



**BERLIN SCHOOL OF  
BUSINESS & INNOVATION**

**Essay / Assignment Title: Database System Design for a Stock  
Exchange Market**

**Programme title: MSc Data Analytics**

**Name: KEVADIYA SMIT KAMLESHBHAI**

**Year: 2023**

## CONTENTS

TABLE OF CONTENTS.....	2
INTRODUCTION.....	4
DESIGNED DATABASE AND EXPLANATION OF DATABASE, ENTITY AND RELATIONSHIP.....	7
EXPLANATION OF QUERIES.....	12
EXPLANATION OF CAP.....	26
CONCLUSIONS.....	27
BIBLIOGRAPHY.....	28



**Statement of compliance with academic ethics and the avoidance of plagiarism**

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people texts I clearly declare it through the good use of references following academic ethics.

(In the case that is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

SMIT KEVADIYA

.....

Date: 07/06/2023

## INTRODUCTION

The history of the stock market is one of ongoing opportunity and development. By purchasing and selling stocks, investors and traders engage in this dynamic market with the goal of profiting from changes in stock prices. The massive volumes of data related to stock market activities are managed behind the scenes by a reliable and effective database system. The construction of a database system that is especially suited for a stock exchange market will be discussed in this assignment.

### **Why Use a Database Management System (DBMS)?**

For the effective handling of stock market data, a database management system (DBMS) is required. Here are some explanations for why a DBMS is the best option:

**Data Organization:** A DBMS offers an organized method for classifying and archiving data. This enables the logical grouping of relevant information, such as stocks, prices, clients, and transactions, in the context of a stock market.

**Data Integrity:** DBMS enforces rules and restrictions to guarantee data integrity. In order to guarantee the quality and consistency of the data, it enables the specification of primary keys, foreign keys, and referential integrity. This is critical in the stock market, where reliable information is necessary to make wise investment choices.

**Data security:** Sensitive stock market data is protected by built-in security measures of a DBMS. It permits the use of access control systems to guarantee that only those with the proper authorization may see or alter the data.

**Concurrent Access:** In a stock market, multiple clients may be buying and selling stocks simultaneously. A DBMS supports concurrent access to the data, enabling multiple users to perform operations concurrently without interfering with each other. This ensures smooth and efficient market operations.

**Data Recovery:** DBMS provides mechanisms for data backup and recovery in case of system failures or data loss. This is crucial in the stock market, where the loss of transactional data can

have significant financial implications.

### **Reasoning for Entity and Attribute Design:**

The special demands and requirements of the market must be taken into account while building the entities and attributes of the stock market database system. Each entity represents a physical thing, and each object's attributes describe its traits or qualities. We guarantee that the database system appropriately captures and depicts the subtleties of the stock market by offering justifications for entity and attribute design.

In a database system for the stock market, for instance, the entities "Stocks" and "Clients" are essential. Clients stand in for the people or organizations who participate in the market, while stocks represent the many equities that are accessible for trading. We gather the data required for stock trading and account management by incorporating features such stock name, current price, client name, and account balance.

### **Understanding DBMS:**

A database management system (DBMS) is software that enables the development, repurposing, and administration of databases. To enable effective data storage, retrieval, and modification, it offers an interface between users or programs and the underlying database.

