Cambodia Academy of Digital Technology

Institute of Digital Technology

Faculty of Digital Engineering

Department of Computer Science

Database Administration Report

lecturer Thear Sophal

Submitted By

Cheng Chanpanha	(IDTB100070)
Chhun Sivheng	(IDTB100035)
Choun Rathanak	(IDTB100043)
Lim Lyheang	(IDTB100001)
Sat Panha	(IDTB100033)
Phy Vathanak	(IDTB100037)

TABLE OF CONTENTS

I. INTRODUCTION

1.1 Background of the project

With the increasing demand for computers and accessories, the manager has decided to create a database to manage the computer shop more effectively. Basic spreadsheets often lead to issues such as poor performance tracking. This project will build a database system for a computer shop that will include inventory, sales, users, user performance, backup and recovery, and user interface to support shop operation.

1.2 Objective of study

The primary of the project is to design and implement a database system for computer shop operations, support inventory management, sales tracking, user access control, and apply advanced SQL techniques including view, Triger, Procedure, and functions. It also includes managing data access using Design role-based access control (RBAC), creating and testing backup and recovery strategy, analyze and compare system performance under different environments and developing a simple web interface for interacting with the database.

1.3 Task Responsibility

Requirement Area	Tasks	Assigned Member(s)	Notes / Details
Database Design	- Create ER Diagram and Relational Schema	Sat Panha	Comprehensive design and detailed report
Database Implementation (DDL)	- Write DDL scripts	Cheng Chanpanha, Lim Lyheang	Collaborate on schema creation, view , trigger , procedure
Data Population	- Develop and run data generation & insertion scripts	Phy Vathanak	support from Chhun Sivheng and Lim Lyheang
User Management & Access Control	- Design user/role model and RBAC documentation	Chhun Sivheng	Document design and practical scenario
	- Backend implementation of user/role management	Phy Vathanak	Backend data flow and logic
	- Frontend web app for user/role management	Choun Rathanak	UI design, including fonts and layout
Backup and Recovery Strategy	- Document backup and recovery plan	Choun Rathanak	Create detailed plan
	- Write backup and recovery scripts	Phy Vathanak	Automate backup and recovery
	-Backend and front recovery	Phy Vathanak	Data flow and logic
Query Development & Optimization	- Write complex SQL queries	Cheng Chanpanha ,Sat Panha	Queries with joins, subqueries, aggregation
Performance Comparison	- Design and implement indexing strategies	Cheng Chanpanha	Lead indexing

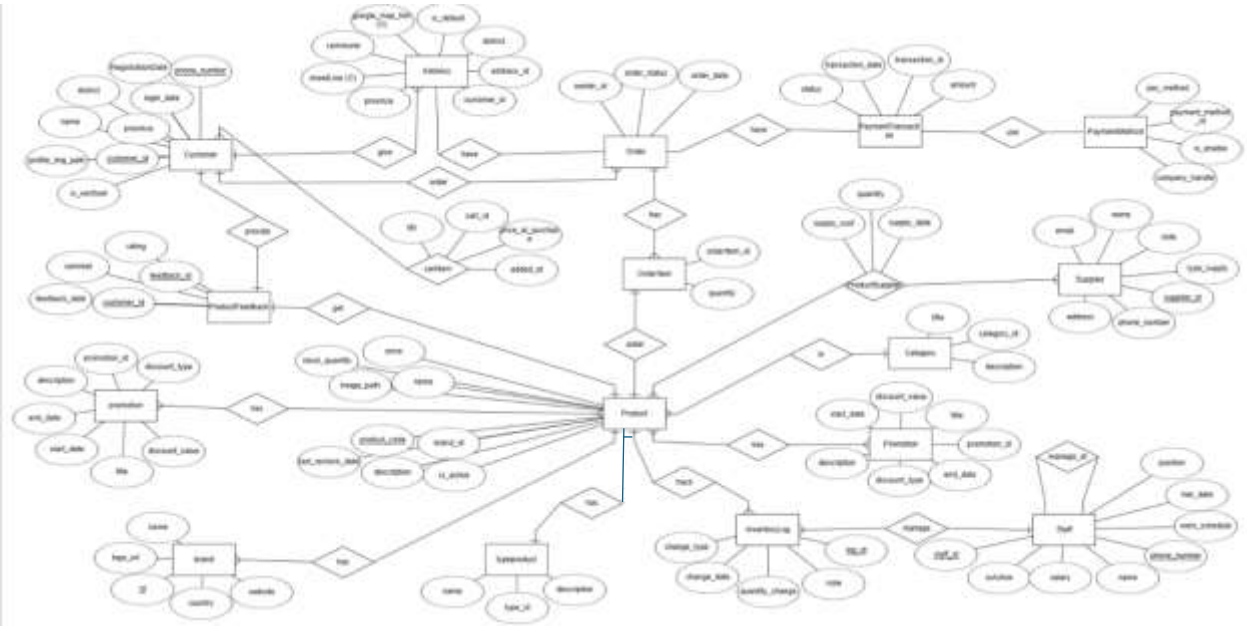
	- Conduct tests and compare performance	Cheng Chanpanha, Phy Vathanak	Run tests and document
	- Setup local server for performance testing	Cheng Chanpanha	Configure and test environment
Document Preparation	- Compile all design, implementation, and testing documentation	All	Collaborative document covering all project phases

