

# **BUSINESS DATABASE IMPLEMENTATION REPORT**

Database - CMP020C106S COURSEWORK 2

Evan Balson: BAL18466416

**The contribution from my partner has  
been removed to maintain their privacy**

Roehampton University London

Lecturer: Dr. Wei Li

April 24 2024

## Table of Contents

Abstract.....	3
Introduction: .....	4
Methodology.....	5
CREATE DATABASE .....	5
CREATE TABLE .....	5
Linkage Tables:.....	6
INSERT .....	7
Testing and Validation .....	8
Test 1: Data Integrity Test Case: .....	8
Test 2: Query Performance Test Case:.....	9
Evan's Conclusion & Reflection.....	10
Partner's Contribution.....	11
References .....	12

## Abstract

This report documents the development journey of Visual Studio Prime, a database tailored for a progressive movie theatre enterprise. Focused on pragmatic implementation, the project explores the utilization of Data Definition Language (DDL) statements, a standardized language with commands to define the storage groups, structures and objects in a database.<sup>[1]</sup>

Central to this endeavour is the meticulous process of creating tables in alignment with the database's physical model. This entails establishing precise constraints, including data types and keys, and populating tables with meaningful sample data. Each step is carefully detailed to provide insights into the meticulous process of database implementation.

By elucidating these processes, this report seeks to offer a practical understanding of database management principles. It serves as a roadmap for aspiring database developers, guiding them through the nuanced terrain of database design and implementation.

## Introduction:

Data, often lauded as one of the world's most valuable resources, lies at the heart of modern enterprise operations. With the advent of sophisticated technologies, the need to effectively organize and manage vast collections of data has become imperative, giving rise to purposeful databases. But what exactly is a database?

According to GeeksforGeeks, a database is a structured collection of data organized into tables comprising rows and columns. These tables are meticulously indexed, facilitating seamless data manipulation, expansion, and deletion. For a dynamic enterprise like Visual Studio Prime, operating in a high-activity environment with multiple branches, the significance of a robust database cannot be overstated.<sup>[2]</sup>

Manual or offline tracking of operations in such a bustling enterprise not only data integrity risks and inefficiency but also opens the door to potential legal ramifications due to negligence. A well-designed database serves as a linchpin in streamlining operations and enhancing customer experiences. By centralizing data storage and management, Visual Studio Prime can optimize resource allocation, minimize operational costs, and mitigate the risks associated with manual tracking methods.

In an era characterized by heightened customer expectations and the imperative for operational efficiency, the role of robust database management systems emerges as paramount. However, implementing a database is far from a simple endeavour. It presents a myriad of challenges, ranging from time constraints and collaborative efforts to scalability and usability concerns. While these challenges may pose hurdles along the development journey, they do not diminish the inherent value of a database for business operations. Nonetheless, addressing these challenges proactively is essential to ensure the timely delivery and efficacy of the end product.

In the subsequent sections of this report, we delve into the fundamental concepts of database design and implementation for Visual Studio Prime, shedding light on key elements such as primary keys, constraints, and data types.

## Methodology

Continuing from the comprehensive methodology outlined in the first document “Business Database Design 2024 - Coursework 1 - Final Report (1)”, we proceed to the database design and implementation phase for Visual Studio Prime. Building on the concepts established during our requirements analysis and database design, we are now equipped to translate the design specifications into a tangible database schema using Data Definition Language (DDL) statements.

This phase leveraged a key component developed during the database design phase: the physical data model. Crafted specifically for the technical implementation of the database, this model served as the blueprint for constructing the database schema, defining tables, columns, indexes, and other essential database objects.

Following the guidance provided by the physical data model, we transitioned into scripting a series of Data Definition Language (DDL) instructions to create the database and its associated tables. Firstly, we initiated the database creation process using the following DDL command: <sup>[3]</sup>

```
CREATE DATABASE visual_studio_prime;
```

Subsequently, we proceeded to create the core tables essential for the functionality of Visual Studio Prime. A pivotal table, such as the transaction table, was created to record all transactions made within the system.<sup>[3]</sup>

```
CREATE TABLE transaction (
```

```
    Transaction_ID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    Employee_ID VARCHAR(10),
```

```
    Customer_ID VARCHAR(10),
```

```
    Transaction_Date DATE,
```

```
    Amount DECIMAL(10,2)
```

```
);
```

