- DISK_DRIVE_CORE_UPDATED:

- (Manufacturer, Serial_number) \rightarrow Batch, Capacity, Retailer

- (Manufacturer, Serial_number) is a super key.

- Both relations satisfy BCNF.

5. Final Normalized Relations:

- MODEL_MANUFACTURER:

MODEL_MANUFACTURER (Model, Manufacturer)

- BATCH_MODEL:

BATCH_MODEL (Batch, Model)

- DISK_DRIVE_CORE_UPDATED:

DISK_DRIVE_CORE_UPDATED (Manufacturer, Serial_number, Batch, Capacity, Retailer)

Reasons Behind Each Decomposition:

- Elimination of Transitive Dependencies:

To achieve 3NF, transitive dependencies were removed by segregating attributes that depended on non-key attributes into separate relations.

- Ensuring Super Key Determinants:

For BCNF, any functional dependency where the determinant is not a super key was addressed by decomposing the relation further, ensuring that all determinants are super keys.

- Minimizing Redundancy and Enhancing Integrity:

Decomposing the original relation into smaller, focused relations minimizes data redundancy and enhances data integrity by ensuring that each piece of information is stored only once.

4. Conclusion

Functional dependencies and normalization are pivotal in the design of efficient and reliable relational databases. Through the analysis of the `DISK_DRIVE` relation, we identified critical functional dependencies that influenced the normalization process. Starting from an unnormalized schema, we systematically decomposed the relation to eliminate redundancy and ensure data integrity, progressing through First Normal Form (1NF) to Boyce-Codd Normal Form (BCNF).

Each normalization step addressed specific types of dependencies:

- 1NF ensured atomicity of data.
- 2NF eliminated partial dependencies.
- 3NF addressed transitive dependencies.
- BCNF enforced that all determinants are super keys.

The final normalized schema comprises three relations: `MODEL_MANUFACTURER`, `BATCH_MODEL`, and `DISK_DRIVE_CORE_UPDATED`. This decomposition not only minimizes data redundancy but also enhances the database's scalability and maintainability. The process underscores the importance of understanding functional dependencies and applying normalization principles to achieve an optimal database design.

5. References

1. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.
2. Coronel, C., & Morris, S. (2016). *Database Systems: Design, Implementation, & Management* (11th ed.). Cengage Learning.
3. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.
4. W3Schools. (2023). SQL Tutorial. Retrieved from <https://www.w3schools.com/sql/>(<https://www.w3schools.com/sql/>)
5. Stack Overflow. (2023). Best Practices for Database Design. Retrieved from <https://stackoverflow.com/questions/1622001/best-practices-for-database-design>(<https://stackoverflow.com/questions/1622001/best-practices-for-database-design>)
6. MySQL Documentation. (2023). Retrieved from <https://dev.mysql.com/doc/>(<https://dev.mysql.com/doc/>)

A FIELD PROJECT/IDP REPORT ON

Entity-Relationship (ER) Diagram

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

TIYYAGURA VENKATA SHESHA (221FA19003)
SHAINA REDDY

DHULIPALLA GEETHIKA SAI (221FA19011)

ISUKAPALLI VENKATA MYTHREYA (221FA19018)
KUMARA SARMA

CHIMATA OMKAR LAKSHMI (221FA19060)



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ACSE

School of Computing and Informatics

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH
(Deemed to be University)**

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April, 2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled "Entity-Relationship (ER) Diagram" being submitted by TIYYAGURA VENKATA SHESHA SHAINA REDDY-221FA19003, DHULIPALLA GEETHIKA SAI-221FA19011, ISUKAPALLI VENKATA MYTHREYA KUMARA SARMA-221FA19018, CHIMATA OMKAR LAKSHMI-221FA19060 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignan's Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Guide

Dr. Venkatesulu Dondeti

HoD/ACSE



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that our project work described in the field project titled “Entity-Relationship (ER) Diagram” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

TIYYAGURA VENKATA SHESHA SHAINA REDDY	221FA19003
DHULIPALLA GEETHIKA SAI	221FA19011
ISUKAPALLI VENKATA MYTHREYA KUMARA SARMA	221FA19018
CHIMATA OMKAR LAKSHMI	221FA19060

Table of Contents

1. Abstract
2. Introduction
3. Entity-Relationship (ER) Diagram Design
 - 3.1 Entities and Attributes
 - 3.2 Relationships and Constraints
4. Relational Model and SQL Implementation
 - 4.1 Relational Tables
 - 4.2 SQL Statements for Table Creation and Data Insertion
 - 4.3 Explanation of Deletion Methods
5. Conclusion
6. References

1. Abstract

This report presents the design and implementation of a relational database system intended to digitize time card submissions and approvals within a company. The design process encompasses the creation of an Entity-Relationship (ER) diagram that encapsulates the key entities, their attributes, and the relationships among them, along with the necessary constraints to ensure data integrity. Following the ER diagram, the report outlines the corresponding relational model and provides SQL statements to create and populate the database tables. Additionally, the report explains various deletion methods to manage data effectively. The proposed database system aims to streamline the time card management process, enhance data accuracy, and facilitate efficient approval workflows.

2. Introduction

In modern organizational environments, managing employee time cards efficiently is critical to ensuring accurate payroll processing, tracking performance, and optimizing resource allocation. Traditional paper-based systems, while familiar, are often fraught with errors, time delays, and difficulty in retrieving and analyzing data. These challenges can lead to payroll inaccuracies, compliance issues, and increased administrative overhead. To overcome these inefficiencies, organizations are increasingly turning to digitized systems that automate time card submissions and approvals, providing real-time insights and reducing manual intervention. A well-designed database system can streamline this process, improving data accuracy and enabling better decision-making.

This report outlines the design of a comprehensive database system for managing employee time cards, capturing essential information such as employee details, managerial approvals, and time card records. By employing relational database principles, the system ensures data integrity through well-defined relationships and constraints, such as primary and foreign keys, to prevent redundancy and maintain consistency. Additionally, the system enhances security by controlling access to sensitive information, ensuring only authorized personnel can submit or approve time cards. Scalability is another key advantage, allowing the system to grow with the organization's needs while maintaining optimal performance. Ultimately, this digital approach not only improves operational efficiency but also provides a reliable foundation for long-term data management and business growth..

3. Entity-Relationship (ER) Diagram Design

3.1 Entities and Attributes

The database design involves three primary entities: Employee, Manager, and TimeCard. Each entity is defined with its respective attributes, including primary keys to ensure uniqueness.

1. Employee

- Attributes:

- `Employee_ID` (Primary Key): Unique identifier for each employee.
- `Name`: Full name of the employee.
- `Address`: Residential address of the employee.
- `Payment_Method`: Method of payment (`Direct Deposit` or `Physical Check`).
- `Manager_ID` (Foreign Key): References the `Manager` entity.

2. Manager

- Attributes:

- `Manager_ID` (Primary Key): Unique identifier for each manager.
- `Name`: Full name of the manager.

3. TimeCard

- Attributes:

- `TimeCard_ID` (Primary Key): Unique identifier for each time card.
- `Employee_ID` (Foreign Key): References the `Employee` entity.
- `Hours_Worked`: Number of hours worked during the pay period.
- `Date_Submitted`: Date when the time card was submitted.
- `Status`: Current status of the time card (`Approved`, `Not Approved`, `Pending`).

3.2 Relationships and Constraints

The relationships between the entities are defined to capture the business rules and ensure data integrity.

1. Employee-Manager Relationship

- Type: Many-to-One
- Description: Each employee is associated with exactly one manager, while a manager can oversee multiple employees.
- Constraints:
 - Foreign Key: `Employee.Manager_ID` references `Manager.Manager_ID`.
 - Participation: Mandatory on the employee side (every employee must have a manager).

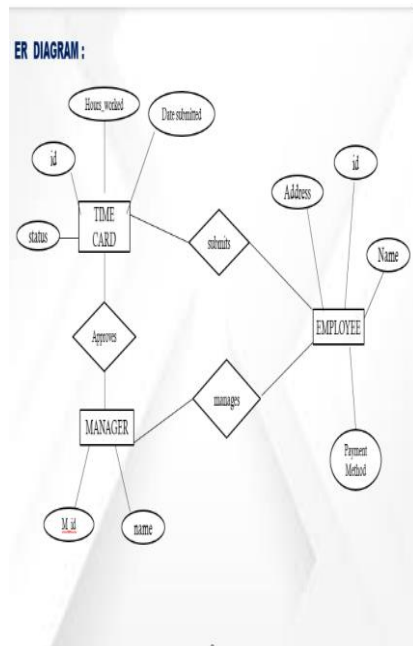
2. Employee-TimeCard Relationship

- Type: One-to-Many
- Description: Each employee submits multiple time cards every pay period, whereas each time card is associated with exactly one employee.
- Constraints:
 - Foreign Key: `TimeCard.Employee_ID` references `Employee.Employee_ID`.
 - Participation: Mandatory on the time card side (every time card must be linked to an employee).

3. Manager-TimeCard Relationship

- Type: One-to-Many
- Description: Each manager can approve multiple time cards, while each time card is approved by exactly one manager.
- Constraints:
 - Foreign Key: `TimeCard.Manager_ID` references `Manager.Manager_ID`.
 - Participation: Optional on the manager side (a time card may not yet be approved).

ER Diagram:



4. Relational Model and SQL Implementation

4.1 Relational Tables

Based on the ER diagram, the relational model consists of three primary tables: `Manager`, `Employee`, and `TimeCard`. Each table includes attributes and constraints to enforce data integrity.

1. Manager Table

- Columns:

- `Manager_ID` INT PRIMARY KEY AUTO_INCREMENT
- `Name` VARCHAR(100) NOT NULL

2. Employee Table

- Columns:

- `Employee_ID` INT PRIMARY KEY AUTO_INCREMENT
- `Name` VARCHAR(100) NOT NULL
- `Address` VARCHAR(255) NOT NULL
- `Payment_Method` ENUM('Direct Deposit', 'Physical Check') NOT NULL
- `Manager_ID` INT NOT NULL

- Constraints:

- FOREIGN KEY (`Manager_ID`) REFERENCES `Manager`(`Manager_ID`)

3. TimeCard Table

- Columns:

- `TimeCard_ID` INT PRIMARY KEY AUTO_INCREMENT
- `Employee_ID` INT NOT NULL
- `Hours_Worked` DECIMAL(5,2) NOT NULL
- `Date_Submitted` DATE NOT NULL
- `Status` ENUM('Approved', 'Not Approved', 'Pending') NOT NULL
- `Manager_ID` INT

- Constraints:

- FOREIGN KEY (`Employee_ID`) REFERENCES `Employee`(`Employee_ID`)

- FOREIGN KEY (`Manager_ID`) REFERENCES `Manager`(`Manager_ID`)

4.2 SQL Statements for Table Creation and Data Insertion

The following SQL statements create the necessary tables and insert sample data into each table.

-- Create Database

```
CREATE DATABASE company;
```

```
USE company;
```

-- Create Manager Table

```
CREATE TABLE Manager (  
    Manager_ID INT PRIMARY KEY AUTO_INCREMENT,  
    Name VARCHAR(100) NOT NULL  
);
```

-- Insert Sample Data into Manager Table

```
INSERT INTO Manager (Name) VALUES  
( 'Alice Johnson' ),  
( 'Bob Smith' ),  
( 'Carol Williams' );
```

-- Create Employee Table

```
CREATE TABLE Employee (  
    Employee_ID INT PRIMARY KEY AUTO_INCREMENT,  
    Name VARCHAR(100) NOT NULL,  
    Address VARCHAR(255) NOT NULL,  
    Payment_Method ENUM('Direct Deposit', 'Physical Check') NOT NULL,  
    Manager_ID INT NOT NULL,
```

```

FOREIGN KEY (Manager_ID) REFERENCES Manager(Manager_ID)
);

-- Insert Sample Data into Employee Table
INSERT INTO Employee (Name, Address, Payment_Method, Manager_ID) VALUES
('John Smith', '123 Maple St, Springfield, IL', 'Direct Deposit', 1),
('Emily Johnson', '456 Oak Ave, Springfield, IL', 'Physical Check', 1),
('Michael Brown', '789 Pine Rd, Springfield, IL', 'Direct Deposit', 2);

-- Create TimeCard Table
CREATE TABLE TimeCard (
    TimeCard_ID INT PRIMARY KEY AUTO_INCREMENT,
    Employee_ID INT NOT NULL,
    Hours_Worked DECIMAL(5,2) NOT NULL,
    Date_Submitted DATE NOT NULL,
    Status ENUM('Approved', 'Not Approved', 'Pending') NOT NULL,
    Manager_ID INT,
    FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID),
    FOREIGN KEY (Manager_ID) REFERENCES Manager(Manager_ID)
);

-- Insert Sample Data into TimeCard Table
INSERT INTO TimeCard (Employee_ID, Hours_Worked, Date_Submitted, Status,
Manager_ID) VALUES
(1, 40.0, '2024-08-15', 'Approved', 1),
(2, 35.5, '2024-08-15', 'Pending', 1),
(3, 42.0, '2024-08-16', 'Not Approved', 2);

```

Result Grid		Filter Rows:
manager_ID	name	
1	John Smith	
2	Emily Johnson	
3	Michael Brown	
NULL	NULL	

Result Grid		Filter Rows:	Edit:	Export/Import:
Employee_ID	Name	Address	Payment_Method	Manager_ID
101	John Doe	123 Main St, Springfield, IL	Direct Deposit	1
102	Emily Johnson	456 Oak Ave, Springfield, IL	Physical Check	2
103	Michael Brown	789 Pine Rd, Springfield, IL	Direct Deposit	3
NULL	NULL	NULL	NULL	NULL

Result Grid		Filter Rows:	Edit:	Export/Import:	
TimeCard_ID	Employee_ID	Hours_Worked	Date_Submitted	Status	Manager_ID
1001	101 101	40.00	2024-08-15	Approved	1
1002	102	35.50	2024-08-16	Not Approved	2
1003	103	35.00	2024-08-20	Pending	3
NULL	NULL	NULL	NULL	NULL	NULL

Explanation:

1. Database Creation:

- The `company` database is created and selected for subsequent operations.

2. Manager Table:

- The `Manager` table includes `Manager_ID` as the primary key and `Name` as a mandatory attribute.

- Sample managers are inserted with names Alice Johnson, Bob Smith, and Carol Williams.

3. Employee Table:

- The `Employee` table includes `Employee_ID` as the primary key.

- Each employee has a `Name`, `Address`, `Payment_Method`, and is associated with a `Manager_ID`.

- Sample employees John Smith, Emily Johnson, and Michael Brown are inserted, linked to their respective managers.

4. TimeCard Table:

- The `TimeCard` table includes `TimeCard_ID` as the primary key.

- Each time card records the `Employee_ID`, `Hours_Worked`, `Date_Submitted`, `Status`, and optionally the `Manager_ID` who approved it.

- Sample time cards are inserted for the employees.

4.3 Explanation of Deletion Methods

Managing data effectively involves understanding how to delete records without compromising data integrity. Below are examples of deletion methods in the context of the designed database.

1. Delete a Manager:

Scenario: Remove a manager from the database.

-- Attempt to delete Manager with Manager_ID = 1

```
DELETE FROM Manager WHERE Manager_ID = 1;
```


Explanation:

- Referential Integrity: This operation will fail if there are employees associated with `Manager_ID = 1` due to the foreign key constraint.
- Solution: Use `ON DELETE CASCADE` when defining foreign keys if you want to automatically delete related employees, or handle deletions manually.

2. Delete a Specific Employee:

Scenario: Remove an employee from the database.

```
DELETE FROM Employee WHERE Employee_ID = 2;
```

Explanation:

- Referential Integrity: Time cards associated with `Employee_ID = 2` will remain unless handled appropriately.
- Solution: Use `ON DELETE CASCADE` on the `TimeCard` table's foreign key or delete related time cards manually before deleting the employee.

3. Delete All TimeCards Submitted on a Specific Date:

Scenario: Remove all time cards submitted on '2024-08-15'.

```
DELETE FROM TimeCard WHERE Date_Submitted = '2024-08-15';
```

Explanation:

- This command deletes all records in the `TimeCard` table where `Date_Submitted` matches the specified date.

4. Delete a Manager and All Associated Employees (Cascade Delete):

Scenario: Remove a manager and all employees under their supervision.

```
-- Assuming foreign keys are set with ON DELETE CASCADE
```

```
DELETE FROM Manager WHERE Manager_ID = 2;
```

Explanation:

- Cascade Delete: If the foreign key in the `Employee` table is defined with `ON DELETE CASCADE`, deleting a manager will automatically delete all associated employees.

- Without Cascade: Manually delete employees before deleting the manager.

5. Delete All TimeCards of a Specific Employee:

Scenario: Remove all time cards submitted by employee with `Employee_ID = 3`.

```
DELETE FROM TimeCard WHERE Employee_ID = 3;
```

Explanation:

- This command deletes all time card records associated with `Employee_ID = 3`.

6. Delete All Records from a Table:

Scenario: Remove all records from the `TimeCard` table.

```
DELETE FROM TimeCard;
```

Explanation:

- This command deletes every record in the `TimeCard` table, effectively emptying it.