II. Database Design

ER Diagram

The ER diagram models the core functionalities of a computer shop management system. It includes the following key components:

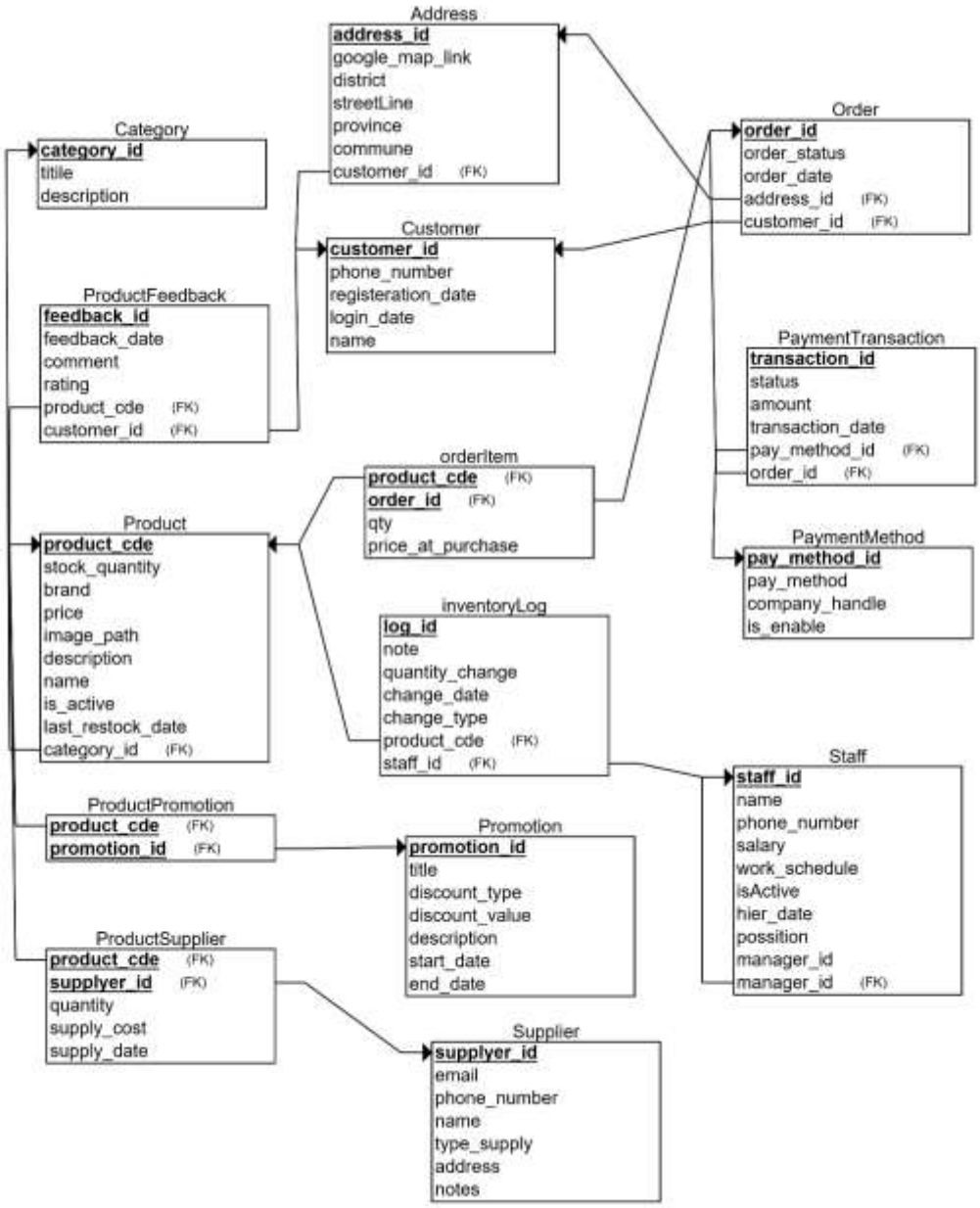
- **Customer & Address**: Customers can save multiple addresses and place multiple orders.
- **Order Management**: Orders link to cart items and order items, tracking products and quantities purchased.
- **Product**: Central entity with attributes like price, stock, and relationships to categories, brands, types, and promotions.
- **Inventory Control**: Inventory logs track stock changes over time (e.g., restocking, sales).
- **Promotions & Feedback**: Products can have promotions and receive customer feedback (ratings/comments).
- **Supplier Management**: Products are supplied by various suppliers, with records of quantity and cost.
- **Payment System**: Payments are handled via different payment methods, with detailed transaction records.
- **Staff Management**: Staff handle operations with assigned roles, salaries, and schedules.