	Transaction_ID	Item_Purchased	Purchase_Date	Total_Paid	Customer_ID	Offer_ID	Film_ID
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	12	2 movie tickets (Black Widow)	2024-04-26	25.00	12345670	4	BW003
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	13	3 movie tickets (Dune)	2024-04-26	36.00	23456701	5	DU002
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	14	1 movie ticket (Fight Club)	2024-04-27	12.50	34567890	2	FC009
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	15	2 movie tickets (Ghostbusters: Frozen Empire)	2024-04-27	20.00	45670123	6	GF017
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	16	1 movie ticket (The Silence of the Lambs)	2024-04-28	15.00	56789012	3	SL008
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	17	2 movie tickets (Abigail)	2024-04-28	25.00	67890123	7	AB011
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	18	3 movie tickets (Back to Black)	2024-04-29	36.00	78901234	10	BB012
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	19	1 movie ticket (Kung Fu Panda 4)	2024-04-29	12.50	89012345	8	KF014
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	20	2 movie tickets (Forrest Gump)	2024-04-30	20.00	90123456	1	FG006
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	21	1 movie ticket (The Godfather)	2024-04-30	15.00	90123456	9	TG004
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	22	3 movie tickets (The Matrix)	2024-05-01	36.00	78901234	10	TM007

**Figure. 1** Showing the transaction table created with the SQL code.

In addition to the transaction table, other auxiliary tables were also created to establish crucial linkages within the database schema. For instance, the payment method table was designed to represent the various payment methods utilized in transactions. Each payment method is associated with a specific transaction through the transaction ID, ensuring accurate tracking and reporting of payment details.<sup>[3]</sup>

```
CREATE TABLE payment_method (
    Payment_ID INT AUTO_INCREMENT
PRIMARY KEY,
    Transaction_ID INT,
    Payment_Method VARCHAR(20),
FOREIGN KEY (Transaction_ID)
REFERENCES
transaction(Transaction_ID)
);
```

	Payment_ID	Payment_Type	Transaction_ID
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	12	Credit Card	12
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	13	Debit Card	13
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	14	Cash	14
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	15	Credit Card	15
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	16	Debit Card	16
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	17	Credit Card	17
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	18	Debit Card	18
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	19	Cash	19
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	20	Credit Card	20
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	21	Cash	21
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	22	Credit Card	22

**Figure. 2** Showing the payment method linkage table for transactions

### Linkage Tables:

To maintain relational integrity and capture transactional nuances, linkage tables were created. For example, the employee transaction linkage table establishes associations between employees, customers, and transactions. This linkage ensures comprehensive tracking of transactions involving employees and customers.<sup>[3]</sup>

	Employee_ID	Transaction_ID
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	AJ1234567	12
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	AS2034567	16
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	DS9876543	14
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	EB2345678	13
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	GA3234567	17
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	GH2345654	18
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	JS9876734	21
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	JT1023456	19
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	LR2345654	15
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	LR3456789	20
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	MS7879098	22

```
CREATE TABLE employee_transaction (
    Transaction_ID INT AUTO_INCREMENT PRIMARY KEY,
    Employee_ID VARCHAR(10),
    Customer_ID VARCHAR(10),
    Transaction_Date DATE,
    Amount DECIMAL(10,2),
FOREIGN KEY (Employee_ID)
REFERENCES employee(Employee_ID),
FOREIGN KEY (Customer_ID)
REFERENCES customer(Customer_ID)
);
```

**Figure. 3** Showing the employee transaction linkage table for the transactions table.

These tables are interconnected, representing related entities within the system. The transaction table records all transactions made by employees with customers, encompassing crucial details such as transaction ID, employee ID, customer ID, transaction date, and amount. Similarly, the payment method table captures diverse payment methods associated with each transaction, facilitating accurate tracking and reporting of payment details.

Using the methods above, we were able to successfully implement and populate not only the tables mentioned above but also the categories, customer, department, employee, employee\_transaction, films, film\_category, license, license\_cinema\_department, membership, offers, payment\_method, suppliers, theme\_stores, ticket, transactions, transaction\_theme\_store, and vs\_cinema\_branches.

<input type="checkbox"/>	categories	★							11	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/>	customer	★							15	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/>	department	★							15	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/>	employee	★							32	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/>	employee_transaction	★							11	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/>	films	★							20	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/>	film_category	★							20	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/>	license	★							20	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/>	license_cinema_department	★							20	InnoDB	utf8mb4_general_ci	48.0 KiB	-
<input type="checkbox"/>	membership	★							8	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/>	offers	★							10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/>	payment_method	★							11	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/>	suppliers	★							10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/>	theme_stores	★							15	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/>	ticket	★							11	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/>	transactions	★							11	InnoDB	utf8mb4_general_ci	64.0 KiB	-
<input type="checkbox"/>	transaction_theme_store	★							11	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/>	vs_cinema_branches	★							15	InnoDB	utf8mb4_general_ci	16.0 KiB	-
18 tables		Sum							266	InnoDB	utf8mb4_general_ci	496.0 KiB	0 B