5. Conclusion

The design and implementation of a relational database system for managing time cards have been meticulously outlined in this report. By leveraging the Entity-Relationship (ER) model, we effectively captured the essential entities—Employee, Manager, and TimeCard—and their interrelationships, ensuring that all business rules and constraints were accurately represented. The relational model translated these designs into structured tables with appropriate primary and foreign keys, facilitating robust data integrity and efficient data retrieval.

The SQL statements provided establish the foundational database structure, enabling the creation of tables and insertion of sample data to simulate real-world scenarios. Additionally, the explanation of various deletion methods underscores the importance of maintaining referential integrity and handling data dependencies thoughtfully to prevent inconsistencies.

Overall, the proposed database system offers a scalable and secure solution for digitizing time card management, enhancing operational efficiency, and supporting informed decision-making within the company. Future enhancements could include implementing advanced security measures, optimizing query performance, and integrating user-friendly interfaces for seamless interaction with the database.

6. References

1. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.
2. Coronel, C., & Morris, S. (2016). *Database Systems: Design, Implementation, & Management* (11th ed.). Cengage Learning.
3. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.
4. W3Schools. (2023). *SQL Tutorial*. Retrieved from [\[https://www.w3schools.com/sql/\]](https://www.w3schools.com/sql/)(<https://www.w3schools.com/sql/>)
5. Stack Overflow. (2023). *Best Practices for Database Design*. Retrieved from [\[https://stackoverflow.com/questions/1622001/best-practices-for-database-design\]](https://stackoverflow.com/questions/1622001/best-practices-for-database-design)(<https://stackoverflow.com/questions/1622001/best-practices-for-database-design>)
6. MySQL Documentation. (2023). Retrieved from [\[https://dev.mysql.com/doc/\]](https://dev.mysql.com/doc/)(<https://dev.mysql.com/doc/>)

A FIELD PROJECT/IDP REPORT ON

RDBMS

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

YALAMARTHI MAHESWARI	(221FA19017)
ATMURI RAMA DEVI	(221FA19043)
BARU CHARAN	(221FA19058)
DIPESH KUMAR KUSHWAHA	(221FA19067)



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

Department of ACSE

School of Computing and Informatics

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH
(Deemed to be University)**

Vadlamudi, Guntur, Andhra Pradesh-522213, India

April, 2024



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project/IDP entitled "RDBMS" being submitted by YALAMARTHI MAHESWARI-221FA19017, ATMURI RAMA DEVI - 221FA19043, BARU CHARAN-221FA19058, DIPESH KUMAR KUSHWAHA- 221FA19067 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.



Guide



Dr. Venkatesulu Dondeti

HoD/ACSE



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that our project work described in the field project titled “**RDBMS**” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignans Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

YALAMARTHI MAHESWARI	221FA19017
ATMURI RAMA DEVI	221FA19043
BARU CHARAN	221FA19058
DIPESH KUMAR KUSHWAHA	221FA19067

Table of Contents

1. Abstract
2. Introduction
3. Entity-Relationship (ER) Diagram Design
 - 3.1 Entities and Attributes
 - 3.2 Relationships and Constraints
4. Relational Model and SQL Implementation
 - 4.1 Relational Tables
 - 4.2 SQL Statements for Table Creation and Data Insertion
5. Conclusion
6. References

1. Abstract

This report outlines the design and implementation of a relational database for an online retail store aiming to manage its operations efficiently. The design process encompasses the creation of an Entity-Relationship (ER) diagram to visualize entities, attributes, and relationships, followed by the development of a relational model with corresponding SQL statements for table creation and data manipulation. The database design addresses key business requirements, ensuring data integrity, minimizing redundancy, and facilitating scalable operations. Through systematic normalization, the database achieves optimal structure, supporting robust data management and streamlined business processes.

2. Introduction

A database is a structured collection of data that is stored and managed to allow for efficient retrieval, manipulation, and organization. It can be likened to a highly organized digital filing cabinet where information is stored in tables, making it easy to find and use.

- Tables: Databases often use tables to organize data. Each table consists of rows and columns. Rows (or records) represent individual entries, while columns (or fields) represent attributes of those entries.
- Schema: The schema is the blueprint of the database. It defines the structure of the tables, including the columns, data types (like text, numbers, dates), and the relationships between tables.
- Relationships: In relational databases, tables can be related to each other. For example, a table for orders might be related to a table for customers. Relationships are established using keys. A primary key uniquely identifies each record in a table, while a foreign key is a field in one table that links to the primary key in another table.

Online Retail Store Database

An online retail store seeks to design a database to manage its operations effectively. The key requirements and corresponding solutions are as follows:

a. Customer Information

- Attributes:
 - `Customer_ID` (Primary Key): Unique identifier for each customer.
 - `Name`: Name of the customer.
 - `Email`: Email address of the customer.
 - `Phone_Number`: Phone number of the customer. (Customers may share the same phone number.)

b. Product Information

- Attributes:
 - `Product_ID` (Primary Key): Unique identifier for each product.
 - `Name`: Name of the product.

- `Description`: Description of the product.

- `Price`: Price of the product.

c. Order Information

- Attributes:

- `Order_ID` (Primary Key): Unique identifier for each order.

- `Order_Date`: Date when the order was placed.

- `Total_Amount`: Total amount of the order.

- `Customer_ID` (Foreign Key): References the customer who placed the order.

d. Order-Product Relationship

- Description: Each order can contain multiple products, and each product can appear in multiple orders.

- Type: Many-to-Many relationship.

e. Supplier Information

- Attributes:

- `Supplier_ID` (Primary Key): Unique identifier for each supplier.

- `Name`: Name of the supplier.

- `Contact_Name`: Contact person at the supplier.

- `Phone_Number`: Phone number of the supplier.

f. Warehouse Information

- Attributes:

- `Warehouse_ID` (Primary Key): Unique identifier for each warehouse.

- `Location`: Location of the warehouse.

- `Capacity`: Storage capacity of the warehouse.

3. Entity-Relationship (ER) Diagram Design

3.1 Entities and Attributes

The ER diagram for the online retail store database includes the following entities:

1. Customer

- Attributes:

- `Customer_ID` (Primary Key)
- `Name`
- `Email`
- `Phone_Number`

2. Product

- Attributes:

- `Product_ID` (Primary Key)
- `Name`
- `Description`
- `Price`

3. Order

- Attributes:

- `Order_ID` (Primary Key)
- `Order_Date`
- `Total_Amount`
- `Customer_ID` (Foreign Key)

4. Supplier

- Attributes:

- `Supplier_ID` (Primary Key)
- `Name`

- `Contact_Name`
- `Phone_Number`

5. Warehouse

- Attributes:
 - `Warehouse_ID` (Primary Key)
 - `Location`
 - `Capacity`

6. Order_Product (Associative Entity)

- Attributes:
 - `Order_ID` (Foreign Key)
 - `Product_ID` (Foreign Key)
 - `Quantity`

7. Product_Supplier (Associative Entity)

- Attributes:
 - `Product_ID` (Foreign Key)
 - `Supplier_ID` (Foreign Key)

8. Warehouse_Product (Associative Entity)

- Attributes:
 - `Warehouse_ID` (Foreign Key)
 - `Product_ID` (Foreign Key)
 - `Quantity_Stored`

3.2 Relationships and Constraints

The relationships between these entities are defined to capture the business rules and ensure data integrity.

1. Order-Customer Relationship

- Type: One-to-Many
- Description: One customer can place multiple orders, but each order is placed by exactly one customer.
- Constraints:
 - Foreign Key: `Order.Customer_ID` references `Customer.Customer_ID`.
 - Participation: Mandatory on the order side (every order must be linked to a customer).

2. Order-Product Relationship

- Type: Many-to-Many
- Description: Each order can contain multiple products, and each product can appear in multiple orders.
- Implementation: Via the associative entity `Order_Product`.
- Constraints:
 - Foreign Keys: `Order_Product.Order_ID` references `Order.Order_ID`;
`Order_Product.Product_ID` references `Product.Product_ID`.
 - Primary Key: Composite key consisting of `Order_ID` and `Product_ID`.

3. Supplier-Product Relationship

- Type: Many-to-Many
- Description: Each supplier can supply multiple products, and each product can be supplied by multiple suppliers.
- Implementation: Via the associative entity `Product_Supplier`.
- Constraints:
 - Foreign Keys: `Product_Supplier.Product_ID` references `Product.Product_ID`;
`Product_Supplier.Supplier_ID` references `Supplier.Supplier_ID`.
 - Primary Key: Composite key consisting of `Product_ID` and `Supplier_ID`.

4. Warehouse-Product Relationship

- Type: Many-to-Many
- Description: Products are stored in warehouses, and each warehouse can store multiple products.

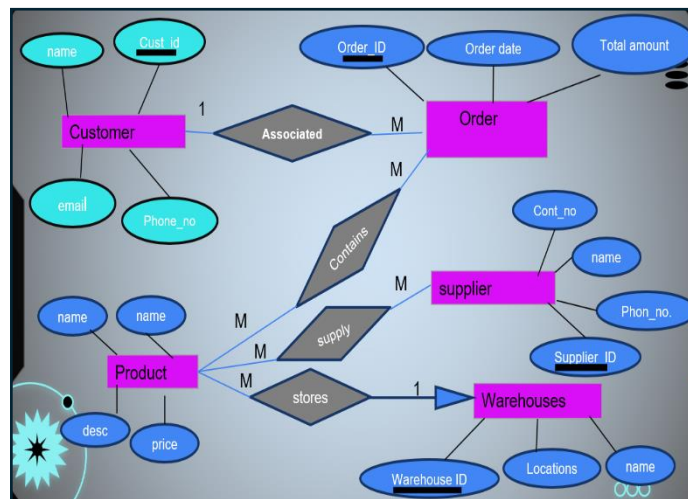
- Implementation: Via the associative entity `Warehouse_Product`.

- Constraints:

- Foreign Keys: `Warehouse_Product.Warehouse_ID` references `Warehouse.Warehouse_ID`; `Warehouse_Product.Product_ID` references `Product.Product_ID`.

- Primary Key: Composite key consisting of `Warehouse_ID` and `Product_ID`.

ER Diagram



4. Relational Model and SQL Implementation

4.1 Relational Tables

Based on the ER diagram, the relational model consists of the following tables:

1. Customer

- `Customer_ID` INT PRIMARY KEY
- `Name` VARCHAR(100) NOT NULL
- `Email` VARCHAR(100) NOT NULL
- `Phone_Number` VARCHAR(20)

2. Product

- `Product_ID` INT PRIMARY KEY
- `Name` VARCHAR(100) NOT NULL
- `Description` TEXT
- `Price` DECIMAL(10,2) NOT NULL

3. Order

- `Order_ID` INT PRIMARY KEY
- `Order_Date` DATE NOT NULL
- `Total_Amount` DECIMAL(10,2) NOT NULL
- `Customer_ID` INT NOT NULL
- FOREIGN KEY (`Customer_ID`) REFERENCES `Customer`(`Customer_ID`)

4. Supplier

- `Supplier_ID` INT PRIMARY KEY
- `Name` VARCHAR(100) NOT NULL
- `Contact_Name` VARCHAR(100)
- `Phone_Number` VARCHAR(20)

5. Warehouse

- `Warehouse_ID` INT PRIMARY KEY
- `Location` VARCHAR(100) NOT NULL
- `Capacity` INT NOT NULL

6. Order_Product

- `Order_ID` INT NOT NULL
- `Product_ID` INT NOT NULL
- `Quantity` INT NOT NULL
- PRIMARY KEY (`Order_ID`, `Product_ID`)
- FOREIGN KEY (`Order_ID`) REFERENCES `Order`(`Order_ID`)
- FOREIGN KEY (`Product_ID`) REFERENCES `Product`(`Product_ID`)

7. Product_Supplier

- `Product_ID` INT NOT NULL
- `Supplier_ID` INT NOT NULL
- PRIMARY KEY (`Product_ID`, `Supplier_ID`)
- FOREIGN KEY (`Product_ID`) REFERENCES `Product`(`Product_ID`)
- FOREIGN KEY (`Supplier_ID`) REFERENCES `Supplier`(`Supplier_ID`)

8. Warehouse_Product

- `Warehouse_ID` INT NOT NULL
- `Product_ID` INT NOT NULL
- `Quantity_Stored` INT NOT NULL
- PRIMARY KEY (`Warehouse_ID`, `Product_ID`)
- FOREIGN KEY (`Warehouse_ID`) REFERENCES `Warehouse`(`Warehouse_ID`)
- FOREIGN KEY (`Product_ID`) REFERENCES `Product`(`Product_ID`)

4.2 SQL Statements for Table Creation and Data Insertion

The following SQL statements create the necessary tables and insert sample data into each table.

```
-- Create Database
```

```
CREATE DATABASE OnlineRetailStore;
```

```
USE OnlineRetailStore;
```

```
-- Create Customer Table
```

```
CREATE TABLE Customer (
```

```
    Customer_ID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    Name VARCHAR(100) NOT NULL,
```

```
    Email VARCHAR(100) NOT NULL,
```

```
    Phone_Number VARCHAR(20)
```

```
);
```

```
-- Insert Sample Data into Customer Table
```

```
INSERT INTO Customer (Name, Email, Phone_Number) VALUES
```

```
('Alice Johnson', 'alice.johnson@example.com', '555-1234'),
```

```
('Bob Smith', 'bob.smith@example.com', '555-5678'),
```

```
('Charlie Brown', 'charlie.brown@example.com', '555-1234'); -- Shared phone number
```

```
-- Create Product Table
```

```
CREATE TABLE Product (
```

```
    Product_ID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    Name VARCHAR(100) NOT NULL,
```

```
    Description TEXT,
```

```
    Price DECIMAL(10,2) NOT NULL
```

```
);
```

-- Insert Sample Data into Product Table

```
INSERT INTO Product (Name, Description, Price) VALUES  
( 'Laptop', 'High-performance laptop with 16GB RAM', 1200.00),  
( 'Smartphone', 'Latest model smartphone with OLED display', 800.00),  
( 'Headphones', 'Noise-cancelling over-ear headphones', 200.00);
```

-- Create Order Table

```
CREATE TABLE `Order` (  
    Order_ID INT PRIMARY KEY AUTO_INCREMENT,  
    Order_Date DATE NOT NULL,  
    Total_Amount DECIMAL(10,2) NOT NULL,  
    Customer_ID INT NOT NULL,  
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)  
);
```

-- Insert Sample Data into Order Table

```
INSERT INTO `Order` (Order_Date, Total_Amount, Customer_ID) VALUES  
( '2024-08-15', 1400.00, 1),  
( '2024-08-16', 800.00, 2),  
( '2024-08-17', 200.00, 3);
```

-- Create Supplier Table

```
CREATE TABLE Supplier (  
    Supplier_ID INT PRIMARY KEY AUTO_INCREMENT,  
    Name VARCHAR(100) NOT NULL,  
    Contact_Name VARCHAR(100),  
    Phone_Number VARCHAR(20)  
);
```

-- Insert Sample Data into Supplier Table

```
INSERT INTO Supplier (Name, Contact_Name, Phone_Number) VALUES
('TechSupplier Inc.', 'David Lee', '555-8765'),
('GadgetWorld', 'Emma Davis', '555-4321');
```

-- Create Warehouse Table

```
CREATE TABLE Warehouse (
    Warehouse_ID INT PRIMARY KEY AUTO_INCREMENT,
    Location VARCHAR(100) NOT NULL,
    Capacity INT NOT NULL
);
```

-- Insert Sample Data into Warehouse Table

```
INSERT INTO Warehouse (Location, Capacity) VALUES
('New York', 5000),
('Los Angeles', 3000);
```

-- Create Order_Product Table

```
CREATE TABLE Order_Product (
    Order_ID INT NOT NULL,
    Product_ID INT NOT NULL,
    Quantity INT NOT NULL,
    PRIMARY KEY (Order_ID, Product_ID),
    FOREIGN KEY (Order_ID) REFERENCES `Order`(Order_ID),
    FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);
```

-- Insert Sample Data into Order_Product Table

```
INSERT INTO Order_Product (Order_ID, Product_ID, Quantity) VALUES
(1, 1, 1), -- Order 1: 1 Laptop
```

(1, 3, 2), -- Order 1: 2 Headphones

(2, 2, 1), -- Order 2: 1 Smartphone

(3, 3, 1); -- Order 3: 1 Headphones

-- Create Product_Supplier Table

```
CREATE TABLE Product_Supplier (  
    Product_ID INT NOT NULL,  
    Supplier_ID INT NOT NULL,  
    PRIMARY KEY (Product_ID, Supplier_ID),  
    FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),  
    FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID)  
);
```

-- Insert Sample Data into Product_Supplier Table

```
INSERT INTO Product_Supplier (Product_ID, Supplier_ID) VALUES  
(1, 1), -- Laptop supplied by TechSupplier Inc.  
(2, 1), -- Smartphone supplied by TechSupplier Inc.  
(3, 2); -- Headphones supplied by GadgetWorld
```