Schema Diagram

The schema represents a comprehensive system for managing a computer shop:

- **Customer & Address**: Customers can have multiple addresses and place orders.
- **Order & Payment**: Orders track items and link to payment transactions and methods.
- **Product Management**: Products include details like stock, price, category, and brand. They can have promotions and be supplied by vendors.
- **Inventory Control**: Logs track stock changes and are managed by staff.
- **Feedback & Suppliers**: Customers give feedback on products. Suppliers provide product stock with cost and supply date.
- **Staff**: Handles operations with defined roles, hierarchy, and schedules.



III. Data Definition Language

- We used **Data Definition Language (DDL)** to design and structure all necessary database objects. DDL plays a critical role in defining the **schema, constraints, relationships**, and **data integrity rules** of the system. We have create using Key DDL Operations Performed like create , alter ,update ... We also using those to prepare some for the query with complex data to test our database. There are:
- 20 **triggers**
- 10 **Views**

- 10 **Stored procedures**
- 10 **User defined functions**

1. Stored Procedures

Procedure Name	Purpose
GetCustomerOrderHistory	Fetches complete customer order and payment history.
GenerateMonthlySalesSummar	Summarizes monthly sales by product category.
RestockProduct	Automates restocking and logs supply from suppliers.

2. Functions

Functions Name	Purpose
countFeedbackByProduct	to count feed for each product.
calculateOrderTotal	To calculate the order item .
getCustomerTotalSpending	To calculate all the amount the customer had been spend

3. Views

View Name	Description
staffPerformance	Shows order value and count per active staff member.
paymentMethodUsage	Summarizes transaction counts and totals by payment method.
customerFeedbackInsights	Aggregates product feedback (average ratings, feedback count).