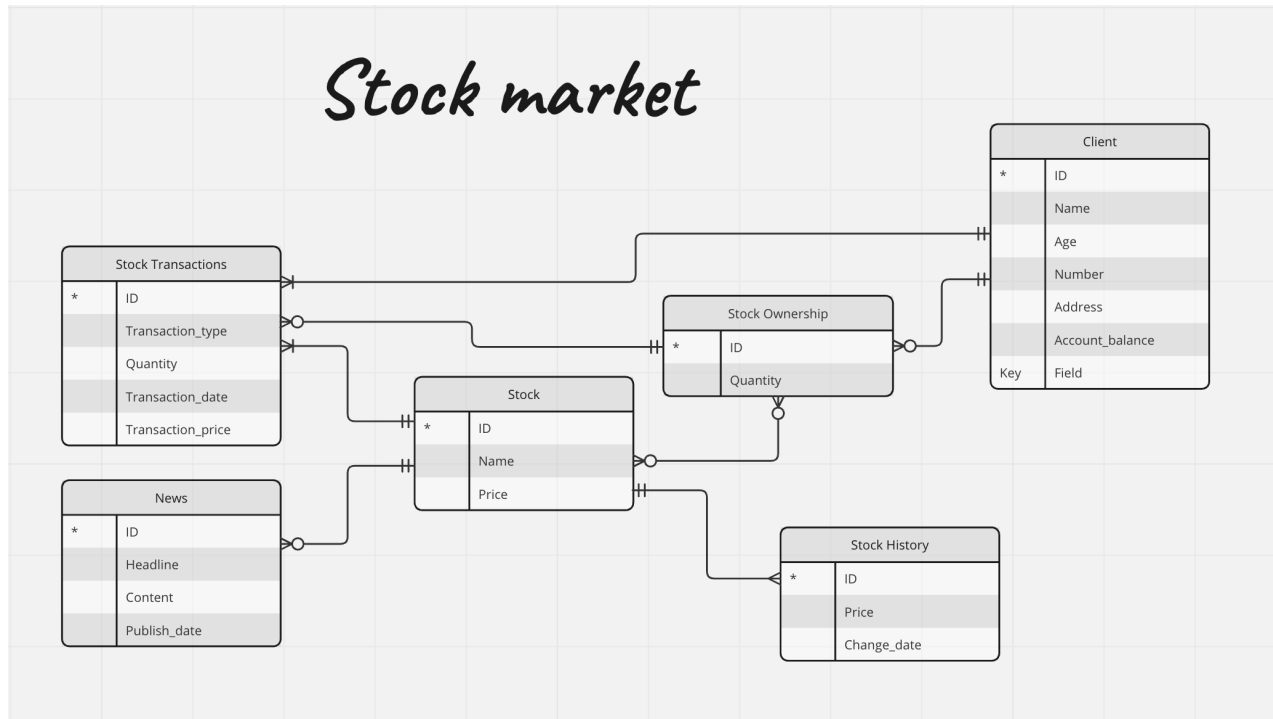
Data definition (making tables and establishing relationships), data manipulation (adding, removing, and updating entries), and data retrieval (querying the database) are all available as tools and functions through DBMSs. In addition, it takes care of other crucial facets of database administration, such as security, concurrency control, and data recovery.

A DBMS provides the effective storing and retrieval of stock information, price history, client information, and transaction data in the setting of a stock market database system. It offers the tools required to uphold data integrity, manage several tasks at once, and guarantee the safety and dependability of the stock market database.

The stock market database system uses a DBMS to transform into a potent instrument for organizing, analyzing, and reporting on stock market data, allowing market players to make decisions quickly and intelligently.

# DESIGNED DATABASE AND EXPLANATION OF DATABASE, ENTITY AND RELATIONSSHIP

## Entity Relationship Diagram:



## Schemas:

### *Schemas*

- Stock: Id(PK), Name, Price
- Stock History: Id(PK), stock\_id(FK), price, change\_date
- Client: Id(PK), bank\_account\_id, name, age, number, address
- Stock\_transactions: Id(PK), buyer\_client\_id(FK), seller\_client\_id(FK), stock\_id(FK), transaction\_type, quantity, transaction\_date, transaction\_price
- Stock Ownership: Id(PK), client\_id(FK), stock\_id(FK), quantity, stock\_transactions\_id(FK)
- News: Id(PK), stock\_id(FK), headline, content, publish\_date

## Entities and Relations in the Stock Market Database:

- Entity:Stock

Attributes:Id(PrimaryKey),Name,Price

The "Stock" entity represents individual stocks available for trading in the stock market. It captures the essential attributes of a stock, including its unique identifier (Id), name, and current price. This entity allows for the tracking and management of stocks within the-system.

- Entity:Stock-History

Attributes:Id(PrimaryKey),stock\_id(ForeignKey),price,change\_date

The "Stock History" entity stores the historical price changes for each stock. It is linked to the "Stock" entity through the foreign key stock\_id. The attributes include a unique identifier (Id), the foreign key referencing the corresponding stock (stock\_id), the price at a specific point in time, and the date of the price change. This entity enables the system to maintain a record of price history for analysis and reporting purposes.

- Entity:Client

Attributes: Id (Primary Key), bank\_account\_id (Foreign Key), name, age, number, address

The "Client" entity represents individuals or entities participating in the stock market. It captures relevant information about clients, including a unique identifier (Id), the foreign key referencing their associated bank account (bank\_account\_id), name, age, contact number, and address. This entity allows the system to manage client details and link them to their transactions and stock ownership-records.