-- Create Warehouse_Product Table

```
CREATE TABLE Warehouse_Product (  
    Warehouse_ID INT NOT NULL,  
    Product_ID INT NOT NULL,  
    Quantity_Stored INT NOT NULL,  
    PRIMARY KEY (Warehouse_ID, Product_ID),  
    FOREIGN KEY (Warehouse_ID) REFERENCES Warehouse(Warehouse_ID),  
    FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)  
);
```

-- Insert Sample Data into Warehouse_Product Table

```
INSERT INTO Warehouse_Product (Warehouse_ID, Product_ID, Quantity_Stored)
VALUES
```

```
(1, 1, 50), -- New York Warehouse: 50 Laptops
```

```
(1, 2, 30), -- New York Warehouse: 30 Smartphones
```

```
(2, 3, 100); -- Los Angeles Warehouse: 100 Headphones
```

```
````
```

### **Explanation:**

#### 1. Database Creation:

- The `OnlineRetailStore` database is created and selected for subsequent operations.

#### 2. Customer Table:

- The `Customer` table includes `Customer\_ID` as the primary key.
- Sample customers are inserted, including customers sharing the same phone number.

#### 3. Product Table:

- The `Product` table includes `Product\_ID` as the primary key.
- Sample products such as Laptop, Smartphone, and Headphones are inserted.

#### 4. Order Table:

- The `Order` table includes `Order\_ID` as the primary key.
- Each order is linked to a customer via `Customer\_ID`.
- Sample orders are inserted.

#### 5. Supplier Table:

- The `Supplier` table includes `Supplier\_ID` as the primary key.
- Sample suppliers are inserted.

#### 6. Warehouse Table:

- The `Warehouse` table includes `Warehouse\_ID` as the primary key.
- Sample warehouses are inserted.

#### 7. Order\_Product Table:

- The `Order\_Product` table represents the many-to-many relationship between orders and products.
- It includes a composite primary key of `Order\_ID` and `Product\_ID`.
- Sample order-product associations are inserted.

#### 8. Product\_Supplier Table:

- The `Product\_Supplier` table represents the many-to-many relationship between products and suppliers.
- It includes a composite primary key of `Product\_ID` and `Supplier\_ID`.
- Sample product-supplier associations are inserted.

#### 9. Warehouse\_Product Table:

- The `Warehouse\_Product` table represents the many-to-many relationship between warehouses and products.
- It includes a composite primary key of `Warehouse\_ID` and `Product\_ID`.
- Sample warehouse-product associations are inserted.

## 5. Conclusion

The design and implementation of a relational database for an online retail store have been meticulously detailed in this report. By identifying key entities—Customer, Product, Order, Supplier, and Warehouse—and establishing the relationships between them, the database schema effectively captures the essential aspects of the store's operations. The use of associative entities such as Order\_Product, Product\_Supplier, and Warehouse\_Product facilitates the management of many-to-many relationships, ensuring data integrity and reducing redundancy.

Normalization principles have been applied to achieve an optimal database structure, enhancing data consistency and efficiency. The provided SQL statements for table creation and data insertion serve as a practical foundation for deploying the database in a real-world environment. This structured approach not only streamlines data management but also supports scalable growth, allowing the retail store to handle increasing volumes of transactions and inventory with ease.

Overall, the proposed database system offers a robust solution for managing the complexities of an online retail operation, ensuring seamless data flow, accurate record-keeping, and efficient resource utilization.



## 6. References

1. GeeksforGeeks. (2023). RDBMS. Retrieved from [\[https://www.geeksforgeeks.org/rdbms/\]](https://www.geeksforgeeks.org/rdbms/)(<https://www.geeksforgeeks.org/rdbms/>)
2. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.
3. Coronel, C., & Morris, S. (2016). *Database Systems: Design, Implementation, & Management* (11th ed.). Cengage Learning.
4. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.
5. W3Schools. (2023). SQL Tutorial. Retrieved from [\[https://www.w3schools.com/sql/\]](https://www.w3schools.com/sql/)(<https://www.w3schools.com/sql/>)
6. Stack Overflow. (2023). Best Practices for Database Design. Retrieved from [\[https://stackoverflow.com/questions/1622001/best-practices-for-database-design\]](https://stackoverflow.com/questions/1622001/best-practices-for-database-design)(<https://stackoverflow.com/questions/1622001/best-practices-for-database-design>)
7. MySQL Documentation. (2023). Retrieved from [\[https://dev.mysql.com/doc/\]](https://dev.mysql.com/doc/)(<https://dev.mysql.com/doc/>)

A FIELD PROJECT/IDP REPORT ON

**ER to relational schema**

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

MEKA CHANDANA REDDY (221FA19002)

GUDIVADA KRISHNA PRAKASH (221FA19027)

JUVVA DIVYA (221FA19028)

DEVABHAKTUNI AMAR SAI  
CHOWDARY (221FA19046)



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**Department of ACSE**

**School of Computing and Informatics**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH  
(Deemed to be University)**

**Vadlamudi, Guntur, Andhra Pradesh-522213, India**

**April, 2024**



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

Est. in 1956

### CERTIFICATE

This is to certify that the field project/IDP entitled "ER to relational schema" being submitted by MEKA CHANDANA REDDY-221FA19002, GUDIVADA KRISHNA PRAKASH-221FA19027, JUVVADIVYA-221FA19028, DEVABHAKTUNIAMARSAI CHOWDARY-221FA19046 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Guide

Dr. Venkatesulu Dondeti

HoD/ACSE



## DECLARATION

We hereby declare that our project work described in the field project titled “ER to relational schema” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

|                                   |            |
|-----------------------------------|------------|
| MEKA CHANDANA REDDY               | 221FA19002 |
| GUDIVADA KRISHNA PRAKASH          | 221FA19027 |
| JUVVA DIVYA                       | 221FA19028 |
| DEVABHAKTUNI AMAR SAI<br>CHOWDARY | 221FA19046 |

## **Abstract**

This report focuses on converting an Entity-Relationship (ER) model for a company database into a relational schema. The ER model describes the structure of the company's data, organized into entities such as Department, Employee, Project, and Dependent. Each entity is defined by its attributes, with relationships linking them to maintain data integrity and ensure efficient data management.

The company consists of multiple departments, each managed by a manager and supervised by supervisors. Employees work in these departments and can be assigned to multiple projects simultaneously. The system also keeps track of the work hours of employees for each project. Additionally, the database manages the dependents of employees, storing details like name, date of birth, gender, and relationship type. Multi-valued attributes, such as skills for employees, are also handled.

The process of converting the ER model to a relational schema involves creating tables for each entity and defining primary keys for unique identification. Relationships are represented using foreign keys, linking related tables to maintain referential integrity. The conversion ensures that multi-valued attributes, like employee skills, are handled by creating separate tables or using composite keys to accommodate multiple values for a single attribute. This relational schema supports the efficient querying and management of company data, allowing for quick access to employee, project, department, and dependent information while maintaining consistency.

By implementing the relational schema derived from the ER model, the company database can efficiently manage complex relationships and large amounts of data while ensuring data integrity. The process demonstrates the effectiveness of converting conceptual models into relational structures, which are fundamental for building and managing robust databases in an organizational environment.

## **Table of Contents**

1. Introduction
2. Definitions
  - 2.1 Entity
  - 2.2 Attribute
  - 2.3 Relation
3. Problem Definition
4. ER Diagram
5. Converting ER to Relational Schema
6. Relational Schema Representation
7. References

## 1. Introduction

A database is a structured collection of data stored and managed efficiently for easy retrieval and manipulation. The database serves as a digital filing system, organizing information into tables and ensuring it can be easily located and used. In this document, we explore a company's structure through an ER diagram and convert it into a relational schema.

In modern organizations, managing large volumes of data efficiently is essential. Databases play a critical role in organizing, storing, and retrieving this information effectively. The Entity-Relationship (ER) model serves as a foundational tool for designing databases by visually representing entities (such as departments, employees, projects, and dependents) and their relationships. It allows database designers to create a blueprint of how data interacts within a system, ensuring clarity and consistency before the physical implementation of the database.

This report focuses on converting an ER model for a company's database into a relational schema, which is essential for building a functional and efficient database system. The ER model outlines key entities such as departments, employees, projects, and dependents, along with their respective attributes and relationships. The company is structured into multiple departments, each managed and supervised by specific roles, while employees can be assigned to various projects. The system also records work hours and maintains information about employees' dependents.

The process of converting the ER model into a relational schema involves creating tables that correspond to the entities, assigning primary keys for unique identification, and establishing foreign keys to maintain relationships between these tables. This conversion is crucial as it transitions the conceptual model into a physical structure that can be implemented using a relational database management system (RDBMS). By ensuring that relationships and constraints are clearly defined, the relational schema supports data integrity, efficiency in query performance, and scalability for future expansions.

## 2. Definitions

### 2.1 Entity

An entity represents a distinct object or concept within the system that has an independent existence. Examples include `Employee`, `Department`, `Project`, and `Dependent`.

### 2.2 Attribute

Attributes are the properties or characteristics that define an entity. They form the columns in a relational database table. For instance, attributes of `Employee` include `eno`, `name`, `dob`, etc.

### 2.3 Relation

A relation refers to a table in a relational database or the relationship between tables. A relation is defined by entities and their attributes, and can also show relationships between entities like `Works\_For` or `Manages`.



### 3. Problem Definition

The company is organized into departments, with employees working in each department. The relationships between employees, departments, projects, and dependents need to be modeled using an ER diagram, which is then converted into a relational schema.

#### Attributes of the Entities:

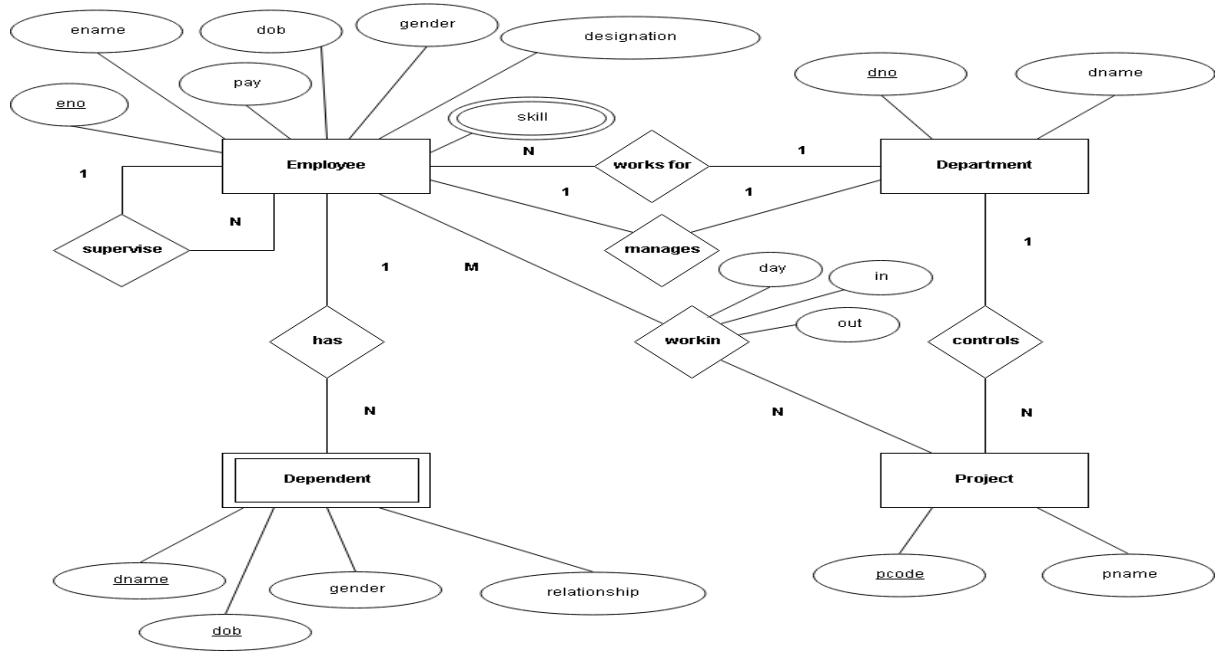
- Department: `dno` (Department Number), `dname` (Department Name)
- Employee: `eno` (Employee Number), `name`, `dob`, `gender`, `doj` (Date of Joining), `designation`, `basic\_pay`, `panno` (PAN Number), `skills` (multi-valued)
- Project: `pcode` (Project Code), `pname` (Project Name)
- Dependent: `dname` (Dependent Name), `dob`, `gender`, `relationship`

#### Relationships:

- Department: Managed by a manager, with supervisors overseeing employees.
- Employee: Can work on multiple projects, and their in-time and out-time are tracked.
- Dependent: Each employee may have dependents with details stored in the database.

## 4. ER Diagram

Below is the ER diagram, showing relationships between `Employee`, `Department`, `Project`, and `Dependent` entities.



## 5. Converting ER to Relational Schema

The process of converting the ER model into a relational schema is the next step in database design. Each entity and its attributes are mapped to a relational schema. Primary keys are underlined, and relationships are represented through foreign keys.

## 6. Relational Schema Representation

This section provides the relational schema representation based on the ER diagram, with all entities and their relationships.

### 1. Department

- Department(`dno` PK, `dname`)
- Each department is uniquely identified by its `dno` and has a `dname`.

### 2. Employee

- Employee(`eno` PK, `name`, `dob`, `gender`, `doj`, `designation`, `basic\_pay`, `panno`)
- Employees have a unique identifier `eno`, and each employee has attributes like name, date of birth, gender, date of joining, and more.

### 3. Skills (Multi-valued attribute of Employee)

- Emp\_Skills(`eno` FK, `skill`)
- Since `skills` is a multi-valued attribute, a separate table is created for `Emp\_Skills` to store multiple skills per employee.

### 4. Project

- Project(`pcode` PK, `pname`, `dno` FK)
- Projects are controlled by departments, and each project has a project code `pcode`, project name `pname`, and is associated with a department `dno`.

## 5. Work-In (Work Record for Employees on Projects)

- Work\_In(`eno` FK, `pcode` FK, `work\_date`, `in\_time`, `out\_time`)
- This table tracks the in-time and out-time for employees working on projects on specific dates.

## 6. Dependent

- Dependent(`eno` FK, `dname`, `dob`, `gender`, `relationship`)
- This table stores the details of dependents of employees, including the employee's ID as a foreign key.

## Schema Diagram Representation

Below is a schema diagram based on the ER model:

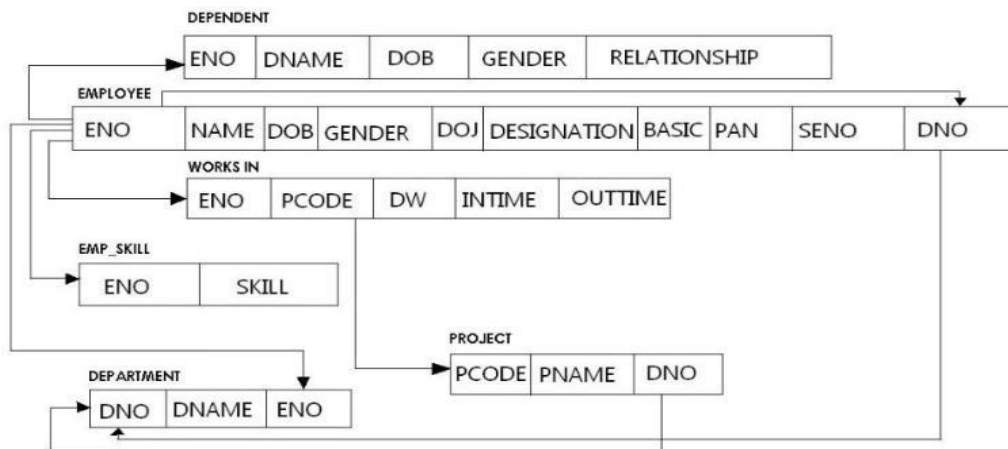
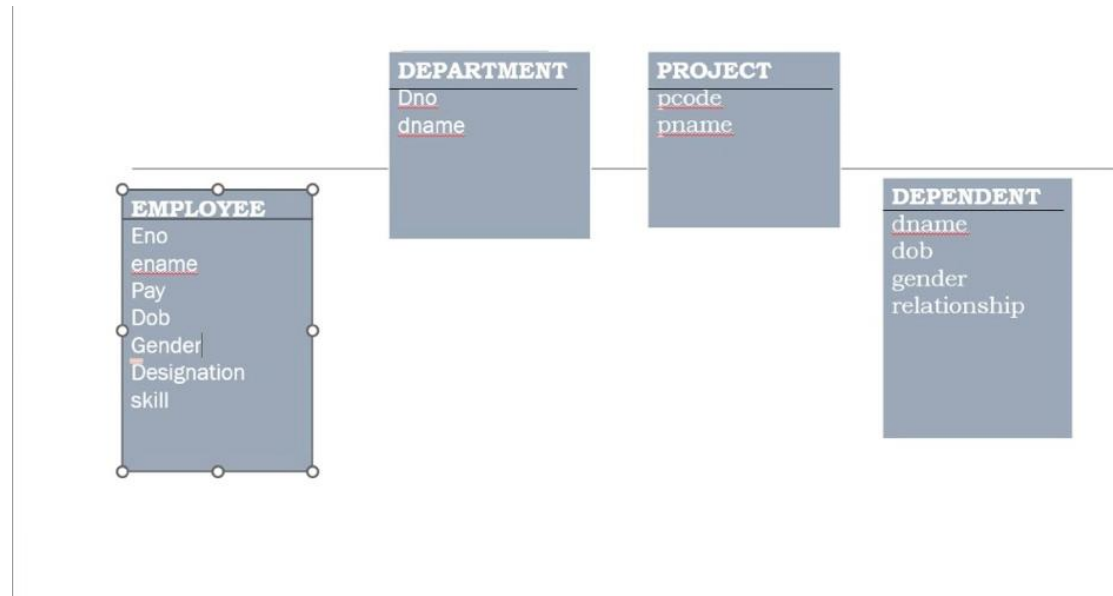


Figure 2 – Relational Model of Database

## Alternate Schema Representation (Tables)



## 7.References

1. Elmasri, R., & Navathe, S. B. (2017). *Fundamentals of Database Systems*. Pearson.
2. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts*. McGraw-Hill.
3. Connolly, T., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Addison-Wesley.
4. Codd, E. F. (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 13(6), 377-387.
5. MySQL Documentation: <https://dev.mysql.com/doc/>
6. ER Diagrams and Relational Model Conversion Guidelines:  
[https://www.tutorialspoint.com/dbms/dbms\\_er\\_diagram\\_representation.htm](https://www.tutorialspoint.com/dbms/dbms_er_diagram_representation.htm)

A FIELD PROJECT/IDP REPORT ON

**ONLINE RETAIL STORE DATABASE**

Submitted partial fulfilment of the requirements for the award of the degree

**BACHELOR OF TECHNOLOGY**

in

**CYBER SECURITY**

Submitted by

|                             |              |
|-----------------------------|--------------|
| LAMBU DAMARUKANATH          | (221FA19008) |
| MULLA JUNAID RAHMAN         | (221FA19010) |
| DHULIPALLA MANASA           | (221FA19032) |
| YARRAM VENKATA<br>VAMSIDHAR | (221FA19059) |



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**Department of ACSE**

**School of Computing and Informatics**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH**

**(Deemed to be University)**

**Vadlamudi, Guntur, Andhra Pradesh-522213, India**

**April, 2024**



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

### CERTIFICATE

This is to certify that the field project/IDP entitled "ONLINE RETAIL STORE DATABASE" being submitted by LAMBU DAMARUKANATH-221FA19008, MULLA JUNAID RAHMAN-221FA19010, DHULIPALLA MANASA-221FA19032, YARRAM VENKATA VAMSIDHAR- 221FA19059 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Guide :

HOD/ACSE

Dr. Venkatesulu Dondeti





**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

## **DECLARATION**

We hereby declare that our project work described in the field project titled “ONLINE RETAIL STORE DATABASE” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

|                          |            |
|--------------------------|------------|
| LAMBU DAMARUKANATH       | 221FA19008 |
| MULLA JUNAID RAHMAN      | 221FA19010 |
| DHULIPALLA MANASA        | 221FA19032 |
| YARRAM VENKATA VAMSIDHAR | 221FA19059 |

## **Abstract**

The database design for an online retail store is structured to efficiently handle various core entities: customers, products, orders, suppliers, and warehouses. Each customer has a unique ID for identification while allowing flexibility in sharing phone numbers. Products are cataloged by unique IDs, descriptions, and prices, and can be included in multiple orders, reflecting typical purchasing behavior. Orders are linked to individual customers but can contain many different products. Additionally, suppliers and products have a many-to-many relationship, meaning one product can be supplied by multiple suppliers, and one supplier can provide multiple products. Warehouses are managed by tracking their locations and capacities, and each warehouse can store many different products. This relational database ensures that these interconnected relationships are well managed, allowing for seamless operations within the store.

# Contents

1. Introduction
2. Problem Statement
3. System Overview
4. Requirements Specification
  - 4.1 Customer Management
  - 4.2 Product Management
  - 4.3 Order Management
  - 4.4 Supplier Management
  - 4.5 Warehouse Management
5. Design Considerations
  - 5.1 Structured Data
  - 5.2 Query Optimization
  - 5.3 Concurrency and Transactions
  - 5.4 Reporting Capabilities
6. Extension Work
7. Conclusion
8. References

## Introduction

The database design for an online retail store plays a pivotal role in managing the key entities required to facilitate smooth operations. Among the core entities are customers, products, orders, suppliers, and warehouses. Each customer has unique attributes, such as a customer ID and contact details, and can place multiple orders over time. Each order can contain several products, reflecting the customer's purchase history and preferences. The relational structure ensures that customer data is accurately stored, allowing for efficient order tracking and personalized service offerings, such as recommendations or loyalty rewards.

Products, another essential entity in the system, are uniquely identified by product IDs and can be linked to multiple suppliers. This enables the store to maintain diverse supplier relationships, ensuring that inventory is consistently replenished and product availability is maximized. The system also tracks stock levels in various warehouses, helping the store to efficiently manage its inventory. By recording which products are stored in which warehouse, the database supports better logistics, optimizing order fulfillment and delivery processes. Products can also be linked to various attributes, such as category, price, and stock levels, allowing the store to update information as needed and offer customers the latest product details.

Furthermore, the database supports supplier and warehouse management, ensuring that relationships with multiple suppliers are properly maintained and that stock can be efficiently distributed across different storage facilities. This design ensures that the retail store can effectively monitor inventory, manage reorders, and streamline the supply chain. By leveraging relational database principles, the system is built to scale with the business, offering real-time insights into product availability and supplier performance while supporting the store's daily operations with minimal manual intervention.

## **Problem Statement**

An online retail store wants to design a database to manage its operations efficiently. The system must manage customers, products, orders, suppliers, and warehouses, with the following details:

- Customer: Each customer has a customer ID, name, email, and phone number. Customers may share the same phone number.
- Product: Each product has a product ID, name, description, and price.
- Order: Each order has an order ID, order date, and total amount. Each order is associated with one customer, but each order can contain multiple products, and each product can appear in multiple orders.
- Supplier: Each supplier has a supplier ID, name, contact name, and phone number. Suppliers can supply multiple products, and each product can be supplied by multiple suppliers.
- Warehouse: Each warehouse has a warehouse ID, location, and capacity. Products are stored in warehouses, and each warehouse can store multiple products.

## System Overview

The system manages the core functionalities of an online retail store:

- Customer Management: Captures customer details such as name, contact info, and ID.
- Product Management: Manages product inventory, tracking product IDs, descriptions, prices, and supplier details.
- Order Management: Allows orders to be associated with customers and linked to products.
- Supplier Management: Maintains supplier information and their product offerings.
- Warehouse Management: Tracks product storage in different warehouses by location and capacity.

This system ensures seamless order processing, supplier relationships, and inventory management, essential for efficient retail operations.

## Requirements Specification

### 4.1 Customer Management:

- Store customer details, including customer ID, name, email, and phone number.
- Allow multiple customers to share the same phone number, as customers may register with the same contact details.

### 4.2 Product Management:

- Maintain product information, including unique product IDs, names, descriptions, and prices.
- Products can be supplied by multiple suppliers.

### 4.3 Order Management:

- Track orders using unique order IDs, order dates, and total amounts.
- Each order is linked to one customer.
- An order can contain multiple products.

### 4.4 Supplier Management:

- Record supplier details, including supplier ID, name, contact person, and phone number.
- A supplier can provide multiple products, and each product may have multiple suppliers.

### 4.5 Warehouse Management:

- Manage warehouse data, including warehouse ID, location, and capacity.
- Track products stored in each warehouse.

## Design Considerations

### 5.1 Structured Data:

A database management system (DBMS) organizes data into structured tables with well-defined relationships, making it easier and faster to query specific information. Unlike flat files, a DBMS provides faster access due to its relational structure and indexing capabilities.

### 5.2 Query Optimization:

DBMSs use techniques such as indexing, caching, and query optimization to enhance performance, allowing quick retrieval of complex data. In contrast, file systems require slower sequential searches to locate data.

### 5.3 Concurrency and Transactions:

DBMSs provide concurrency control, ensuring that multiple users can access and modify data simultaneously without compromising data integrity. This feature is essential in an online retail store where many users may interact with the system at the same time.

### 5.4 Reporting Capabilities:

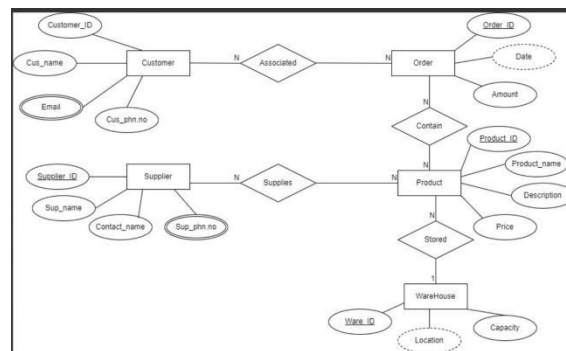
A DBMS simplifies the generation of complex reports using SQL queries that can aggregate data from different tables. In contrast, file systems would require manual processing to generate reports, making it less efficient.



## Extension Work

Here is an illustration of the database management system for the online retail store:

ER-Diagram of the Database Management System for an Online Retail Store (This will include the core entities: Customer, Product, Order, Supplier, and Warehouse).



## Relational Model

The relational model for the online retail store is illustrated as follows:

1. Customer(`CustomerID`, `Name`, `Email`, `PhoneNumber`)
2. Product(`ProductID`, `Name`, `Description`, `Price`)
3. Order(`OrderID`, `OrderDate`, `TotalAmount`, `CustomerID`)
4. OrderProduct(`OrderID`, `ProductID`, `Quantity`)
5. Supplier(`SupplierID`, `Name`, `ContactName`, `PhoneNumber`)
6. SupplierProduct(`SupplierID`, `ProductID`)
7. Warehouse(`WarehouseID`, `Location`, `Capacity`)
8. WarehouseProduct(`WarehouseID`, `ProductID`)

| order_id | product_id | quantity |
|----------|------------|----------|
| 1111     | 111        | 4        |
| 2222     | 222        | 3        |
| 3333     | 333        | 2        |
| 4444     | 444        | 1        |
| 5555     | 555        | 3        |
| NULL     | NULL       | NULL     |

| order_id | order_date | total_amount | customer_id |
|----------|------------|--------------|-------------|
| 1111     | 2024-10-07 | 49999.00     | 1           |
| 2222     | 2024-09-03 | 4987.00      | 2           |
| 3333     | 2024-10-04 | 6500.00      | 3           |
| 4444     | 2024-02-06 | 36000.00     | 4           |
| 5555     | 2024-09-07 | 17500.00     | 5           |
| NULL     | NULL       | NULL         | NULL        |




| supplier_id | name  | contact_name | phone_number |
|-------------|-------|--------------|--------------|
| 11111       | sai   | 221c6        | 123654       |
| 22222       | eswar | 221c7        | 123754       |
| 33333       | hari  | 221c8        | 124654       |
| 44444       | john  | 221c9        | 113654       |
| 55555       | jack  | 221c10       | 133654       |
| NULL        | NULL  | NULL         | NULL         |

| warehouse_id | location                  | capacity |
|--------------|---------------------------|----------|
| 11           | raiwatcolony 8 - line     | 125      |
| 22           | teachers colony 4 - line  | 1256     |
| 33           | reddycolony 7 - line      | 1125     |
| 44           | puttam street dbno: 456/1 | 1325     |
| 55           | raiwatcolony 9 - line     | 9125     |
| NULL         | NULL                      | NULL     |




Result Grid | Filter Rows:

|   | product_id | supplier_id |
|---|------------|-------------|
| ▶ | 111        | 11111       |
|   | 222        | 22222       |
|   | 333        | 33333       |
|   | 444        | 44444       |
|   | 555        | 55555       |

## My SQL QUERIES:

Result Grid | Filter Rows:  | Edit:   

| customer_id | name    | email             | phone_number |
|-------------|---------|-------------------|--------------|
| ▶ 1         | naveen  | naveen@gmail.com  | 1234567      |
| 2           | navya   | navya@gmail.com   | 1234908      |
| 3           | teja    | teja@gmail.com    | 7476889      |
| 4           | ramya   | ramya@gmail.com   | 679943       |
| 5           | pallavi | pallavi@gmail.com | 687386       |
| ★ NULL      | NULL    | NULL              | NULL         |

Result Grid | Filter Rows:  | Edit:    | Ex

| product_id | name     | description                      | price    |
|------------|----------|----------------------------------|----------|
| ▶ 111      | laptop   | i 5 generation new model         | 49999.00 |
| 222        | mouse    | wireless smooth to operate       | 600.00   |
| 333        | keyboard | wireless , glass cover           | 6000.00  |
| 444        | phone    | one plus 10 R. 50000 mah battery | 35000.00 |
| 555        | cpu      | ram speed processor              | 17000.00 |
| ★ NULL     | NULL     | NULL                             | NULL     |