**Figure. 4** Showing all tables that were created for Visual Studio Prime's Database.

		Payment_ID	Payment_Type	Transaction_ID
<input type="checkbox"/>		12	Credit Card	12
<input type="checkbox"/>		13	Debit Card	13
<input type="checkbox"/>		14	Cash	14
<input type="checkbox"/>		15	Credit Card	15
<input type="checkbox"/>		16	Debit Card	16
<input type="checkbox"/>		17	Credit Card	17
<input type="checkbox"/>		18	Debit Card	18
<input type="checkbox"/>		19	Cash	19
<input type="checkbox"/>		20	Credit Card	20
<input type="checkbox"/>		21	Cash	21
<input type="checkbox"/>		22	Credit Card	22

**Figure. 5** Demonstrating the result of using the insert SQL code to add valued to the payment method table.

### Inserting data into the payment\_method table

**INSERT INTO** payment\_method  
(Payment\_ID, Payment\_Type,  
Transaction\_ID)

**VALUES**

(12, 'Credit Card', 12),  
(13, 'Debit Card', 13),  
(14, 'Cash', 14),  
(15, 'Credit Card', 15),  
(16, 'Debit Card', 16),  
(17, 'Credit Card', 17),  
(18, 'Debit Card', 18),  
(19, 'Cash', 19),  
(20, 'Credit Card', 20),  
(21, 'Cash', 21),  
(22, 'Credit Card', 22);

## Testing and Validation


During the testing phase, we meticulously validate the functionality and performance of the database. Through a series of rigorous test cases, we assess data integrity and query performance.

Each test case is meticulously executed to identify any discrepancies or issues. Prompt action is taken to address these findings through iterative refinements, ensuring that the database meets the specified requirements and operates smoothly.



### Test 1: Data Integrity Test Case:

**Objective:** Verify that data integrity is maintained within the database, ensuring that all data entered adheres to defined constraints and rules.

1. Insert a new transaction record into the **employee\_transaction** table with invalid data (e.g., a non-existent Employee\_ID or Customer\_ID).

<b>Error</b>	<b>The SQL Statement Executed:</b>
SQL query: <a href="#">Copy</a>	INSERT INTO employee_transaction
	(Employee_ID,
	Customer_ID,
	Transaction_Date,
	Amount)
	VALUES ('6516515555',
	'customer12222',
	'2024-04-20', 100.00);
MySQL said: 	
#1054 - Unknown column 'Customer_ID' in 'field list'	

2. Attempt to execute a SELECT query to retrieve the inserted record.

<b>Error</b>	<b>The SQL Statement Executed:</b>
SQL query: <a href="#">Copy</a> 	SELECT *
	FROM employee_transaction
	WHERE Employee_ID = '22223355'
	AND Customer_ID = ' 12345670' LIMIT 0, 25
MySQL said: 	
#1054 - Unknown column 'Customer_ID' in 'where clause'	

**Result:** The database rejects the insertion of invalid data and return an error message, indicating a violation of data integrity constraints.



## Test 2: Query Performance Test Case:

**Objective:** Evaluate the performance of database queries, particularly those commonly executed by users, to ensure acceptable response times.

3. Execute a SELECT query to retrieve a large dataset from the **employee\_transaction** table.
4. Record the time taken to execute the query.
5. Repeat the same query multiple times to measure consistency in performance.

### The SQL Statement executed:

```
SET @start_time = NOW();
```

```
SELECT * FROM employee_transaction WHERE Transaction_Date BETWEEN '2024-01-01'  
AND '2024-12-31';
```

```
SET @end_time = NOW();
```

```
SELECT TIMEDIFF(@end_time, @start_time) AS Execution_Time;
```

✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)								
SET @start_time = NOW();								
[ Edit inline ] [ Edit ] [ Create PHP code ]								
⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available. ⓘ								
✔ Showing rows 0 - 10 (11 total, Query took 0.0005 seconds.)								
SELECT * FROM transactions JOIN employee_transaction ON transactions.Transaction_ID = employee_transaction.Transaction_ID WHERE Purchase_Date BETWEEN '2024-01-01' AND '2024-12-31';								
<input type="checkbox"/> Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]								
Transaction_ID	Item_Purchased	Purchase_Date	Total_Paid	Customer_ID	Offer_ID	Film_ID	Employee_ID	Transaction_ID
12	2 movie tickets (Black Widow)	2024-04-26	25.00	12345670	4	BW003	AJ1234567	12
16	1 movie ticket (The Silence of the Lambs)	2024-04-28	15.00	56789012	3	SL008	AS2034567	16
14	1 movie ticket (Fight Club)	2024-04-27	12.50	34567890	2	FC009	DS9876543	14
13	3 movie tickets (Dune)	2024-04-26	36.00	23456701	5	DU002	EB2345678	13
17	2 movie tickets (Abigail)	2024-04-28	25.00	67890123	7	AB011	GA3234567	17
18	3 movie tickets (Back to Black)	2024-04-29	36.00	78901234	10	BB012	GH2345654	18
21	1 movie ticket (The Godfather)	2024-04-30	15.00	90123456	9	TG004	JS9876734	21
19	1 movie ticket (Kung Fu Panda 4)	2024-04-29	12.50	89012345	8	KF014	JT1023456	19
15	2 movie tickets (Ghostbusters: Frozen Empire)	2024-04-27	20.00	45670123	6	GF017	LR2345654	15
20	2 movie tickets (Forrest Gump)	2024-04-30	20.00	90123456	1	FG006	LR3456789	20
22	3 movie tickets (The Matrix)	2024-05-01	36.00	78901234	10	TM007	MS7879098	22