- Entity:Stock\_transactions

Attributes: Id (Primary Key), buyer\_client\_id (Foreign Key), seller\_client\_id (Foreign Key), stock\_id (Foreign Key), transaction\_type, quantity, transaction\_date, transaction\_price

The "Stock\_transactions" entity records the details of each transaction made in the stock market. It includes a unique identifier (Id), foreign keys referencing the buyer and seller clients (buyer\_client\_id and seller\_client\_id), foreign key referencing the stock involved



in the transaction (stock\_id), transaction type (buy/sell), quantity of stocks traded, transaction date, and transaction price. This entity allows the system to track and analyze transaction-history.

- Entity:Stock-Ownership

Attributes: Id (Primary Key), client\_id (Foreign Key), stock\_id (Foreign Key), quantity, stock\_transactions\_id(ForeignKey)

The "Stock Ownership" entity represents the ownership relationship between clients and stocks. It captures the unique identifier (Id), foreign key referencing the client (client\_id), foreign key referencing the stock (stock\_id), quantity of stocks owned by the client, and foreign key referencing the associated stock transaction (stock\_transactions\_id). This entity allows the system to keep track of stock ownership by clients and link it to specific stock-transactions.

- Entity:News

Attributes: Id (Primary Key), stock\_id (Foreign Key), headline, content, publish\_date

The "News" entity captures news articles or updates related to specific stocks. It includes a unique identifier (Id), foreign key referencing the stock associated with the news (stock\_id), headline, content of the news article, and the publish date. This entity allows the system to store and link relevant news articles to specific stocks, providing valuable information to clients and facilitating analysis of market trends.

## **Relations:**

- One-to-Many: Stock to Stock History

Each stock can have multiple price changes recorded in the stock history table.

- One-to-Many: Client to Stock Ownership

Each client can have multiple stock ownership records.

- One-to-Many: Stock to Stock Ownership

Each stock can be owned by multiple clients.

- One-to-Many: Client to Stock Transactions  
Each client can perform multiple stock transactions.
- One-to-Many: Stock to Stock Transactions  
Each

### **One-to-One Relationship:**

One record in one table corresponds exactly to one record in another table in a one-to-one connection. A foreign key constraint is used to create this link, making the primary key of one table the foreign key of the other table. Each entry is guaranteed to be unique by the primary key, and the foreign key creates the link between the tables.

For illustration, have a look at the two tables "Employee" and "EmployeeAddress." There can only be one address per employee, and there can only be one employee per address. These tables would have a one-to-one connection with one another. The foreign key in the "EmployeeAddress" table, which connects the address record to its matching employee, would be the primary key of the "Employee" database (for example, EmployeeID).

### **One-to-Many Relationship:**

One record in one table is linked to numerous records in another table in a one-to-many connection. A foreign key in the "many" side table that refers to the main key in the "one" side table is used to construct this relationship. A single entry in the "one" side table can be linked to several entries in the "many" side table thanks to the foreign key.

For illustration, have a look at the tables "Department" and "Employee." Although a department may have several employees, each one is only connected to one department. These tables would have a one-to-many connection with one another. The "Employee" table's foreign key would be the primary key of the "Department" table (for example, DepartmentID), allowing numerous workers to be associated with a single department.

In summary, a one-to-one connection denotes a unique association between entries in two tables, but a one-to-many relationship denotes that one record in one table can be connected with

several records in another table. Both sorts of relationships are essential in database architecture and play an important role in properly organizing and organising data.

### **Reasons for Foreign Keys:**

Foreign keys play a crucial role in establishing relationships between tables in a relational database. Here are the reasons why foreign keys are essential in the stock market database:

- **Data Integrity:** Foreign keys enforce referential integrity, ensuring that the referenced data exists in the related table. For example, in the "Stock Transactions" schema, the foreign keys (Buyer\_Client\_Id, Seller\_Client\_Id, Stock\_Id) ensure that the clients and stocks referenced in a transaction actually exist.
- **Data Consistency:** Foreign keys help maintain data consistency by preventing orphaned or inconsistent data. By establishing relationships between tables, foreign keys ensure that related data remains synchronized and accurate. For example, the foreign key Stock\_Transactions\_Id in the "Stock Ownership" schema links the ownership records to the corresponding stock transactions, ensuring consistency between ownership and transaction data.
- **Query Performance:** Foreign keys optimize query performance by allowing efficient joins between tables. By using foreign keys, queries involving related data can be executed more efficiently, resulting in faster retrieval of information. For instance, joining the "Stock" and "Stock History".