4. Triggers (Audit Logging)

- A **trigger-based audit log system** records changes to critical tables (customers, products, orders) in an isolated audit_log table.
- Triggers include:
  - customerDelete – tracks account deletions
  - customerUpdate – tracks updates to customer info
  - staffUpdate – tracks staff info changes

IV. Data population

To test the performance and reliability of the Computer Shop Database System, a large volume of dummy data was generated and inserted into the database. A total of 16 million records were populated across key tables such as customer, orders, orderitem, and paymenttransaction.

- **SQL Script Insertion:** We used **SQL insert scripts** for small, static, or reference tables such as product categories, fixed product types, and admin users.
- Python Script Insertion: For large-scale data generation, we developed **Python scripts for** Automatically created random data and use indexing for optimized speed and stability.

Here is our data set that we have populated

Table	Data	Table	Data
address	140,000	paymenttransaction	3,000,000
brand	16	product	16,000
paymentmethod	5	productfeedback	80,000
category	9	productpromotion	3,000
customer	140,000	productsupplier	
inventorylog		promotion	6
orderitem	10,501,623	staff	30
orders	3,000,000	supplier	30

V. **USER GRANT AND ACCESS**

1. User role and privileges

- We defined 7 roles based on responsibilities in the system:
- **Database admin**: Full control with ALL PRIVILEGES and GRANT OPTION.
- **Lead developer**: Full development rights including CREATE, ALTER.
- **Senior backend developer**: Full data access (CRUD) on all tables.
- **DevOps engineer**: Limited to SELECT, ALTER for monitoring and schema changes.
- **Junior backend developer**: SELECT, INSERT, UPDATE on product-related tables.
- **QA validation engineer** & **Data Analyst**: Read-only (SELECT) on all tables.

These roles align with real-world responsibilities and limit unnecessary access.

2. User Account

Username	**Role**	**Privileges Summary**
sok_dara	Database Admin	Full access with ability to grant permissions
sat_panha	Lead Developer	Full development privileges (DDL + DML)
phy_vathanak	Senior Backend Developer	Full data manipulation (CRUD)
choun_rathanak	DevOps Engineer	Schema monitoring (SELECT, ALTER only)
cheng_chanpanha	Junior Backend Developer	Can view/modify product data (SELECT, INSERT, UPDATE only)
chhun sivheng	QA/Validation Engineer	Read-only access
lim_lyheang	Data Analyst	Read-only access

To simplify administration, we also designed a dedicated web interface for managing user roles and accounts. This interface allows administrators to:

- Create new roles with customized sets of privileges
- Assign or revoke privileges to specific roles
- Create and manage user accounts
- Assign users to predefined roles
- View role-based access summaries

The system ensures that all user actions are validated against predefined rules to prevent privilege escalation or unauthorized access. By centralizing role and user management through a web interface, we improved both usability and security, enabling easier delegation and auditing for database administrators.

VI. Backup and Recovery Strategies

This document outlines a comprehensive Backup and Recovery Strategy for the Computer Shop Management System, which stores critical business data including:

- Customer records
- Product inventory
- Sales transactions
- User accounts
- Payment history
- more

The goal is to ensure **data integrity**, **minimal downtime**, and **reliable recovery** in case of system failure, hardware crash, or human error.

1. Backup Types and Schedules

Backup Type	What It Backs Up	Recovery Requirement	Tools Used
Full Backup	Entire shop system database and settings	Restore only the latest full backup	mysqldump, system snapshots
Differential Backup	Data changed since the last full backup	Full backup + latest differential backup	mysqlbinlog, custom scripts
Incremental Backup	Only the changes since the last backup (any type)	Full backup + all incrementals after it	mysqlbinlog, binary log manager

2. Backup Workflow

1. **Full Backup** every Sunday 12 AM backup everything in the system (all tables, users, inventory, logs, etc.).
2. **Differential Backup** every 8 Hour at 2 AM, 10AM ,6PM every day saves changes since last full backup.
3. **Incremental Backup** every day at 6 AM only new changes since the last backup (keeps up with sales, stock updates, etc.).

We implemented automated database backups using Windows Task Scheduler. Each backup is stored in a uniquely named folder based on its type, associated database, and timestamp. The system includes a web interface that allows users to restore from any backup type. To manage storage efficiently, the system retains only the latest five backups and automatically deletes backups older than four weeks.