```

1 • CREATE DATABASE online_retail_store2;
2 • USE online_retail_store2;
3 • CREATE TABLE customers (
4 customer_id INT AUTO_INCREMENT,
5 name VARCHAR(255) NOT NULL,
6 email VARCHAR(255) NOT NULL,
7 phone_number VARCHAR(20),
8 PRIMARY KEY (customer_id)
9);
10 • insert into customers values('1','naveen','naveen@gmail.com','1234567');
11 • insert into customers values('2','navya','navya@gmail.com','1234908');
12 • insert into customers values('3','teja','teja@gmail.com','7476889');
13 • insert into customers values('4','ramya','ramya@gmail.com','679943');
14 • insert into customers values('5','pallavi','pallavi@gmail.com','687386');
15 • select * from customers;
16 • CREATE TABLE products1 (
17 product_id INT AUTO_INCREMENT,
18 name VARCHAR(255) NOT NULL,
19 description TEXT,
20 price DECIMAL(10, 2) NOT NULL,

```

```

PRIMARY KEY (product_id)
);
23 • insert into products1 values('111','laptop','i 5 generation new model','49999');
24 • insert into products1 values('222','mouse','wireless smooth to operate','600');
25 • insert into products1 values('333','keyboard','wireless , glass cover','6000');
26 • insert into products1 values('444','phone','one plus 10 R 50000 mah battery','35000');
27 • insert into products1 values('555','cpu','ram speed processor','17000');
28 • select * from products1;
29 • CREATE TABLE orders (
30 order_id INT AUTO_INCREMENT,
31 order_date DATE NOT NULL,
32 total_amount DECIMAL(10, 2) NOT NULL,
33 customer_id INT,
34 PRIMARY KEY (order_id),
35 FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
36);
37 • insert into orders values('1111','2024/10/07','49999','1');
38 • insert into orders values('2222','2024/09/03','4987','2');
39 • insert into orders values('3333','2024/10/04','6500','3');
40 • insert into orders values('4444','2024/02/06','36000','4');

```

```

41 • insert into orders values('5555','2024/09/07','17500','5');
42 • select * from orders;
43 • desc orders;
44 • CREATE TABLE order_products2 (
45 order_id INT,
46 product_id INT,
47 quantity INT NOT NULL,
48 PRIMARY KEY (order_id, product_id),
49 FOREIGN KEY (order_id) REFERENCES orders(order_id),
50 FOREIGN KEY (product_id) REFERENCES products1(product_id)
51);
52 • insert into order_products2 values('1111','111','4');
53 • insert into order_products2 values('2222','222','3');
54 • insert into order_products2 values('3333','333','2');
55 • insert into order_products2 values('4444','444','1');
56 • insert into order_products2 values('5555','555','3');
57 • select * from order_products2 ;
58 • CREATE TABLE suppliers (
59 supplier_id INT AUTO_INCREMENT,
60 name VARCHAR(255) NOT NULL,

```

```

61 contact_name VARCHAR(255) NOT NULL,
62 phone_number VARCHAR(20),
63 PRIMARY KEY (supplier_id)
64);
65 • insert into suppliers values('11111','sai','221c6','123654');
66 • insert into suppliers values('22222','eswar','221c7','123754');
67 • insert into suppliers values('33333','hari','221c8','124654');
68 • insert into suppliers values('44444','john','221c9','113654');
69 • insert into suppliers values('55555','jack','221c10','133654');
70 • select * from suppliers ;
71 • CREATE TABLE product_suppliers1 (
72 product_id INT,
73 supplier_id INT,
74
75 FOREIGN KEY (product_id) REFERENCES products1(product_id),
76 FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)
77);
78 • insert into product_suppliers1 values('111','11111');
79 • insert into product_suppliers1 values('222','22222');
80 • insert into product_suppliers1 values('333','33333');

```