# EXPLANATION OF QUERIES









## Client Table:

Name: <input type="text" value="client"/>										
Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
age	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
number	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
address	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
account_bala...	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

	id	name	age	number	address	account_balan...
▶	10	smit	21	9907392	Berlin	9999
▶	11	vivek	23	9898379	Surat	4500
▶	12	Raj	35	98078678	Berlin	9000
▶	13	Hiren	27	9876656	Frankfurt	3800
▶	14	Ashkay	25	97665456	Botad	7099
▶	15	Dhaval	28	78889909	Berlin	45000
▶	16	bhagavat	20	98274859	Botad	3400
▶	17	Ashitosh	18	90989756	Frankfurt	1280
	NULL	NULL	NULL	NULL	NULL	NULL

### Stock Table:

Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G
 id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 price	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>





	id	name	price	
	1	SG infotech	2200	
	2	Infosys	1100	
	3	Jio	2000	
	4	Tata	3400	
	5	Birla	1500	
	6	Rentech	4500	
	7	RJ info	800	
	NULL	NULL	NULL	

## News Table:

Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
headline	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
content	VARCHAR(450)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
publish_date	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
stock_id	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	












	id	headline	content	publish...	stock_id
▶	1	SG infotech	this stock is goinf to up and they have good bus...	2023-12...	1
▶	2	Down	Do not invest in the stock jio.	2023-12...	3
▶	3	buy this stock	buy tata stock bocuse this stock buy the jio stock.	2023-12...	4
▶	4	medium stock	if you want to invest safe, we recoumend invest...	2023-12...	2
▶	5	About Birla	Berlin is going to down becouse of the some fak...	2023-12...	5
▶	6	SG infotech is reach on hiegst lavel	this stock is incredible of other stocks	2023-12...	1
	NULL	NULL	NULL	NULL	NULL

## Stock History table:

Column	Datatype		PK	NN	UQ	B...	UN	ZF	AI	G
 id	INT	↕	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 price	INT	↕	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 change_date	DATETIME	↕	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 stock_id	INT	↕	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<click to edit>		↕	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	id	price	change_date	stock_id	
▶	1	800	2023-11-01 12:34:45	1	
■	2	500	2023-04-01 12:34:45	2	
■	3	780	2023-05-01 12:34:45	3	
■	4	700	2023-09-01 12:34:45	4	
■	5	560	2022-12-01 12:34:45	5	
■	6	234	2023-12-01 12:34:45	6	
■	7	120	2023-01-01 12:34:45	7	
■	NULL	NULL	NULL	NULL	

### Stock Ownership Table:

Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G
 id	INT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 quantity	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 client_id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 stock_id	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 stock_transa...	INT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<click to edit>			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

	id	quantity	client_id	stock_id	stock_transactions...	
	1	6	14	2	50	
	66	2	11	1	51	
	67	17	16	6	52	
	NULL	NULL	NULL	NULL	NULL	
						
						



## Stock Transactions Table:

Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
transaction_t...	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
quantity	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
transaction_...	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
transaction_...	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
seller_client_id	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
stock_id	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
buyer_client_id	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

	id	transaction_ty...	quantity	transaction_date	transaction_pri...	seller_client...	stock_id	buyer_client...
▶	12	done	2	2023-11-02 00:00:00	4000	10	1	11
▶	48	fail	10	2023-11-02 00:00:00	5000	17	2	12
▶	49	fail	10	2023-11-02 00:00:00	5000	17	2	12
▶	50	done	5	2023-11-02 00:00:00	5000	17	2	12
▶	51	done	2	2023-11-02 00:00:00	2390	14	5	11
▶	52	done	8	2023-11-02 00:00:00	2100	10	1	11
▶	53	fail	1	2023-11-02 00:00:00	1100	12	4	11
▶	54	done	16	2023-11-02 00:00:00	9988	12	6	17
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Below I provide queries, output and their explanation.