**Result:** The query executed within an acceptable time frame, demonstrating efficient query performance. Consistency in execution time across multiple runs indicates stable performance.

## Evan's Conclusion & Reflection

As part of my responsibilities, I was tasked with populating half of the database's tables with data, particularly the transaction table, while also proofing other table data to ensure integrity. Inserting data into seven tables closely related to the transaction is part of this phase. Test 1: Data Integrity Test Case: Objective: Verify that data integrity is maintained within the database, ensuring that all data entered adheres to defined constraints and rules. Result: The database rejects the insertion of invalid data and returns an error message, indicating a violation of data integrity constraints. Further detail is available in the methodology and testing phase for each part.

Throughout this process, we encountered several challenges, highlighting areas for improvement:

Our initial aim was to construct a robust database structure tailored for Visual Studio Prime. However, the design was based primarily on a theoretical understanding of a movie theatre company's operations. In a real-world scenario, refining our relationships to tightly incorporate unmentioned attributes is essential for optimal database design.

To maintain focus on functionality within a smaller relation set, we intentionally scaled down the database. Despite its reduced size, the intricate connections between tables demanded meticulous data entry. Even a minor discrepancy, such as a key mismatch, could lead to complex resolution procedures, potentially compromising data accuracy and integrity.

Incorporating normalization principles during the design phase greatly facilitated implementation. Normalization, a process aimed at eliminating redundancies and maintaining consistency, proved instrumental in developing a medium-sized database.

Comprehensive documentation, including our brief, requirements outline, Entity-Relationship Diagram (ERD), and Physical Data Model (PDM), proved invaluable during implementation. These resources streamlined the process, ensuring clarity and alignment with project goals.

Leveraging SQL facilities to enhance database integrity proved to be a tedious yet necessary task. Implementing SQL constraints and validations contributed significantly to the overall usability and reliability of the project.

Although the database underwent rigorous testing by Evan, the test results may not encompass all potential scenarios. Further refinement through diverse workloads could offer deeper insights into the durability and reliability of our database design.

Time management and collaboration emerged as significant concerns. Working on a local host restricted database access to the designing system. However, leveraging PHPMYADMIN server allowed for seamless database export and import, enabling efficient collaboration and task management across multiple machines.

## Partner's Conclusion & Reflection

**The contribution from my partner has been removed to maintain their privacy**

## References

- [1] TechTarget. "Data Definition Language (DDL) - Definition", TechTarget. Available: <https://www.techtarget.com/whatis/definition/Data-Definition-Language-DDL#:~:text=DDL%20is%20a%20standardized%20language%20with%20commands%20to,database%20objects%2C%20such%20as%20tables%2C%20indexes%20and%20stogroups>. Accessed: [04, 01, 2024].
- [2] GeeksforGeeks. "What is Database?", GeeksforGeeks. Available: <https://www.geeksforgeeks.org/what-is-database/>. Accessed: [04, 20, 2024].
- [3] GeeksforGeeks. "SQL Cheat Sheet", GeeksforGeeks. Available: <https://www.geeksforgeeks.org/sql-cheat-sheet/>. Accessed: [04, 16, 2024].
- [4] "SQL Foreign Key Constraint", W3 Schools, [SQL FOREIGN KEY Constraint \(w3schools.com\)](https://www.w3schools.com/sql/sql_foreign_key_constraint.asp) , [Accessed: April 11, 2024].
- [5] "Relational Data Modeling Many-to-Many Relationship (Two-way-relationship)", DataCadamia, [Relational Data Modeling - Many-to-many Relationship \(Two-way relationship\) \(datacadamia.com\)](https://datacadamia.com/relational-data-modeling-many-to-many-relationship-two-way-relationship/) [Accessed: April 17, 2024]
- [6] Odeon, [ODEON Cinemas | Book Cinema Tickets & Watch New Movies](https://www.odeon.dk/) [Accessed: April 12, 2024]