```

81 • insert into product_suppliers1 values('444','44444');
82 • insert into product_suppliers1 values('555','55555');
83 • select * from product_suppliers1 ;
84 • CREATE TABLE warehouses (
85 warehouse_id INT AUTO_INCREMENT,
86 location VARCHAR(255) NOT NULL,
87 capacity INT NOT NULL,
88 PRIMARY KEY (warehouse_id)
89);
90 • insert into warehouses values('011','raiwatcolony 8 - line',125);
91 • insert into warehouses values('022','teachers colony 4 - line',1256);
92 • insert into warehouses values('033','reddycolony 7 - line',1125);
93 • insert into warehouses values('044','puttam street dbno: 456/1 ',1325);
94 • insert into warehouses values('055','raiwatcolony 9 - line',9125);
95 • select * from warehouses ;
96 • CREATE TABLE product_warehouses (
97 product_id INT,
98 warehouse_id INT,
99 quantity INT NOT NULL,
100 PRIMARY KEY (product_id, warehouse_id),
101
102 FOREIGN KEY (product_id) REFERENCES products(product_id),
103 FOREIGN KEY (warehouse_id) REFERENCES warehouses(warehouse_id)
104);
105 • select * from product_warehouses ;
106

```

## **Conclusion**

The online retail store's database is designed to efficiently manage various aspects of the store's operations, including customer information, product inventory, order processing, supplier relationships, and warehouse storage. By organizing data into structured tables with defined relationships, the system ensures seamless operations, allowing the store to handle complex queries, maintain data integrity, and generate detailed reports.

## References

1. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*. Pearson.
2. Date, C. J. (2004). *An Introduction to Database Systems*. Addison-Wesley.
3. Connolly, T. M., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson.
4. Rob, P., & Coronel, C. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.
5. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
6. Korth, H. F., & Silberschatz, A. (2002). *Database System Concepts*. McGraw-Hill.
7. O'Neil, P., & O'Neil, E. (2009). *Database: Principles, Programming, and Performance*. Morgan Kaufmann.
8. Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, 13(6), 377-387.
9. Chen, P. P. (1976). "The Entity-Relationship Model—Toward a Unified View of Data." *ACM Transactions on Database Systems*, 1(1), 9-36.

A FIELD PROJECT/IDP REPORT ON

**ER DIAGRAM to RELATIONAL MODEL**

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

PALLA MAHESWARARAO (221FA19016)

POTLA AJAY (221FA19037)

YADLAPALLI VISHNU  
VARDHAN (221FA19053)

LAVU UHA SARANYA (221FA19063)



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**Department of ACSE**

**School of Computing and Informatics**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH  
(Deemed to be University)**

**Vadlamudi, Guntur, Andhra Pradesh-522213, India**

**April, 2024**





**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**CERTIFICATE**

This is to certify that the field project/IDP entitled "ER DIAGRAM to Relational MODEL" being submitted by PALLA MAHESWARA RAO-221FA19016, POTLA AJAY-221FA19037, YADLAPALLI VISHNU VARDHAN-221FA19053, LAVU UHA SARANYA-221FA19063 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignan's Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

**Guide**

**Dr. Venkatesulu Dondeti**

**HoD/ACSE**



**VIGNAN'S**  
Foundation for Science, Technology & Research  
(Deemed to be University)  
-Estd. u/s 3 of UGC Act 1956

## **DECLARATION**

We hereby declare that our project work described in the field project titled “ER DIAGRAM to Relational MODEL” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

|                              |            |
|------------------------------|------------|
| PALLA MAHESWARARAO           | 221FA19016 |
| POTLA AJAY                   | 221FA19037 |
| YADLAPALLI VISHNU<br>VARDHAN | 221FA19053 |
| LAVU UHA SARANYA             | 221FA19063 |

## **Abstract**

The Delivery Management System database is designed to efficiently manage a range of key entities involved in the delivery process, including delivery items, destinations, schedules, vehicles, and final delivery routes. This relational database structure ensures that each delivery item is tracked using a unique identification number, along with essential details such as weight, dimensions, insurance coverage, and delivery priority. By capturing comprehensive item information, the system can optimize logistical planning, ensuring that the right resources are allocated for each delivery based on item characteristics and urgency. Additionally, the system tracks the final destination of each item, ensuring that delivery locations are accurately recorded and managed.

The database also integrates delivery schedules to monitor deliveries in progress and link items to their respective delivery vehicles and routes. By maintaining up-to-date schedules, the system can adjust delivery times based on real-time conditions, such as traffic or vehicle availability, minimizing delays and enhancing delivery accuracy. Each schedule is tied to a specific delivery route, which is optimized for efficient travel between multiple destinations. This helps reduce fuel costs, improve delivery time, and ensure that items reach their destinations in the most efficient manner possible. The system supports dynamic updates, allowing route modifications in response to last-minute changes in delivery priorities or unforeseen delays.

In addition, the system manages relationships between delivery items, vehicles, and personnel, ensuring smooth coordination across all stages of the delivery process. Vehicles are assigned to specific routes based on capacity, distance, and availability, while drivers are linked to scheduled deliveries, allowing the system to track driver performance and availability. By leveraging this comprehensive relational database design, the Delivery Management System enhances the organization's ability to manage large volumes of delivery items, streamline operations, and provide customers with accurate, real-time tracking and updates, ultimately improving the overall efficiency and reliability of the delivery process.

# CONTENTS

1. Introduction
2. Problem Statement
3. System Overview
4. Requirements Specification
5. Design Considerations
6. Database Structure
  - Delivery Item Table
  - Destination Details Table
  - Schedule and Delivery Table
7. Conclusion
8. References

## INTRODUCTION

The delivery management system is designed to streamline and optimize the tracking and management of delivery items, ensuring seamless logistics operations. Each delivery item is meticulously documented with a unique identification number, along with essential attributes such as weight, dimensions, insurance status, and final destination. This detailed item tracking enables the system to handle diverse delivery requirements, ensuring that each package is transported under the appropriate conditions, whether it requires special handling due to size, fragility, or insurance coverage. By capturing this crucial data, the system ensures that each item's journey from origin to destination is efficiently monitored, allowing for proactive issue resolution and real-time updates.

In addition to item tracking, the system manages multiple delivery routes and schedules, providing end-to-end visibility of items in transit. By associating delivery items with specific routes and schedules, the system can accurately track their location and status throughout the delivery process. This integration allows for dynamic route optimization, adapting to real-time factors such as traffic conditions, weather disruptions, or delivery prioritization. Schedules are linked to each delivery route, ensuring that all items are delivered within the stipulated time frames, enhancing the overall efficiency and reliability of the system. This routing flexibility also improves resource utilization by ensuring that delivery vehicles are assigned optimal routes based on load capacity and delivery priorities.

The system's robust database design ensures efficient data handling and retrieval, supporting the management of large volumes of delivery items, routes, and schedules without compromising performance. By maintaining an organized and scalable structure, the system can handle increased demand, such as peak delivery periods or expanding business operations, while ensuring data accuracy and operational transparency. The design provides high reliability, minimizing the risk of lost or misrouted items, and enhancing customer satisfaction by offering accurate delivery timelines and real-time tracking capabilities. This comprehensive approach ensures that delivery logistics are managed with precision, contributing to the overall success of the business's supply chain operations.

## **PROBLEM STATEMENT**

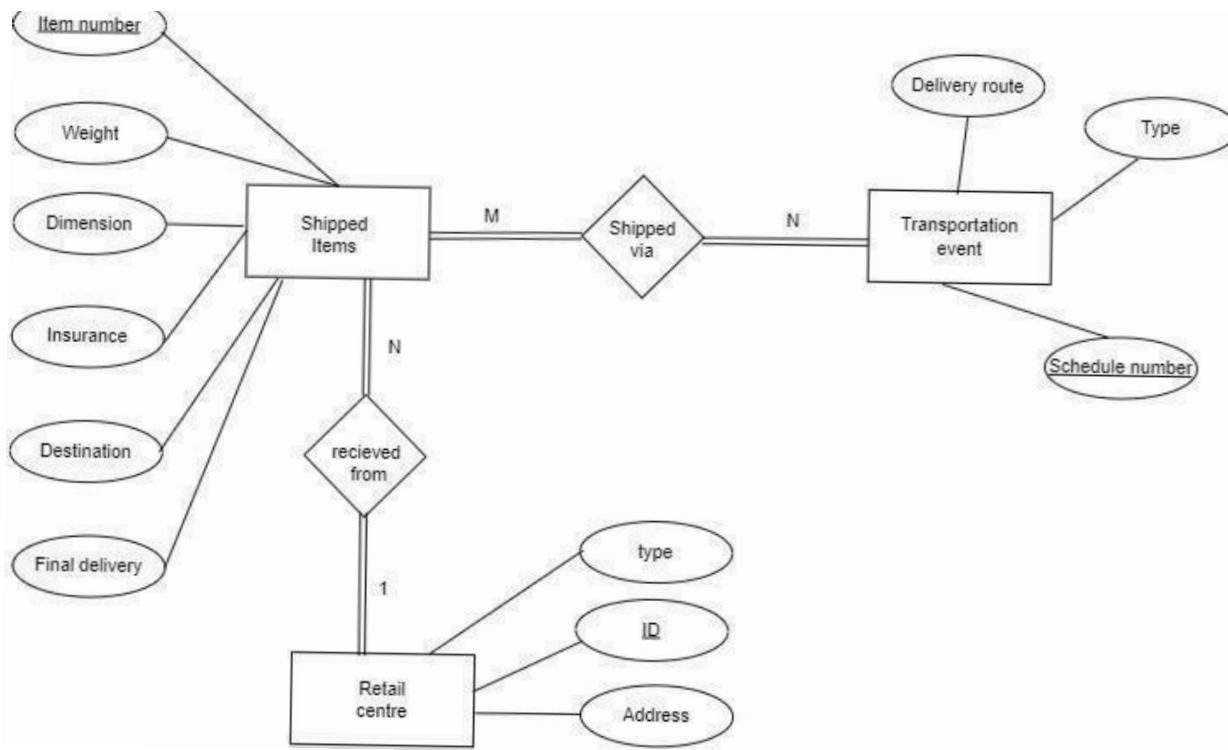
To manage and monitor the deliveries of items effectively, a relational database is needed to:

- Track delivery items by item number, weight, dimension, insurance, and final destination.
- Maintain records of delivery schedules and routes.
- Allow cross-referencing between delivery items and their final destination or route.
- Ensure data integrity and accuracy for seamless logistics operations.

## SYSTEM OVERVIEW

The system focuses on:

1. Delivery Item Management: Storing details about each delivery item, such as its dimensions, weight, insurance status, and destination.
2. Destination Management: Maintaining the location of delivery items and their final destination address.
3. Schedule Management: Tracking the delivery schedule and route that each item follows.
4. Delivery Route Tracking: Mapping items to specific delivery routes and schedules.



## REQUIREMENTS SPECIFICATION

### 1. Delivery Item Management:

- Each item should have a unique item number.
- The system must store attributes such as weight, dimensions, insurance, and destination.

### 2. Destination Details:

- Record the destination for each item, including address and delivery type.

### 3. Schedule Management:

- Maintain delivery schedules by linking them to delivery routes and items.

### 4. Relationships:

- Each delivery item must be mapped to its delivery schedule and route.
- Maintain the relationship between destinations and items for accurate delivery.



## **DESIGN CONSIDERATIONS**

### **1. Data Organization:**

- Use structured tables to represent delivery items, schedules, and destinations.

### **2. Data Integrity:**

- Ensure that data is not duplicated or lost during operations. Enforce primary key constraints for item numbers and schedule numbers.

### **3. Performance:**

- Efficient queries are essential for retrieving item details and schedule information during real-time operations.

## DATABASE STRUCTURE

### SHIPPED ITEMS

|                    |        |           |           |             |                |
|--------------------|--------|-----------|-----------|-------------|----------------|
| <u>Item number</u> | Weight | Dimension | Insurance | Destination | Final Delivery |
|--------------------|--------|-----------|-----------|-------------|----------------|

### TRANSPORTATION EVENTS

|                        |                |      |
|------------------------|----------------|------|
| <u>Schedule number</u> | Delivery Route | Type |
|------------------------|----------------|------|

|           |      |         |                    |
|-----------|------|---------|--------------------|
| <u>ID</u> | Type | Address | <u>Item number</u> |
|-----------|------|---------|--------------------|

|                        |                    |
|------------------------|--------------------|
| <u>Schedule number</u> | <u>Item number</u> |
|------------------------|--------------------|

### RETAIL CENTRE

### SHIPPED VIA

### SQL CODE:

```
CREATE database ups;
```

```
USE ups;
```

```
SHOW databases;
```

```
SHOW tables;
```

```
CREATE table retailcentre (id varchar (10) primary key, type char (29), address char (100));
```

```
CREATE table shippeditems (itno int primary key, weight float, dimension int, insurance int, destination char (100), finaldel date, id varchar (18), foreign key(id) references retailcentre(id));
```

```
CREATE table transportationevents (schemenum int, type char (20), deliveryroute char (50));
```

```
INSERT INTO retailcentre VALUES ('2a15', 'truck', 'hyd');
```

```
INSERT INTO retailcentre VALUES ('2b15', 'flight', 'hyd');
```

```
INSERT INTO retailcentre VALUES ('2c16', 'train', 'vizag');
```

```
INSERT INTO shippeditems VALUES (1,5,20,2, jubilee hills', '2023-08-29',"2015");
```

```
INSERT INTO shippeditems VALUES (2,3,5,48,1, 'banjara hills', '2023-09-20',"2615");
```

```
INSERT INTO shippeditems VALUES (3,4.8,35,2, MVP colony", "2023-09-09", "2c16");
```

```
INSERT INTO transportationevents VALUES (25,'truck','Vijayawada to hyd');
```