3. Use Case Scenarios

Scenario A: Inventory Data Corruption (Use Differential Backup)

- **Issue**: At 3 PM on Wednesday, staff accidentally deletes several inventory items while importing supplier data.
- **Recovery Plan**:
  - Restore **Sunday's full backup**.
  - Apply **Wednesday 2 AM differential backup**.
- **Result**: Inventory is restored up to that morning, and the corrupted changes are removed.

- **Scenario B: Database Crash Due to Power Loss (Use Incremental Backup)**

- **Issue**: The power goes out at 10:45 PM on Friday. Recent sales data is lost.
- **Available**:
  - Full backup: Sunday 12 AM
  - Incremental: Friday 6 AM
- **Recovery Plan**:
  - Restore Sunday's full backup.
  - Apply all incremental backups for Friday.
- **Result**: System is recovered up to the last incremental at 10 PM. Minimal data loss.

- **Scenario C: Complete Server Failure (Full Recovery Path)**
- **Issue**: On Thursday, the server's hard drive fails completely.
- **Recovery Plan**:
  - Set up a new server or cloud environment.
  - Restore the **Sunday full backup**.
  - Apply **Thursday 2 AM differential backup**.
  - Apply **Thursday incremental backups** (Since Monday).
- **Result**: Full restoration of shop system, up to Thursday 2:00 PM.

4. Technical Implementation

- **Backup Tools**:
  - mysqldump for full database exports
  - mysqlbinlog to generate incremental backups using binary logs
  - Scheduled via task scheduler automation
- **Storage Strategy**:
  - Backups saved in local storage
  - Retention policy: 5 full backup (weekly), 105 differential, 35 incremental
- **Monitoring**:
  - log on backup success/failure
- **Framework for restore**

- Node JS (Express) for handle backend request and using react js for webpage that provide restore feature
- Python script for create backup and restore

VII. Index and Query Performance

To ensure the **Computer Shop Management System** can handle complex operations and large volumes of data efficiently, we implemented an advanced database logic layer combined with a well-planned indexing strategy using both sub query and join. This included the creation of:

- 7 index
- Using left join and right join
- Using With query

All of these were built using **subqueries**, **JOINs**, and **conditional logic** to support real-world business operations like customer management, inventory tracking, product feedback, and sales analytics.

1.Performance Optimization(Index)

we implemented targeted **indexing** To support these complex operations. Here are some of index we have create:

- Customer (phoneNumber): Fast lookup for customer updates/delete
- Product(name, category):Quick search and filter
- **Trigger efficiency**, particularly during high-volume INSERT/UPDATE operations.
- Reduced I/O load and improved user experience on front-end reports and dashboards.

With over **10 million records** in the database, these indexes drastically improved:

- **JOIN performance**, especially in views and stored procedures.
- **Subquery execution time**, by reducing full table scans.

N0	Summary	Time No Index (second)	Time With Index	Row
1	Summarizes total number of orders and total amount spent by each customer.	165.68152	70.66715	140000
2	Shows total quantity sold and total revenue generated for each product.	2.46087	0.364402	1700
3	Shows staff managing orders, with total order value handled and number of orders, for active staff only.	61.2373	0.0598	29
4	Analyzes payment method usage, showing transaction counts and total amounts by method.	24.20135	4.379647	5

5	Evaluates promotions by calculating total sales under each promotion and affected products.	2822.2702	1677.1416	3000037
6	Summarizes product feedback, including average rating and feedback count per product.	4.59508	0.274491	15883
7	Identifies categories with the highest total sales revenue.	1973.16096	13.40897	7
8	Provides detailed information for all orders that are currently in 'Pending' status.	43.80282	22.313472	1001372
9	Find the top 5 best-selling products by total quantity sold	1951.40894	4.89547	5
10	Lists products with low stock quantity (less than 10) and their last restock date.	0.293725	0.055852	3809

VIII. Database Server

We deployed and configured the MySQL Database Server to serve as the core data platform for our Computer Shop Management System. The database server setup was designed with scalability, performance, and multi-user access in mind.