- Query 1:

```

SELECT s.Name, sh.price, sh.change_date
FROM Stock s
JOIN Stock_History sh ON s.Id = sh.stock_id
WHERE sh.change_date = (
    SELECT MAX(change_date)
    FROM Stock_History
    WHERE stock_id = s.Id
)

```

Output 1:

'SG infotech','800','2023-11-01 12:34:45'

'Infosys','500','2023-04-01 12:34:45'

'Jio','780','2023-05-01 12:34:45'

'Tata','700','2023-09-01 12:34:45'

'Birla','560','2022-12-01 12:34:45'

'Rentech','234','2023-12-01 12:34:45'

'RJ info','120','2023-01-01 12:34:45'

#### **Explanation 1:**

This query retrieves the latest price change (price and change date) for each stock. It joins the "Stock" table with the "Stock\_History" table using the stock ID as the foreign key. The subquery finds the maximum change date for each stock. By comparing the change date with the maximum date, the query returns only the latest price change for each stock.

- **Query 2:**

```
SELECT c.Name, SUM(so.quantity) AS Total_Quantity
FROM Client c
JOIN Stock_Ownership so ON c.Id = so.client_id
GROUP BY c.Name
```

Output 2:

'Ashkay','6'

'vivek','2'

'bhagavat','17'

### **Explanation 2:**

This query calculates the total quantity of stocks owned by each client. It joins the "Client" table with the "Stock\_Ownership" table using the client ID as the foreign key. The SUM() function aggregates the quantity column to calculate the total quantity for each client. The results are then grouped by the client's name using the GROUP BY clause.

- **Query 3:**

```
SELECT c.Name, st.transaction_date, st.quantity, st.transaction_price
      FROM Client c
      JOIN Stock_Transactions st ON c.Id = st.buyer_client_id
      WHERE st.stock_id = (SELECT Id FROM Stock WHERE Name = 'SG
                           infotech')
```

Output 3:

```
'vivek','2023-11-02 00:00:00','2','4000'
```

```
'vivek','2023-11-02 00:00:00','8','2100'
```

Explanation 3:

This query retrieves the clients who have bought a specific stock. It joins the "Client" table with the "Stock\_Transactions" table using the buyer client ID as the foreign key. The WHERE clause filters the results based on the stock ID, which is obtained using a subquery to find the stock's ID based on its name.

- **Query 4:**

```
SELECT s.Name, st.transaction_type, st.quantity, st.transaction_date
      FROM Stock s
      JOIN Stock_Transactions st ON s.Id = st.stock_id
      WHERE st.transaction_date = ( SELECT MAX(transaction_date) FROM
                                   Stock_Transactions WHERE stock_id = s.Id)
```

Output 4:

```
'SG infotech','done','2','2023-11-02 00:00:00'
'Infosys','fail','10','2023-11-02 00:00:00'
'Infosys','fail','10','2023-11-02 00:00:00'
'Infosys','done','5','2023-11-02 00:00:00'
'Birla','done','2','2023-11-02 00:00:00'
'SG infotech','done','8','2023-11-02 00:00:00'
'Tata','fail','1','2023-11-02 00:00:00'
'Rentech','done','16','2023-11-02 00:00:00'
```

Explanation 4:

This query retrieves the stocks along with their latest transaction details. It joins the "Stock" table with the "Stock\_Transactions" table using the stock ID as the foreign key. The subquery finds the maximum transaction date for each stock. By comparing the transaction date with the maximum date, the query returns only the latest transaction details for each stock.

- **Query 5:**

```
SELECT n.headline, n.content, n.publish_date
      FROM News n
     JOIN Stock s ON n.stock_id = s.Id
    WHERE s.Name = 'SG infotech'
```

Output 5:

```
'SG infotech','this stock is goinf to up and they have good bussiness.','2023-12-01
12:34:45'
'SG infotech is reach on hiegst lavel','this stock is incredible of other stocks','2023-12-01
12:34:45'
```

Explanation 5:

This query retrieves the news articles associated with a specific stock. It joins the "News" table with the "Stock" table using the stock ID as the foreign key. The WHERE clause filters the results based on the stock name, retrieving only the news articles related to the specified stock.