```
INSERT INTO transportationevents VALUES (26,'flight','Vizag to hyd');
```

```
INSERT INTO transportationevents VALUES (27,'truck','hyd to vizag');
```

```
SELECT *FROM retailcentre;
```

```
SELECT *FROM shippeditems; SELECT *FROM transportationevents;
```






```
1 CREATE database ups;
2 USE ups;
3 SHOW databases;
4 SHOW tables;
5 CREATE table retailcentre(id varchar(10) primary key,type char(20),addre
6 CREATE table shippeditems(itno int primary key,weight float,dimension in
7 CREATE table transportationevents(schedulenum int,type char(50),delivery
8 INSERT INTO retailcentre VALUES('2a15','truck','hyd');
9 INSERT INTO retailcentre VALUES('2b15','flight','hyd');
10 INSERT INTO retailcentre VALUES('2c16','train','vizag');
11 INSERT INTO transportationevents VALUES(25,'truck','vijayawada to hyd');
12 INSERT INTO transportationevents VALUES(26,'flight','vizag to hyd');
13 INSERT INTO transportationevents VALUES(27,'truck','hyd to vizag');
14 INSERT INTO shippeditems VALUES(1,5,20,2,'jublie hills','2023-08-29','2a
15 INSERT INTO shippeditems VALUES(2,3.5,40,1,'banjara hills','2023-09-20',
16 INSERT INTO shippeditems VALUES(3,4.8,35,2,'MVP colony','2023-09-09','2c
17 SELECT *from retailcentre;
18 SELECT *from shippeditems;
19 SELECT *from transportationevents;
```

Result Grid | Filter Rows

|   | id   | type   | address |
|---|------|--------|---------|
| ▶ | 2a15 | truck  | hyd     |
|   | 2b15 | flight | hyd     |
|   | 2c16 | train  | vizag   |
| * | NULL | NULL   | NULL    |

|    |        |                   |
|----|--------|-------------------|
| 25 | truck  | vijayawada to hyd |
| 26 | flight | vizag to hyd      |
| 27 | truck  | hyd to vizag      |

transportationevents5 x

Result Grid |   Filter Rows:  | Edit:    | Export/Im

|   | itno | weight | dimension | insurance | destination   | finaldel   | id   |
|---|------|--------|-----------|-----------|---------------|------------|------|
| ▶ | 1    | 5      | 20        | 2         | jublie hills  | 2023-08-29 | 2a15 |
|   | 2    | 3.5    | 40        | 1         | banjara hills | 2023-09-20 | 2b15 |
|   | 3    | 4.8    | 35        | 2         | MVP colony    | 2023-09-09 | 2c16 |
| • | NULL | NULL   | NULL      | NULL      | NULL          | NULL       | NULL |

## **CONCLUSION**

The Delivery Management System database is designed to handle the critical components of the logistics process, ensuring efficient tracking of delivery items, schedules, routes, and destinations. By organizing data into structured relational tables, the system ensures data accuracy and quick retrieval, enabling seamless delivery operations.

## References

1. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*. Pearson.
2. Date, C. J. (2004). *An Introduction to Database Systems*. Addison-Wesley.
3. Connolly, T. M., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson.
4. Rob, P., & Coronel, C. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.
5. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
6. Korth, H. F., & Silberschatz, A. (2002). *Database System Concepts*. McGraw-Hill.
7. O'Neil, P., & O'Neil, E. (2009). *Database: Principles, Programming, and Performance*. Morgan Kaufmann.
8. Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, 13(6), 377-387.
9. Chen, P. P. (1976). "The Entity-Relationship Model—Toward a Unified View of Data." *ACM Transactions on Database Systems*, 1(1), 9-36.

A FIELD PROJECT/IDP REPORT ON

**Database architecture**

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

BADDIGAM SUCHETHA (221FA19004)

ATCHYUTA LOHITH (221FA19040)

GAVARA VEERA VENKATA  
KRISHNA VAMSI (221FA19044)

VUNADI TARUN KUMAR  
REDDY (221FA19057)



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**Department of ACSE**

**School of Computing and Informatics**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH  
(Deemed to be University)**

**Vadlamudi, Guntur, Andhra Pradesh-522213, India**

**April, 2024**



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

### CERTIFICATE

This is to certify that the field project/IDP entitled "Database architecture" being submitted by BADDIGAMSUCHETHA-221FA19004, ATCHYUTALOHITH-221FA19040, GAVARA VEERA VENKATA KRISHNA VAMSI -221FA19044, VUNADI TARUN KUMAR REDDY-221FA19057 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

Guide :

HOD/ACSE

Dr. Venkatesulu Dondeti





**VIGNAN'S**  
Foundation for Science, Technology & Research  
(Deemed to be University)  
-Estd. u/s 3 of UGC Act 1956

## **DECLARATION**

We hereby declare that our project work described in the field project titled “Database architecture” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

|                                       |            |
|---------------------------------------|------------|
| BADDIGAM SUCHETHA                     | 221FA19004 |
| ATCHYUTA LOHITH                       | 221FA19040 |
| GAVARA VEERA VENKATA<br>KRISHNA VAMSI | 221FA19044 |
| VUNADI TARUN KUMAR REDDY              | 221FA19057 |

## Table of Contents

1. Abstract
2. Introduction
3. Problem Statement
4. DBMS Architectures
  - 2-Tier Architecture
    - Advantages
    - Disadvantages
  - 3-Tier Architecture
    - Advantages
    - Disadvantages
  - Comparison
5. Conclusion
6. References

## I. ABSTRACT

For a web-based airline reservation and ticketing system, a 3-tier architecture is preferable due to its scalability, security, and maintainability. The system must be capable of managing large volumes of user traffic, processing complex queries in real-time, and handling sensitive information such as personal and payment details. Two commonly considered architectural models are the 2-tier and 3-tier architectures, each offering distinct advantages and trade-offs in terms of complexity, resource management, and security. The decision between these architectures has significant implications for how the system performs under high demand and how easily it can adapt to future growth and evolving user needs. In this architecture, the application server mediates between the client and the database, ensuring efficient request handling, secure data processing, and easy updates to business logic without affecting other layers. The 2-tier architecture, where clients directly access the database, lacks scalability and poses security risks, making it unsuitable for large, complex systems. Therefore, the 3-tier architecture is ideal for handling high user loads and ensuring data integrity in an airline booking system.

## II. INTRODUCTION

When designing a web-based system for making airline reservations and selling tickets, selecting the appropriate database architecture is crucial for ensuring the system's performance, scalability, security, and long-term maintainability. The system must be capable of managing large volumes of user traffic, processing complex queries in real-time, and handling sensitive information such as personal and payment details. Two commonly considered architectural models are the 2-tier and 3-tier architectures, each offering distinct advantages and trade-offs in terms of complexity, resource management, and security. The decision between these architectures has significant implications for how the system performs under high demand and how easily it can adapt to future growth and evolving user needs.

The 2-tier architecture involves direct communication between the client and the database, which simplifies the design and may be suitable for smaller-scale applications with limited user interaction. However, for an airline reservation system, which requires robust performance, high availability, and secure handling of sensitive data, the 2-tier model may introduce limitations. Direct client access to the database can lead to security vulnerabilities and scalability issues as the user base grows, potentially affecting response times and creating bottlenecks. Additionally, maintaining business logic within the client or database layer in this architecture can complicate updates and system upgrades.

In contrast, the 3-tier architecture separates the application into three distinct layers: the presentation layer (user interface), the application layer (business logic), and the database layer. This model enhances scalability and security by decoupling the client from direct database access, allowing the application layer to manage interactions and enforce data security protocols. In the case of an airline reservation system, the 3-tier architecture enables the system to handle large volumes of concurrent users, distribute the load efficiently across multiple servers, and provide secure, real-time access to flight data, reservations, and payment processing. By separating concerns across these layers, the 3-tier architecture also simplifies system maintenance and upgrades, making it easier to implement new features, optimize performance, and ensure the system can scale as user demand increases. Ultimately, the 3-tier architecture is better suited for the complex, high-traffic requirements of an airline reservation system, providing the flexibility and security needed for seamless operations.

### **III. PROBLEM STATEMENT**

If you were designing a web-based system to make airline reservations and sell airline tickets, which database architecture would you choose from different types of architectures (2-tier and 3-tier)? Why? Why would the other architectures not be a good choice?

## IV. DBMS ARCHITECTURES

### 2-Tier Architecture

In a 2-tier architecture, there are two main components: the client-side (user interface) and the server-side (database server). The client communicates directly with the database server.

#### Advantages

- **Simplicity:** Easy to implement and manage as there are fewer components involved.
- **Lower Latency:** Direct access to the database can result in lower latency for simple operations.
- **Suitable for Small Applications:** Works well for small-scale applications with limited concurrent users and transactions.

#### Disadvantages

- **Scalability Issues:** It's difficult to scale because the client communicates directly with the database. This can lead to performance bottlenecks as the user base grows.
- **Security Risks:** Storing database credentials on the client-side can lead to unauthorized access.
- **Maintenance:** Changes to the database schema may require updates on all client systems, which complicates system upgrades.

### 3-Tier Architecture

In a 3-tier architecture, three main components are present: the client-side (user interface), the application server (business logic), and the database server. The client communicates with the application server, which in turn communicates with the database.

#### Advantages

- **Scalability:** The application server can be scaled horizontally, allowing the system to handle more users efficiently. This architecture is suitable for large applications with high concurrency.
- **Security:** By separating the business logic from the database, access control to sensitive data is more robust.
- **Modularity:** Better separation of concerns makes the system more maintainable and adaptable to change. It also enables easy updates to business logic without affecting the client or database layers.

- Load Balancing: Distributes user requests across multiple servers, improving performance and fault tolerance.

#### Disadvantages

- Complexity: More components make this architecture more complex to implement and maintain.
- Higher Latency: The added application server layer can introduce slightly higher latency compared to direct database access in a 2-tier model.

#### Comparison

In the context of an airline reservation and ticketing system, a 3-tier architecture is preferable due to the following reasons:

- Scalability: The system must handle high traffic, especially during peak travel seasons. The 3-tier architecture's ability to scale makes it ideal.
- Security: Sensitive data like passenger details and payment information requires strong protection, which is more effectively managed in the 3-tier architecture.
- Maintainability: The airline industry frequently changes schedules, routes, and fares. The modular nature of 3-tier architecture allows for smoother updates without disrupting the entire system.

## V. CONCLUSION

For an airline reservation and ticketing system, the 3-tier architecture is the optimal choice due to its scalability, security, and maintainability. The application server layer ensures that user interactions are handled efficiently while protecting sensitive data and maintaining high system performance under heavy loads. On the other hand, the 2-tier architecture falls short in terms of scalability and security, making it unsuitable for complex, high-traffic systems like airline booking platforms. Therefore, the 3-tier architecture provides a robust infrastructure that supports the long-term reliability and flexibility required in this industry.



## VI. REFERENCES

1. Korth, H. F., & Silberschatz, A. (2011). \*Database System Concepts\* (6th ed.). McGraw-Hill.
2. [JavaTpoint DBMS Tutorial](<https://www.javatpoint.com/dbms-tutorial>)

A FIELD PROJECT/IDP REPORT ON

**ER – diagram into Relational**

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

ABUBBAKAR SIDDIQ SHARIFF (221FA19006)

SUGGUNA RAMYA SRI (221FA19013)

GADIPUDI VINEELA CHOWDARY (221FA19022)

VANKUDOTHU HARSHITH (221FA19026)



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**Department of ACSE**

**School of Computing and Informatics**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH  
(Deemed to be University)**

**Vadlamudi, Guntur, Andhra Pradesh-522213, India**

**April, 2024**



**VIGNAN'S**  
Foundation for Science, Technology & Research  
(Deemed to be University)  
-Estab. u/s 3 of UGC Act 1956

### CERTIFICATE

This is to certify that the field project/IDP entitled "ER – diagram into Relational" being submitted by ABUBBAKAR SIDDIQ SHARIFF-221FA19006, SUGGUNA RAMYA SRI-221FA19013, GADIPUDI VINEELA CHOWDARY-221FA19022, VANKUDOTHU HARSHITH-221FA19026 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

**Guide :**

**HOD/ACSE**

Dr.Venkatesulu Dondeti



**VIGNAN'S**  
Foundation for Science, Technology & Research  
(Deemed to be University)  
-Estd. u/s 3 of UGC Act 1956

## DECLARATION

We hereby declare that our project work described in the field project titled “ER – diagram into Relational” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH.Rose Rani.

|                           |            |
|---------------------------|------------|
| ABUBBAKAR SIDDIQ SHARIFF  | 211FA19006 |
| SUGGUNA RAMYA SRI         | 211FA19013 |
| GADIPUDI VINEELA CHOWDARY | 211FA19022 |
| VANKUDOTHU HARSHITH       | 211FA19026 |

## Table of Contents

1. Abstract
2. Introduction
3. Database Design and Implementation
  1. Software and Hardware Requirements
4. Entity-Relationship (ER) Model
  1. Entities and Attributes
  2. Relationships
5. Relational Model
  1. Tables and Constraints
6. Query Implementation
7. Result Analysis
  1. Data Integrity
  2. Query Performance
  3. Scalability and Future Considerations
8. Conclusion
9. References

## **1. Abstract**

This document presents the design and implementation of a relational database system for DAG's galleries in New Delhi and Mumbai. The system is designed to capture key information about artists, artworks, groups, and customers, including their preferences. By mapping an Entity-Relationship (ER) model into a relational schema, the database supports many-to-many and one-to-many relationships. The aim is to provide a scalable, secure, and efficient system that handles customer transactions, tracks artwork classifications, and offers insightful queries to aid decision-making for the galleries.

## 2. Introduction

Art galleries, such as DAG, require an efficient and robust system to manage their diverse data and operations, which include tracking artists, artworks, and customer preferences. Effective management of this data is crucial not only for improving daily operations but also for providing valuable insights into customer trends, enabling galleries to make informed business decisions. By organizing and maintaining accurate records of artists and their works, as well as customer interactions, the gallery can enhance customer engagement and personalize their offerings. This project aims to design and implement a relational database that captures all the relevant information needed by DAG's galleries, facilitating better operational management and strategic planning.

The relational database must effectively handle a variety of relationships between key entities, such as artists, artworks, groups, and customers. This includes the ability to link artists to their respective artworks while allowing for the categorization of artworks by style, medium, and other relevant attributes. Additionally, the system must accommodate the galleries' need to track customer spending habits and their preferences for specific artists and types of artwork. By capturing this data, the gallery can analyze customer behavior, tailor marketing strategies, and curate exhibitions that resonate with their audience.

Furthermore, the database should be designed with scalability in mind, allowing it to grow as the gallery network expands. As new galleries are added or as the volume of artworks and customer data increases, the system must maintain its performance and efficiency. To achieve this, the database design will incorporate normalization techniques to reduce redundancy and improve data integrity, along with robust indexing strategies to enhance query performance. Ultimately, the successful implementation of this relational database will empower DAG's galleries to streamline operations, enhance customer relationships, and foster a deeper understanding of the art market.

### **3. Database Design and Implementation**

#### 3.1 Software and Hardware Requirements

The implementation of this database requires the following software and hardware:

- Software Requirements:

- MySQL database management system (DBMS)
- SQL Query tools such as MySQL Workbench
- Server OS: Windows/Linux

- Hardware Requirements:

- Minimum 8GB RAM
- 2.5 GHz Processor or higher
- Minimum 100 GB storage space



## 4. Entity-Relationship (ER) Model

### 4.1 Entities and Attributes

The key entities involved in the database design for DAG's galleries are:

- Artists: Information about artists, such as name, birthplace, and style.
- Artwork: Details of each piece of artwork, including title, type, and price.
- Groups: Categories or classifications for artwork (e.g., styles, periods).
- Customers: Information about customers, including their preferences for artists and groups.

### 4.2 Relationships

- One-to-Many (1:N):
  - Artists can create multiple pieces of artwork, but each artwork is linked to only one artist.
- Many-to-Many (N:M):
  - Each artwork can belong to multiple groups, and a group can include multiple pieces of artwork.
  - Customers may have preferences for multiple artists and artwork groups.

## 5. Relational Model

### 5.1 Tables and Constraints

The relational model consists of several tables, including:

- Artists Table: Stores details of each artist with a unique `ArtistID`.
- Artwork Table: Contains the details of each artwork and a foreign key referencing `ArtistID`.
- Groups Table: Lists different categories of artwork, with a unique `GroupID`.
- Customers Table: Stores customer information, including spending habits.

Constraints:

- Primary Keys: `ArtistID`, `ArtworkID`, `GroupID`, and `CustomerID`.
- Foreign Keys: Links between `Artwork` and `Artists`, as well as junction tables for many-to-many relationships (e.g., `Artwork\_Groups`, `Customer\_Artists`, `Customer\_Groups`).

### Artist Table

| ArtistID | Name              | Birthplace           | Age | Style             |
|----------|-------------------|----------------------|-----|-------------------|
| 1        | Vincent van Gogh  | Zundert, Netherlands | 37  | Post-Impressionis |
| 2        | Pablo Picasso     | Málaga, Spain        | 91  | Cubism            |
| 3        | Frida Kahlo       | Coyoacán, Mexico     | 47  | Surrealism        |
| 4        | Leonardo da Vinci | Vinci, Italy         | 67  | Renaissance       |

## Artwork Table

| ArtworkID | Title          | ArtistID | YearCreated | Type          | Price       |
|-----------|----------------|----------|-------------|---------------|-------------|
| 1         | Starry Night   | 1        | 1889        | Oil on Canvas | 1000000.00  |
| 2         | Guernica       | 2        | 1937        | Oil on Canvas | 2000000.00  |
| 3         | The Two Fridas | 3        | 1939        | Oil on Canvas | 1500000.00  |
| 4         | Mona Lisa      | 4        | 1503        | Oil on Poplar | 85000000.00 |

## Group Table

| GroupID | GroupName     |
|---------|---------------|
| 1       | Impressionism |
| 2       | Surrealism    |
| 3       | Cubism        |
| 4       | Renaissance   |

## Customer Table

| CustomerID | Name          | Address                        | TotalSpent |
|------------|---------------|--------------------------------|------------|
| 1          | Alice Johnson | 123 Art St, New York, NY       | 25000.00   |
| 2          | Bob Smith     | 456 Canvas Rd, Los Angeles, CA | 50000.00   |
| 3          | Carol White   | 789 Gallery Ave, Chicago, IL   | 30000.00   |
| 4          | David Brown   | 101 Painting Blvd, Miami, FL   | 75000.00   |

## 6. Query Implementation

```
CREATE TABLE Artists (ArtistID INT PRIMARY KEY,
Name VARCHAR(100) NOT NULL,
Birthplace VARCHAR(100), Age INT,
Style VARCHAR(100)
);
```

```
CREATE TABLE Artwork (ArtworkID INT PRIMARY KEY, Title VARCHAR(100) NOT
NULL,
ArtistID INT, YearCreated YEAR,
Type VARCHAR(50), Price DECIMAL(10, 2), FOREIGN KEY (ArtistID)
REFERENCES Artists(ArtistID));
```

```
CREATE TABLE Groups (GroupID INT PRIMARY KEY,
GroupName VARCHAR(100) UNIQUE NOT NULL
);
```

```
CREATE TABLE Customers (CustomerID INT PRIMARY
KEY,
Name VARCHAR(100) NOT NULL,
Address VARCHAR(255), TotalSpent DECIMAL(10, 2)
);
```

```
INSERT INTO Artists (ArtistID, Name, Birthplace, Age, Style) VALUES
(1, 'Vincent van Gogh', 'Zundert, Netherlands', 37, 'Post- Impressionism'),
(2, 'Pablo Picasso', 'Málaga, Spain', 91, 'Cubism'),
(3, 'Frida Kahlo', 'Coyoacán, Mexico', 47, 'Surrealism'),
(4, 'Leonardo da Vinci', 'Vinci, Italy', 67, 'Renaissance');
```

```
INSERT INTO Artwork (ArtworkID, Title, ArtistID, YearCreated, Type, Price)
VALUES
```

```
(1, 'Starry Night', 1, 1889, 'Oil on
Canvas', 1000000.00),
(2, 'Guernica', 2, 1937, 'Oil on
Canvas', 2000000.00),
(3, 'The Two Fridas', 3, 1939, 'Oil on
Canvas', 1500000.00),
(4, 'Mona Lisa', 4, 1503, 'Oil on
Poplar', 85000000.00);
```

```
INSERT INTO Groups (GroupID,
GroupName) VALUES (1, 'Impressionism'),
```

```
(2, 'Surrealism'),
(3, 'Cubism'),
(4, 'Renaissance');
```

```
INSERT INTO Customers (CustomerID, Name, Address,
TotalSpent) VALUES
```

```
(1, 'Alice Johnson', '123 Art St, New York, NY', 25000.00),
(2, 'Bob Smith', '456 Canvas Rd, Los Angeles, CA', 50000.00),
(3, 'Carol White', '789 Gallery Ave,
Chicago, IL', 30000.00),
(4, 'David Brown', '101 Painting Blvd, Miami, FL', 75000.00);
```