1. Server Setup & Configuration

The MySQL server was installed on a Windows-based machine and configured for local network access. Static IP assignment was used to ensure consistent availability across devices. Network settings and firewall rules were configured to allow remote access to the MySQL server via port **3306**.

2. strategy

- Find Main computer Local IP Address.
- Allow Firewall Access
- Using same network
- We host sever by using one computer open sql in thier cmd .
- We have to create user for the other so they can use the database.
- The people that have user have to go to thier cmd and using like this

```
mysql -u sat_panha -p -h <YOUR_SERVER_IP>
# Enter password: 1234
```

IX. Challenge and Learning outcomes

During the development of the Computer Shop Database System, we encountered several challenges:

- Complex and Performance-Intensive Queries:Writing efficient JOINs, subqueries, and aggregation queries required careful indexing and testing to reduce execution time on a dataset of over 10 million records was a significant challenge.
- Backup Strategy Planning: Designing a reliable and automated backup strategy was essential to prevent data loss.
- Algorithmic Thinking for Data Generation:Maintain **logical relationships** between tables and Ensure **data distribution** is realistic also Avoid breaking **foreign key constraints**

Learning outcomes

What we have learned from this project:

- Database Hosting and Server Configuration: We learned how to host a database server on a local network and configure access securely using user privileges and firewall rules.
- Query Optimization Techniques:By analyzing slow queries with and applying proper indexing and schema design, we drastically improved performance.
- **Data Loss Prevention and Recovery:**
- We practiced creating and restoring full backups, which reinforced the importance of disaster recovery planning.
- Designing Role-Based Permissions: analyst each user and their role what kind of permission should we give them for thier task.
- Python-Based Data Generation: This enhanced our understanding of **data simulation**, scripting, and the challenges of maintaining data integrity during bulk inserts.

X. CONCLUSION

This project successfully developed a comprehensive database system for managing a computer shop, incorporating key features such as user management with role-based access

control, backup and recovery strategies, query optimization, and integration with a web application.

Key challenges included modeling complex data relationships, optimizing reusable queries for performance, and implementing robust backup and recovery mechanisms. Overcoming these challenges strengthened our understanding of database administration and emphasized the importance of secure architecture, performance tuning, and data protection in real-world systems.

XI. References

1. MySQL. (n.d.). *MySQL 8.0 Reference Manual*. Oracle Corporation. Retrieved from https://dev.mysql.com/doc/
2. Express.js. (n.d.). *Express – Node.js web application framework*. Retrieved from https://expressjs.com/
3. JWT.io. (n.d.). *JSON Web Tokens Introduction*. Retrieved from https://jwt.io/introduction/
4. Microsoft. (2023). *Role-Based Access Control (RBAC): Best Practices*. Retrieved from https://learn.microsoft.com/en-us/azure/role-based-access-control/best-practices
5. IBM. (n.d.). *Database backup and recovery best practices*. IBM Knowledge Center. Retrieved from https://www.ibm.com/docs/en/db2
6. Stackify. (2023). *SQL Performance Tuning: 7 Practical Tips for Developers*. Retrieved from https://stackify.com/sql-performance-tuning-tips/
7. W3Schools. (n.d.). *SQL Tutorial*. Retrieved from https://www.w3schools.com/sql/
8. DigitalOcean. (2022). *How to Back Up and Restore MySQL Databases on Ubuntu 20.04*. Retrieved from https://www.digitalocean.com/community/tutorials/how-to-back-up-mysql-databases-on-ubuntu-20-04
9. Oracle. (n.d.). *MySQL Backup and Recovery*. Oracle Documentation. Retrieved from https://dev.mysql.com/doc/mysql-backup-excerpt/8.0/en/
10. PostgreSQL Global Development Group. (n.d.). *Views, Triggers, and Stored Procedures in PostgreSQL*. Retrieved from https://www.postgresql.org/docs/current/