- **Query 6:**

-- Step 1: Insert a new record in the Stock\_History table

```
INSERT INTO Stock_History (stock_id, price, change_date)
VALUES (1, new_price, CURRENT_TIMESTAMP);
```

-- Step 2: Update the Stock table with the new price

```
UPDATE Stock
SET Price = new_price
WHERE Id = 1;
```

-- Step 3: Insert a new record in the Stock\_Transactions table

```
INSERT INTO Stock_Transactions (buyer_client_id, seller_client_id, stock_id,
transaction_type, quantity, transaction_date, transaction_price)
VALUES (11, 13, 1, 'Buy', quantity_value, CURRENT_TIMESTAMP, (new_price *
quantity_value));
```

-- Step 4: Update or insert a record in the Stock\_Ownership table

```
IF EXISTS (
    SELECT 1
    FROM Stock_Ownership
    WHERE client_id = 11 AND stock_id = 1
) THEN
```

```

-- If the buyer already owns some quantity of the stock, update the quantity

UPDATE Stock_Ownership

SET quantity = quantity + quantity_value,

    stock_transactions_id = LAST_INSERT_ID()

WHERE client_id = 11 AND stock_id = 1;

ELSE

    -- If the buyer doesn't own any quantity of the stock, insert a new record

    INSERT INTO Stock_Ownership (client_id, stock_id, quantity, stock_transactions_id)

    VALUES (11, 1, quantity_value, LAST_INSERT_ID());

END IF;

```

#### Explanation 6:

In this example, I have used the stock ID as 1, buyer client ID as 11, and seller client ID as 13. You can replace these values with the appropriate IDs for your scenario.

The query performs the following steps:

1. Inserts a new record in the Stock\_History table to store the price change.
2. Updates the Stock table with the new price.
3. Inserts a new record in the Stock\_Transactions table to log the transaction details, including the buyer and seller IDs, stock ID, transaction type ('Buy'), quantity, transaction date, and transaction price calculated based on the quantity and new price. Updates or inserts a record in the Stock\_Ownership table based on whether the buyer already owns some quantity of the stock. If the buyer exists in the Stock\_Ownership table, the quantity is updated by adding the purchased quantity, and the stock\_transactions\_id is set to the ID of the last inserted record. If the buyer doesn't exist in the Stock\_Ownership table, a new record is inserted with the buyer's ID, stock ID, purchased quantity, and the stock\_transactions\_id.
4. By executing this query, the stock's price will be updated, the transaction details will be logged, and the stock ownership records will be updated accordingly based on the buyer's ID and the stock ID provided.

#### Query:7

-- Step 1: Retrieve the current price of stock 4

```
SELECT Price FROM Stock WHERE Id = 4;
```

-- Step 2: Update the price of stock 4 in the Stock table

```
UPDATE Stock SET Price = 2800 WHERE Id = 4;
```

-- Step 3: Insert the new price in the Stock\_History table

```
INSERT INTO Stock_History (stock_id, price, change_date)
VALUES (4,2800 , CURRENT_TIMESTAMP);
```

### Output: 7

#### Stock Table:

'1','SG infotech','2200'	'1','SG infotech','2200'
'2','Infosys','1100'	'2','Infosys','1100'
'3','Jio','2000'	'3','Jio','2000'
'4','Tata','3400' <----->	'4','Tata','2800'
'5','Birla','1500'	'5','Birla','1500'
'6','Rentech','4500'	'6','Rentech','4500'
'7','RJ info','800'	'7','RJ info','800'

#### Stock Transactions Table:

'1','800','2023-11-01 12:34:45','1'	'1','800','2023-11-01 12:34:45','1'
'2','500','2023-04-01 12:34:45','2'	'2','500','2023-04-01 12:34:45','2'
'3','780','2023-05-01 12:34:45','3'	'3','780','2023-05-01 12:34:45','3'
'4','700','2023-09-01 12:34:45','4'	'4','700','2023-09-01 12:34:45','4'
'5','560','2022-12-01 12:34:45','5'	'5','560','2022-12-01 12:34:45','5'
'6','234','2023-12-01 12:34:45','6'	'6','234','2023-12-01 12:34:45','6'
'7','120','2023-01-01 12:34:45','7'	'7','120','2023-01-01 12:34:45','7'
<-----New Recoed----->	'8','2800','2023-06-07 01:53:50','4'

### Explanation: 7

In this query, you need to replace <new\_price> with the desired new price for stock 4.

The query performs the following steps:

The first step is to retrieve the current price of stock 4 from the Stock table. This allows you to verify the existing price before making any changes.

Next, the query updates the Price column of stock 4 in the Stock table with the new price.

The WHERE clause ensures that only stock 4 is updated.

Finally, the query inserts a new record into the Stock\_History table, capturing the change in price for stock 4. It includes the stock\_id (4), the new price, and the current timestamp as the change\_date.

By executing this query, you can change the price of stock 4 and store the previous price in the Stock\_History table with the corresponding change date.