```
Select *from Artists ;
```

```
Select * from Artwork ;
```

```
Select *from Groups ;
```

```
Select *from Customer;
```

```
Select *from Artword_Groups;
```

```
Select *from Customer_Artists;
```

```
Select *from Customer_Groups;
```

## **7. Result Analysis**

### **8.1 Data Integrity**

The database maintains data integrity by enforcing primary and foreign key constraints. Each piece of data is uniquely identified, ensuring no duplication or inconsistency.

### **8.2 Query Performance**

The relational database supports efficient querying for customer and artist data. Indexed fields such as `ArtistID` and `ArtworkID` help optimize query performance, especially for large datasets.

### **8.3 Scalability and Future Considerations**

The database is designed to scale with the growth of the galleries. As new artworks, artists, and customers are added, the database structure can easily accommodate them without requiring major changes.

## **8. Conclusion**

This database design for DAG's galleries provides a robust solution to manage essential data such as artists, artworks, groups, and customers. The ER model and relational schema ensure efficient handling of many-to-many relationships and support for future expansion. The system's scalability and query performance enable DAG's galleries to maintain data integrity while supporting a growing number of users and transactions.

## 9. References

- Korth, H. F., & Silberschatz, A. (2011). \*Database System Concepts\* (6th ed.). McGraw-Hill.
- MySQL Documentation. <https://dev.mysql.com/doc/>
- Javatpoint: <https://www.javatpoint.com/dbms>



A FIELD PROJECT/IDP REPORT ON

**ER – diagram into SQL Query**

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

|                                  |              |
|----------------------------------|--------------|
| PARUCHURI ASRITHA                | (221FA19001) |
| RAAVI NAGA SUMITHRA<br>CHOWDARY  | (221FA19009) |
| KILARI DHANA MALLIKARJUNA<br>RAO | (221FA19019) |
| MUVVA JITENDRA                   | (221FA19056) |



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**Department of ACSE**

**School of Computing and Informatics**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH  
(Deemed to be University)**

**Vadlamudi, Guntur, Andhra Pradesh-522213, India**

**April, 2024**



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

## CERTIFICATE

This is to certify that the field project/IDP entitled "ER – diagram into SQL Query" being submitted by PARUCHURI ASRITHA-221FA19001, RAAVI NAGA SUMITHRA CHOWDARY- 221FA19009, KILARI DHANA MALLIKARJUNA RAO-221FA19019, MUVVA JITENDRA-221FA19056 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

  
Guide :

  
HOD/ACSE

Dr.Venkatesulu Dondeti



**VIGNAN'S**  
Foundation for Science, Technology & Research  
(Deemed to be University)  
-Estd. u/s 3 of UGC Act 1956

## **DECLARATION**

We hereby declare that our project work described in the field project titled “ER – diagram into SQL Query” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

|                                  |            |
|----------------------------------|------------|
| PARUCHURI ASRITHA                | 221FA19001 |
| RAAVI NAGA SUMITHRA<br>CHOWDARY  | 221FA19009 |
| KILARI DHANA MALLIKARJUNA<br>RAO | 221FA19019 |
| MUVVA JITENDRA                   | 221FA19056 |

## Table of Contents

1. Abstract
2. Introduction
3. Entity-Relationship (ER) Model
4. Relational Models
5. Optimization
6. SQL Code
7. Result Analysis
8. Conclusion
9. References

## **1. Abstract**

This project aims to design a database for Dane County Airport to improve organization and management of critical operations. The database captures detailed information about airplanes, models, technicians, traffic controllers, tests, and employees. The system is designed to handle various relationships such as those between technicians and airplanes, employees and unions, and airplanes and models. By using structured query language (SQL), the database supports efficient management and retrieval of information, enhancing the overall functionality and organization of the airport.

## 2. Introduction

A database is a structured collection of data that is stored and managed in a way that allows for efficient retrieval, manipulation, and organization. Think of it as a highly organized digital filing cabinet where information can be stored in tables, making it easy to find and use.

In this project, a database management system (DBMS) is designed for the Dane County Airport to address issues of poor organization. The airport houses numerous airplanes and requires maintenance work performed by technicians who specialize in different airplane models. Additionally, there are traffic controllers and tests conducted periodically for the airplanes to ensure safety standards. The goal of this database design is to effectively manage all relevant data in a systematic manner that ensures smooth airport operations.

### Entities:

An entity in a database is a distinct object or concept that you want to store information about. It could be a person, place, thing, or event. Each entity has attributes that describe its characteristics.

### Relationships:

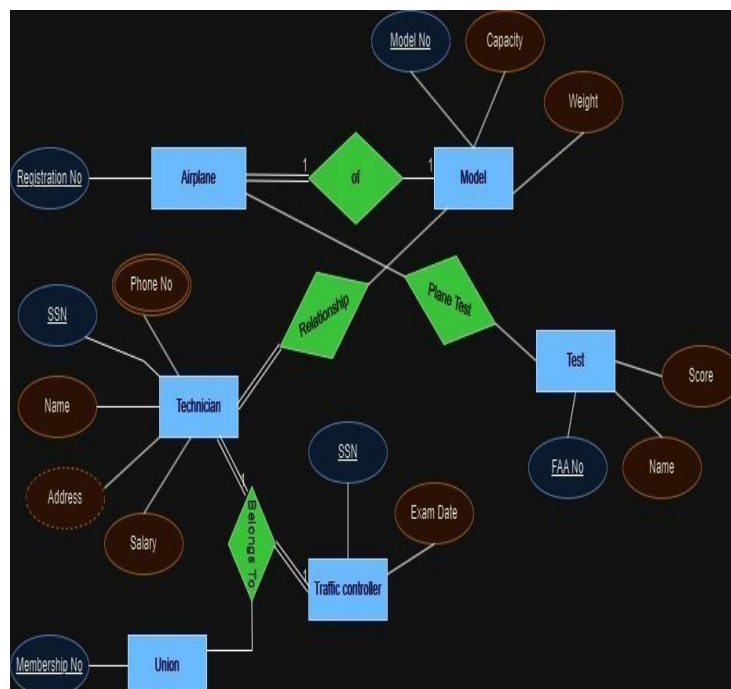
In relational databases, tables can be related to each other. For example, a table for orders might be related to a table for customers. Relationships are established using keys. A primary key uniquely identifies each record in a table, while a foreign key is a field in one table that links to the primary key in another table.

This database will organize all the critical information related to airplanes, models, technicians, traffic controllers, tests, and employees to create an efficient and scalable system.

### 3. Entity-Relationship (ER) Model

The ER model for the airport database includes the following entities:

1. Airplane: Each airplane is identified by a registration number ( `Aregno` ) and is associated with a specific model ( `Amodelno` ).
2. Model: Each airplane model is identified by a model number ( `Modelno` ) and has attributes such as capacity and weight.
3. Technician: Technicians are responsible for maintaining airplanes. The database will store technician details such as SSN ( `TSSN` ), name, address, phone number, and salary. Each technician can work on multiple airplane models.
4. Traffic Controller: Traffic controllers must undergo annual medical exams. The date of their most recent medical exam is stored.
5. Test: Airplanes are subject to various tests. Each test is identified by an FAA test number ( `FAANO` ), a name, and a maximum possible score.
6. Employee: Employees (including technicians) are associated with a union, and their union membership number is stored.



## 4. Relational Models

The relational model maps the ER entities to database tables:

- Airplane Table: `Aregno` (primary key), `Amodelno` (foreign key referencing `Model`)
- Model Table: `Modelno` (primary key), `Capacity`, `Weight`
- TechnicianMaintains Table: `TSSN` (primary key), `Name`, `Address`, `PhoneNo`, `Aregno` (foreign key referencing `Airplane`), `Salary`
- Traffic Controller Table: `TCSSN` (primary key), `DateOfMostRecentExam`, `Aregno` (foreign key referencing `Airplane`)
- Expertise Table: `TSSN` (foreign key referencing `TechnicianMaintains`), `Modelno` (foreign key referencing `Model`)
- Test Table: `FAANO` (primary key), `MaxScore`, `Name`, `Aregno` (foreign key referencing `Airplane`)
- Employee Table: `ESSN` (primary key), `Ename`, `Unionno`
- BelongsTo Table: `TSSN` (foreign key referencing `TechnicianMaintains`), `Unionno` (foreign key referencing `Employee`)



## 5. Optimization

To optimize database structure, relationships between entities such as technicians and airplanes are efficiently handled using primary and foreign key constraints. Indexing is applied to frequently queried fields, such as `Aregno` and `TSSN`, to speed up query performance. Additionally, redundant data is minimized to reduce storage costs.

Airplane has ( A reg\_no, A model\_no, model\_no)

Model (Model\_no, capacity, weight)

TechnicianMaintains(Address, TSSN, Tname, Phoneno, A reg\_no, Salary)

Trafficcontrollers have( TCSSN, Date of most recent exam, A reg\_no)

Expertise (TSSN, Model\_no )

Testundergoes ( FAA\_no, Maxscore, name, A reg\_no)

Employee( Ename, ESSN, unionmembership\_no )

Belongs to(TSSN, union-membership\_no)

## 6. SQL Code

```
CREATE DATABASE Airport;
```

```
USE Airport;
```

```
-- Create Model Table
```

```
CREATE TABLE Model (
```

```
 modelno INT PRIMARY KEY,
```

```
 capacity INT,
```

```
 weight FLOAT
```

```
);
```

```
INSERT INTO Model (modelno, capacity, weight) VALUES
```

```
(1, 150, 60.5), (2, 180, 70.0), (3, 200, 75.2), (4, 220, 80.1), (5, 250, 85.0);
```

```
-- Create Airplane Table
```

```
CREATE TABLE Airplane (
```

```
 Aregno INT PRIMARY KEY,
```

```
 Amodelno INT,
```

```
 FOREIGN KEY (Amodelno) REFERENCES Model(modelno)
```

```
);
```

```
INSERT INTO Airplane (Aregno, Amodelno) VALUES
```

```
(101, 1), (102, 2), (103, 3), (104, 4), (105, 5);
```

```
-- Create Employee Table
```

```
CREATE TABLE Employee (
```

```
 ESSN INT PRIMARY KEY,
```

```
 Ename VARCHAR(50),
```

```
 Unionno INT
```

```
);
```

```
INSERT INTO Employee (ESSN, Ename, Unionno) VALUES
```

(501, 'Alice Brown', 1001), (502, 'Bob White', 1002), (503, 'Charlie Green', 1003),  
(504, 'David Black', 1004), (505, 'Eva Blue', 1005);

-- Create TechnicianMaintains Table

```
CREATE TABLE TechnicianMaintains (
 TSSN INT PRIMARY KEY,
 TName VARCHAR(50),
 Address VARCHAR(100),
 PhoneNo VARCHAR(15),
 Aregno INT,
 Salary INT,
 FOREIGN KEY (Aregno) REFERENCES Airplane(Aregno)
);

INSERT INTO TechnicianMaintains (TSSN, TName, Address, PhoneNo, Aregno, Salary)
VALUES
(201, 'John Doe', '123 Main St', '555-1234', 101, 50000),
(202, 'Jane Smith', '456 Elm St', '555-5678', 102, 52000),
(203, 'Mike Johnson', '789 Oak St', '555-9012', 103, 54000),
(204, 'Emily Davis', '321 Pine St', '555-3456', 104, 56000),
(205, 'Sarah Wilson', '654 Cedar St', '555-7890', 105, 58000);
```

-- Create Traffic Controller Table

```
CREATE TABLE Traffic (
 TCSSN INT PRIMARY KEY,
 DateOfMostRecentExam DATE,
 Aregno INT,
 FOREIGN KEY (Aregno) REFERENCES Airplane(Aregno)
);

INSERT INTO Traffic (TCSSN, DateOfMostRecentExam, Aregno) VALUES
(301, '2024-01-15', 101), (302, '2024-02-20', 102), (303, '2024-03-10', 103),
(304, '2024-04-05', 104), (305, '2024-05-25', 105);
```

-- Create Expertise Table

```
CREATE TABLE Expertise (
 TSSN INT,
 Modelno INT,
 PRIMARY KEY (TSSN, Modelno),
 FOREIGN KEY (TSSN) REFERENCES TechnicianMaintains(TSSN),
 FOREIGN KEY (Modelno) REFERENCES Model(modelno)
);
INSERT INTO Expertise (TSSN, Modelno) VALUES
(201, 1), (202, 2), (203, 3), (204, 4), (205, 5);
```

-- Create Test Table

```
CREATE TABLE Test (
 FAANO INT PRIMARY KEY,
 MaxScore INT,
 Name VARCHAR(10),
 Aregno INT,
 FOREIGN KEY (Aregno) REFERENCES Airplane(Aregno)
);
INSERT INTO Test (FAANO, MaxScore, Name, Aregno) VALUES
(401, 100, 'Alpha', 101), (402, 95, 'Bravo', 102), (403, 98, 'Charlie', 103),
(404, 97, 'Delta', 104), (405, 99, 'Echo', 105);
```

-- Create Belong Table

```
CREATE TABLE Belong (
 TSSN INT,
 Unionno INT,
 PRIMARY KEY (TSSN, Unionno),
 FOREIGN KEY (TSSN) REFERENCES TechnicianMaintains(TSSN),
```

FOREIGN KEY (Unionno) REFERENCES Employee(Unionno)

);

INSERT INTO Belong (TSSN, Unionno) VALUES

(201, 1001), (202, 1002), (203, 1003), (204, 1004), (205, 1005);

---

## 7. Result Analysis

The database design and implementation for the Dane County Airport were executed successfully, resulting in a structured system capable of efficiently managing and retrieving essential information. This section provides an analysis of the system's key aspects, including data integrity, query performance, and scalability.

### 7.1 Data Integrity

Data integrity is crucial for maintaining accurate and reliable data within the database. The following measures were implemented to ensure data integrity:

- **Primary Keys:** Each table has a defined primary key that uniquely identifies each record. For example, the `Aregno` in the `Airplane` table and `modelno` in the `Model` table serve as unique identifiers, preventing duplicate entries.
- **Foreign Keys:** Relationships between tables are enforced through foreign key constraints. For instance, `Amodelno` in the `Airplane` table references `modelno` in the `Model` table, ensuring that only valid airplane models are associated with registered airplanes. This also prevents orphan records in the `Airplane` table.
- **Data Types and Constraints:** Appropriate data types and constraints (such as NOT NULL) were defined for each column, which helps maintain consistency in the data. For example, the `Salary` column in the `TechnicianMaintains` table is defined as an integer to avoid erroneous data entries.

### 7.2 Query Performance

The implementation of the database allows for efficient query processing, which is essential for operational tasks at the airport. Key observations regarding query performance include:

- **Indexing:** Indexes can be applied to frequently queried columns, such as `Aregno` and `TSSN`, to enhance search performance. This optimization facilitates quicker retrieval of records, which is critical in a dynamic environment like an airport.
- **Join Operations:** The relational model allows for easy join operations, enabling the extraction of related data across multiple tables. For instance, a query that retrieves all technicians and their associated airplane models can be executed efficiently using JOIN statements.

- Example Queries: Sample SQL queries demonstrated the performance of the database:
- Retrieve all airplanes and their models:

```
SELECT Airplane.Aregno, Model.modelno, Model.capacity
FROM Airplane
JOIN Model ON Airplane.Amodelno = Model.modelno;
```

This query returns the registration numbers of airplanes along with their model numbers and capacities, highlighting the relational capabilities of the database.

### **7.3 Scalability and Future Considerations**

The design of the airport database is scalable, accommodating potential growth in operations and data. Future considerations include:

- Data Volume: As the airport grows, the volume of data may increase significantly. The current structure can be expanded by adding more records to existing tables without requiring a redesign.
- New Features: The database can be enhanced to include additional features, such as integrating a user interface for data entry and retrieval, or incorporating advanced analytics to monitor airplane performance and technician workload.
- Modular Design: The modular nature of the database allows for new tables and relationships to be added easily. For instance, a new table for airport passenger data could be integrated without disrupting existing structures.
- Performance Monitoring: Regular monitoring of database performance and usage patterns will help identify areas that may require optimization, such as indexing or query tuning, to maintain efficient operation as the system scales.

## **8. Conclusion**

In conclusion, the airport database effectively meets the current organizational needs while remaining adaptable to future growth and enhancements. The implementation of stringent data integrity measures, efficient query performance, and scalable architecture positions the airport for successful operational management.



## 9.References

1. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*. Pearson.
2. Date, C. J. (2004). *An Introduction to Database Systems*. Addison-Wesley.
3. Connolly, T. M., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson.
4. Rob, P., & Coronel, C. (2016). *Database Systems: Design, Implementation, & Management*. Cengage Learning.
5. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). *Database Systems: The Complete Book*. Prentice Hall.
6. Korth, H. F., & Silberschatz, A. (2002). *Database System Concepts*. McGraw-Hill.
7. O'Neil, P., & O'Neil, E. (2009). *Database: Principles, Programming, and Performance*. Morgan Kaufmann.
8. Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, 13(6), 377-387.
9. Chen, P. P. (1976). "The Entity-Relationship Model—Toward a Unified View of Data." *ACM Transactions on Database Systems*, 1(1), 9-36.
10. Firebird Project. (n.d.). "Firebird SQL." Retrieved from <https://firebirdsql.org/>

A FIELD PROJECT/IDP REPORT ON

**ORDER and ORDER\_ITEM relations**

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

|                              |              |
|------------------------------|--------------|
| SUGGUNA PAVAN SAI<br>KRISHNA | (221FA19014) |
| SANDIREDDY VENKATA<br>RAKESH | (221FA19021) |
| KATARU TARUN KUMAR<br>REDDY  | (221FA19039) |
| PATIBANDLA SAI DEEPAK        | (221FA19064) |



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**Department of ACSE**

**School of Computing and Informatics**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH  
(Deemed to be University)**

**Vadlamudi, Guntur, Andhra Pradesh-522213, India**

**April, 2024**



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

### CERTIFICATE

This is to certify that the field project/IDP entitled "ORDER and ORDER\_ITEM relations" being submitted by SUGGUNA PAVAN SAI KRISHNA-221FA19014, SANDIREDDY VENKATA RAKESH-221FA19021, KATARU TARUN KUMAR REDDY-221FA19039, PATIBANDLA SAI DEEPAK-221FA19064 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

  
Guide :

  
HOD/ACSE

Dr.Venkatesulu Dondeti



**VIGNAN'S**  
Foundation for Science, Technology & Research  
(Deemed to be University)  
-Estd. u/s 3 of UGC Act 1956

## **DECLARATION**

We hereby declare that our project work described in the field project titled “ORDER and ORDER\_ITEM relations” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

|                           |            |
|---------------------------|------------|
| SUGGUNA PAVAN SAI KRISHNA | 221FA19014 |
| SANDIREDDY VENKATA RAKESH | 221FA19021 |
| KATARU TARUN KUMAR REDDY  | 221FA19039 |
| PATIBANDLA SAI DEEPAK     | 221FA19064 |

## **Table of Contents**

1. Abstract
2. Introduction
3. Database Design and Implementation
  - 3.1 Software and Hardware Requirements
  - 3.2 Entity-Relationship (ER) Model
4. Relational Model
  - 4.1 Tables and Constraints
5. Query Implementation
6. Result Analysis
  - 6.1 Data Integrity
  - 6.2 Query Performance
  - 6.3 Scalability and Future Considerations
7. Conclusion
8. References

## 1. Abstract

This document presents a comprehensive database design and implementation plan for the order-processing application at ABC, Inc. The primary objective is to establish a structured and efficient database that effectively manages customer orders and order items, while ensuring data integrity, optimal performance, and scalability. As the volume of transactions grows, it is essential that the database can accommodate increasing data loads without sacrificing speed or reliability. By focusing on these core principles, the database will support ABC, Inc.'s operational needs and facilitate enhanced customer service.

The design process involves a thorough analysis of key entities and their interrelationships within the order-processing domain. Critical entities include customers, orders, and order items, each of which plays a vital role in the overall functionality of the application. Understanding the relationships between these entities is crucial for creating a cohesive data model that accurately reflects the business logic of the order-processing system. Additionally, the design incorporates normalization processes to eliminate redundancy and minimize the potential for data anomalies. This ensures that the database not only remains efficient but also retains the integrity of the information stored within it.

Ultimately, the outcome of this design process is a well-organized database schema that adheres to the principles of database normalization. The schema will facilitate efficient data retrieval and manipulation, supporting the operational demands of ABC, Inc. By implementing this structured approach, the database will provide a robust foundation for managing customer orders, enhancing reporting capabilities, and enabling data-driven decision-making. This comprehensive database design is set to improve operational efficiencies and support the long-term growth of ABC, Inc. in the competitive market landscape.

## 2. Introduction

A database is a structured collection of data that is stored and managed in a way that allows for efficient retrieval, manipulation, and organization. Think of it as a highly organized digital filing cabinet where information can be systematically stored in tables, making it easy to find and use. Each table consists of rows and columns, where rows represent individual records and columns represent attributes of those records. This structure not only enhances data organization but also facilitates complex queries and reporting, making it an essential tool for businesses and organizations.

In ABC, Inc.'s order-processing database, the ORDER and ORDER\_ITEM relations play a critical role in managing vital information about customer orders. The ORDER relation includes key attributes such as order number (O), order date (Odate), customer number (Cust), and total amount (Total\_amount). This information allows the company to track customer purchases and analyze order trends over time. On the other hand, the ORDER\_ITEM relation provides detailed information about individual items within each order. It includes attributes such as order number (O), item number (I), quantity ordered (Qty\_ordered), total price (Total\_price), and item-specific discount (Discount%). This level of detail is crucial for managing inventory and ensuring accurate billing.

By applying a natural join between these relations, we can generate a comprehensive view of each order along with its associated items. The natural join merges the data from the ORDER and ORDER\_ITEM tables based on the common attribute, which in this case is the order number (O). This combined dataset enables ABC, Inc. to easily access information about the specific items ordered, their quantities, prices, and any applicable discounts for each order. Such insights facilitate effective order management, enhance customer service, and support data-driven decision-making processes, ultimately contributing to the overall efficiency of the order-processing system.

## **3. Database Design and Implementation**

### **3.1 Software and Hardware Requirements**

- Software:
  - Database Management System (DBMS) such as MySQL or PostgreSQL
  - Development tools for SQL queries and database management
- Hardware:
  - Server with adequate RAM and storage capacity
  - Backup solutions to ensure data integrity

### **3.2 Entity-Relationship (ER) Model**

- Entities:
  - ORDER: Represents customer orders.
  - ORDER\_ITEM: Represents individual items within an order.
- Attributes:
  - ORDER: O, Odate, Cust, Total\_amount
  - ORDER\_ITEM: O, I, Qty\_ordered, Total\_price, Discount%



## 4. Relational Model

### 4.1 Tables and Constraints

The following tables will be created in the database:

- ORDER:

- Attributes: O, Odate, Cust, Total\_amount

- Primary Key: O

- ORDER\_ITEM:

- Attributes: O, I, Qty\_ordered, Total\_price, Discount%

- Primary Key: {O, I}

- Foreign Key: O references ORDER(O)

## 5. Query Implementation

```
CREATE DATABASE ABC_Orders;
```

```
USE ABC_Orders;
```

```
CREATE TABLE ORDER (
```

```
 O INT PRIMARY KEY,
```

```
 Odate DATE,
```

```
 Cust INT,
```

```
 Total_amount DECIMAL(10, 2)
```

```
);
```

```
CREATE TABLE ORDER_ITEM (
```

```
 O INT,
```

```
 I INT,
```

```
 Qty_ordered INT,
```

```
 Total_price DECIMAL(10, 2),
```

```
 Discount DECIMAL(5, 2),
```

```
 PRIMARY KEY (O, I),
```

```
 FOREIGN KEY (O) REFERENCES ORDER(O)
```

```
);
```

## 6. Result Analysis

### 6.1 Resulting Relation Schema (RES)

After performing a natural join on the `ORDER` and `ORDER\_ITEM` relations, the resulting schema RES can be defined as follows:

| O | Odate      | Cust | Total_amount | I    | Qty_ordered | Total_price | Discount% |
|---|------------|------|--------------|------|-------------|-------------|-----------|
| 1 | 2023-01-15 | 101  | 250.00       | 101  | 2           | 100.00      | 10        |
| 1 | 2023-01-15 | 101  | 250.00       | 102  | 1           | 80.00       | 5         |
| 2 | 2023-01-18 | 102  | 150.00       | NULL | NULL        | NULL        | NULL      |

### 6.2 Keys in the Relation RES

The primary key for the RES relation can be determined as:

- {O, I}: This composite key uniquely identifies each record in the RES relation.

### 6.3 Normalization Process

To achieve 3NF, we decompose the RES relation into two separate relations:

- RES1 (Order Details):
  - Attributes: O, Odate, Cust, Total\_amount
  - Primary Key: O
- RES2 (Order Item Details):
  - Attributes: O, I, Qty\_ordered, Total\_price, Discount%
  - Primary Key: {O, I}

This decomposition eliminates transitive dependencies and ensures that each relation adheres to 3NF.

## 7. Conclusion

The natural join of the ORDER and ORDER\_ITEM relations results in the RES schema, which contains all relevant attributes, and the primary keys O (Order Number) and I (Item Number). This schema is in the Third Normal Form (3NF), ensuring data integrity and efficiency by eliminating partial and transitive dependencies. This well-structured and normalized database design is ideal for ABC, Inc.'s order-processing application, facilitating organized and non-redundant data management, ultimately improving the reliability and efficiency of order processing.

## 8. References

1. Date, C. J. (2004). *\*An Introduction to Database Systems\**. Addison-Wesley.
2. Elmasri, R., & Navathe, S. B. (2015). *\*Fundamentals of Database Systems\**. Pearson.
3. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). *\*Database System Concepts\**. McGraw-Hill.

A FIELD PROJECT/IDP REPORT ON

**ORDER and ORDER\_ITEM relations**

Submitted partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

CYBER SECURITY

Submitted by

|                               |              |
|-------------------------------|--------------|
| DAMARACHARLA GEETHA<br>HARIKA | (221FA19020) |
| PALGULLA RANGASWAMI<br>REDDY  | (221FA19030) |
| SYED SHANIKA ZAIDA            | (221FA19038) |
| YADAVALLI MEENAKSHI           | (221FA19055) |



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

**Department of ACSE**

**School of Computing and Informatics**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH  
(Deemed to be University)**

**Vadlamudi, Guntur, Andhra Pradesh-522213, India**

**April, 2024**



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

•Estd. u/s 3 of UGC Act 1956

### **CERTIFICATE**

This is to certify that the field project/IDP entitled "ORDER and ORDER\_ITEM relations" being submitted by DAMARACHARLA GEETHA HARIKA 221FA19020, PALGULLA RANGASWAMIREDDY 221FA19030, SYEDSHANIKAZAIDA 221FA19038, YADAVALL IMEENAKSHI-221FA19055 in partial fulfilment of Bachelor of Technology in the Department of Advanced Computer Science and Engineering, Vignans Foundation For Science Technology & Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India, is a bonafide work carried out by them under my guidance and supervision.

  
Guide

  
Dr. Venkatesulu Dondeti  
HoD/ACSE



**VIGNAN'S**

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

## **DECLARATION**

We hereby declare that our project work described in the field project titled “ORDER and ORDER\_ITEM relations” which is being submitted by us for the partial fulfilment in the department of ACSE, Vignan’s Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, and the result of investigations are carried out by us under the guidance of Dr. CH. Rose Rani.

|                            |            |
|----------------------------|------------|
| DAMARACHARLA GEETHA HARIKA | 221FA19012 |
| PALGULLA RANGASWAMI REDDY  | 221FA19015 |
| SYED SHANIKA ZAIDA         | 221FA19062 |
| YADAVALLI MEENAKSHI        | 221FA19065 |



## Table of Contents

1. Abstract
2. Introduction
3. Database Design
  - 3.1 Schema Overview
  - 3.2 Natural Join Result
4. Keys in Relation RES
5. Normalization Process
  - 5.1 Identification of Functional Dependencies
  - 5.2 Decomposition to 3NF
6. Resulting Schema and Analysis
7. Conclusion
8. References

## 1. Abstract

This document provides a thorough analysis of the order-processing application database for ABC, Inc., with a particular emphasis on the two key relations: ORDER and ORDER\_ITEM. These relations are crucial for managing customer transactions and ensuring accurate order fulfillment. By applying a natural join to these relations, we derive a new schema, referred to as RES, which integrates relevant data from both tables into a comprehensive view of each order and its associated items. This step is essential in streamlining the database structure and facilitating more efficient data retrieval processes.

As part of this analysis, we explore the keys within the RES schema, identifying primary keys that uniquely define each record and foreign keys that establish relationships between the integrated data. Understanding these keys is fundamental for maintaining referential integrity, which ensures that the data remains consistent and accurate across the database. Additionally, the document delves into the normalization process, advancing through the stages necessary to achieve Third Normal Form (3NF). This process involves identifying functional dependencies and systematically eliminating any redundancy or anomalies that may arise from the existing data structure.

The analysis highlights the critical importance of normalization in the database design process, emphasizing its role in ensuring data integrity and reducing redundancy. By adhering to the principles of normalization, the database design not only enhances performance but also simplifies maintenance and scalability. A well-normalized schema leads to improved data consistency and reliability, which are vital for the effective operation of ABC, Inc.'s order-processing application. Ultimately, this document aims to provide insights and recommendations for optimizing the database structure, thereby supporting the company's goal of efficient order management and exceptional customer service.

## 2. Introduction

In ABC, Inc.'s order-processing database, two primary relations play a crucial role in storing vital information about customer orders: ORDER and ORDER\_ITEM. The ORDER relation includes key attributes such as order number (O), order date (Odate), customer number (Cust), and total amount (Total\_amount). This relation captures essential details regarding each order, providing insight into customer transactions and facilitating the tracking of orders over time. Understanding this information is critical for managing customer relationships, analyzing purchasing patterns, and ensuring timely fulfillment of orders.

On the other hand, the ORDER\_ITEM relation offers detailed insights into individual items associated with each order. It comprises attributes such as order number (O), item number (I), quantity ordered (Qty\_ordered), total price (Total\_price), and item-specific discount (Discount%). This relation allows the company to monitor inventory levels, calculate total sales, and apply any discounts that may affect pricing. By managing these details effectively, ABC, Inc. can ensure accurate billing and maintain customer satisfaction through precise order fulfillment.

By applying a natural join to these relations, we can consolidate the information from both the ORDER and ORDER\_ITEM relations into a more comprehensive schema. This new schema provides a unified view of each order along with its associated items, enabling more efficient data retrieval and reporting. For instance, this consolidated view allows the company to quickly access all relevant details about an order, including the customer who placed it, the date it was made, and the specific items purchased. Such integration not only enhances the efficiency of the order-processing system but also supports better decision-making by providing a holistic perspective of customer transactions. Ultimately, this approach streamlines operations, enhances data management, and fosters improved customer service.

## 3. Database Design

### 3.1 Schema Overview

The original schemas for the relations are as follows:

- ORDER (O, Odate, Cust, Total\_amount)
- ORDER\_ITEM (O, I, Qty\_ordered, Total\_price, Discount%)

After applying the natural join, the resulting schema RES will be:

| O | Odate      | Cust | Total_amount | I    | Qty_ordered | Total_price | Discount% |
|---|------------|------|--------------|------|-------------|-------------|-----------|
| 1 | 2023-01-15 | 101  | 250.00       | 101  | 2           | 100.00      | 10        |
| 1 | 2023-01-15 | 101  | 250.00       | 102  | 1           | 80.00       | 5         |
| 2 | 2023-01-18 | 102  | 150.00       | NULL | NULL        | NULL        | NULL      |

### 3.2 Natural Join Result

The natural join combines records from the `ORDER` and `ORDER\_ITEM` relations based on the matching order number (O). This results in a relation RES with the following attributes:

- O
- Odate
- Cust
- Total\_amount
- I
- Qty\_ordered
- Total\_price
- Discount%

#### **4. Keys in Relation RES**

The primary key for the resulting relation RES is the composite key {O, I}. This key uniquely identifies each record, ensuring that each item in an order is associated with its corresponding order number.

## 5. Normalization Process

### 5.1 Identification of Functional Dependencies

To analyze the relation and determine its normalization, we must identify the functional dependencies:

1.  $O \rightarrow Odate, Cust, Total\_amount$  (An order number uniquely determines its date, customer number, and total amount.)
2.  $\{O, I\} \rightarrow Qty\_ordered, Total\_price, Discount\%$  (A combination of order number and item number uniquely determines the quantity ordered, total price, and discount.)

### 5.2 Decomposition to 3NF

To ensure the relation is in the Third Normal Form (3NF), we can decompose RES into two relations:

1. RES1 (Order Details):
  - Attributes: O, Odate, Cust, Total\_amount
  - Primary Key: O
2. RES2 (Order Item Details):
  - Attributes: O, I, Qty\_ordered, Total\_price, Discount%
  - Primary Key: {O, I}

This decomposition eliminates transitive dependencies and ensures that each relation adheres to 3NF.

## 6. Resulting Schema and Analysis

After normalization, the schemas for RES1 and RES2 are:

### RES1:

| O | Odate      | Cust | Total_amount |
|---|------------|------|--------------|
| 1 | 2023-01-15 | 101  | 250.00       |
| 2 | 2023-01-18 | 102  | 150.00       |

### RES2:

| O | I   | Qty_ordered | Total_price | Discount% |
|---|-----|-------------|-------------|-----------|
| 1 | 101 | 2           | 100.00      | 10        |
| 1 | 102 | 1           | 80.00       | 5         |

This design optimizes data organization and adheres to normalization principles, ensuring data integrity and reducing redundancy.

## 7. Conclusion

The analysis of the natural join between the ORDER and ORDER\_ITEM relations results in a comprehensive schema RES. By identifying keys and normalizing the schema up to 3NF, we ensure data integrity, efficiency, and adherence to best practices in database design. This well-structured and normalized design is ideal for ABC, Inc.'s order-processing application, enhancing the reliability and efficiency of order management.



## 8. References

1. Date, C. J. (2004). *\*An Introduction to Database Systems\**. Addison-Wesley.
2. Elmasri, R., & Navathe, S. B. (2015). *\*Fundamentals of Database Systems\**. Pearson.
3. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). *\*Database System Concepts\**. McGraw-Hill.