### Query: 8

```
SELECT c.*, n.headline, n.content, n.publish_date
FROM Client c
JOIN Stock_Ownership so ON c.Id = so.client_id
JOIN News n ON so.stock_id = n.stock_id;
```

Output: 8

id	name	age	number	address	account_balan...	headline	content	publish_date
14	Ashkay	25	97665456	Botad	7099	medium stock	if you want to invest safe. we reco...	2023-12-01 12:34:45
11	vivek	23	9898379	Surat	4500	SG infotech	this stock is goinf to up and they h...	2023-12-01 12:34:45
11	vivek	23	9898379	Surat	4500	SG infotech is reach on hiegst level	this stock is incredible of other stocks	2023-12-01 12:34:45

Explanation: 8

This query joins the Client, Stock\_Ownership, and News tables using the stock\_id foreign key. It selects all columns from the Client table (c.\*) along with the headline, content, and publish\_date columns from the News table (n.headline, n.content, n.publish\_date).

The join conditions are as follows:



- The Client table is joined with the Stock\_Ownership table using the client\_id foreign key from the Client table and the client\_id foreign key from the Stock\_Ownership table.
- The Stock\_Ownership table is joined with the News table using the stock\_id foreign key from the Stock\_Ownership table and the stock\_id foreign key from the News table.

By executing this query, you will obtain a result set that includes information about the clients and the corresponding news headlines, content, and publish dates for the stocks they own. This join allows you to retrieve data from both the Client and News tables based on the relationship defined through the Stock\_Ownership table.

## EXPLANATION OF CAP

The CAP theorem, also known as Brewer's theorem, says that it is impossible for a distributed database system to provide three desired features at the same time: consistency, availability, and partition tolerance.

The need that all nodes in a distributed system have the same view of data at any one time is referred to as consistency. Even in the face of failures, availability ensures that the system remains functioning and responsive to user demands. Partition tolerance is the ability of a system to continue operating and preserving consistency despite network partitions.

The CAP theorem states that a distributed database system can only prioritize two of the three qualities. In other words, when confronted with a network split, the system must choose between preserving consistency (guaranteeing that all nodes have the same data) and ensuring availability (response to user requests). Because of the inherent difficulties of distributed systems, this trade-off is required.

Several things impact database system efficiency. Transactions are critical to preserving data integrity and consistency. Transactions' ACID (Atomicity, Consistency, Isolation, and Durability) qualities ensure that database operations are completed reliably and predictably.

Another key component is consistency, which guarantees that data is legitimate and correct. A consistent database ensures that all data modifications follow the same set of rules and restrictions.

Scalability is critical for dealing with rising workloads and datasets. It refers to the database system's capacity to manage increased traffic, greater volumes of data, and more concurrent user requests without sacrificing speed.

Indexing methods, query optimization, data caching, hardware architecture, and database management system (DBMS) setups are all elements that influence efficiency.

Based on the unique needs and limits of their application, database designers and administrators must carefully analyze these characteristics and make suitable trade-offs to achieve the required balance of consistency, availability, scalability, and overall system efficiency.

## CONCLUDING REMARKS

In conclusion, when designing a database system for a stock exchange market, it is important to consider entities such as stocks, clients, transactions, and news. A relational DBMS is suitable due to its structured data and ability to handle complex relationships. Factors affecting efficiency include transactions, consistency, and scalability. Transactions ensure data integrity, consistency guarantees accurate information, and scalability allows for handling increasing workloads. It is essential to carefully consider trade-offs based on specific requirements. Overall, a well-designed database system is crucial for a stock exchange market to provide reliable and accurate information to clients and stakeholders.

## BIBLIOGRAPHY

1. Date, C.J. (2003). An Introduction to Database Systems. Addison-Wesley.
2. Elmasri, R., & Navathe, S.B. (2019). Fundamentals of Database Systems. Pearson.
3. Silberschatz, A., Korth, H.F., & Sudarshan, S. (2019). Database System Concepts.
4. Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems. McGraw-Hill.
5. Garcia-Molina, H., Ullman, J.D., & Widom, J. (2008). Database Systems: The Complete Book. Pearson.
6. Oracle Corporation. (n.d.). Oracle Database Concepts. Retrieved from <https://docs.oracle.com/database/121/CNCPT/toc.htm>
7. Microsoft. (n.d.). SQL Server Documentation. Retrieved from <https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-ver15>

## **APPENDIX (if necessary)**

