Database Project

Mehmet Turhan

Domain Description

Title: Stock Market Investing and Advising Database

We will create a database for a brokerage firm that manages investment portfolios for its clients. The firm has a number of financial advisors, who each have a set of clients they work with. Clients can hold multiple accounts with the firm, each of which can contain multiple investments in various stocks. The firm also tracks various financial metrics for each investment, such as its current price and its historical performance.

Entities and Relationships:

Clients: This entity will contain information about individual clients, including their name, contact information, and other relevant details. Each client will be assigned a unique identifier, client_id, to ensure that their information can be easily tracked and accessed.

Accounts: This entity will contain information about individual accounts held by clients, including the account_id, client_id (foreign key), balance, and performance. Each account will be assigned a unique identifier, account_id, to ensure that its information can be easily tracked and accessed. It also has an advisor_id (foreign key) assigned to see which advisor is helping to make decisions on the investments for that account.

Investments: This entity will contain information about individual investments made by clients, including the investment_id, stock_id (foreign key), account_id (foreign key), quantity, purchase_price, and purchase_date. Each investment will be assigned a unique identifier, investment_id, to ensure that its information can be easily tracked and accessed.

Advisors: This entity will contain information about individual advisors, including their name, contact information, and other relevant details. Each advisor will be assigned a unique identifier, advisor_id, to ensure that their information can be easily tracked and accessed. It will also contain an attribute called performance, which tracks the overall performance of the advisor's investments.

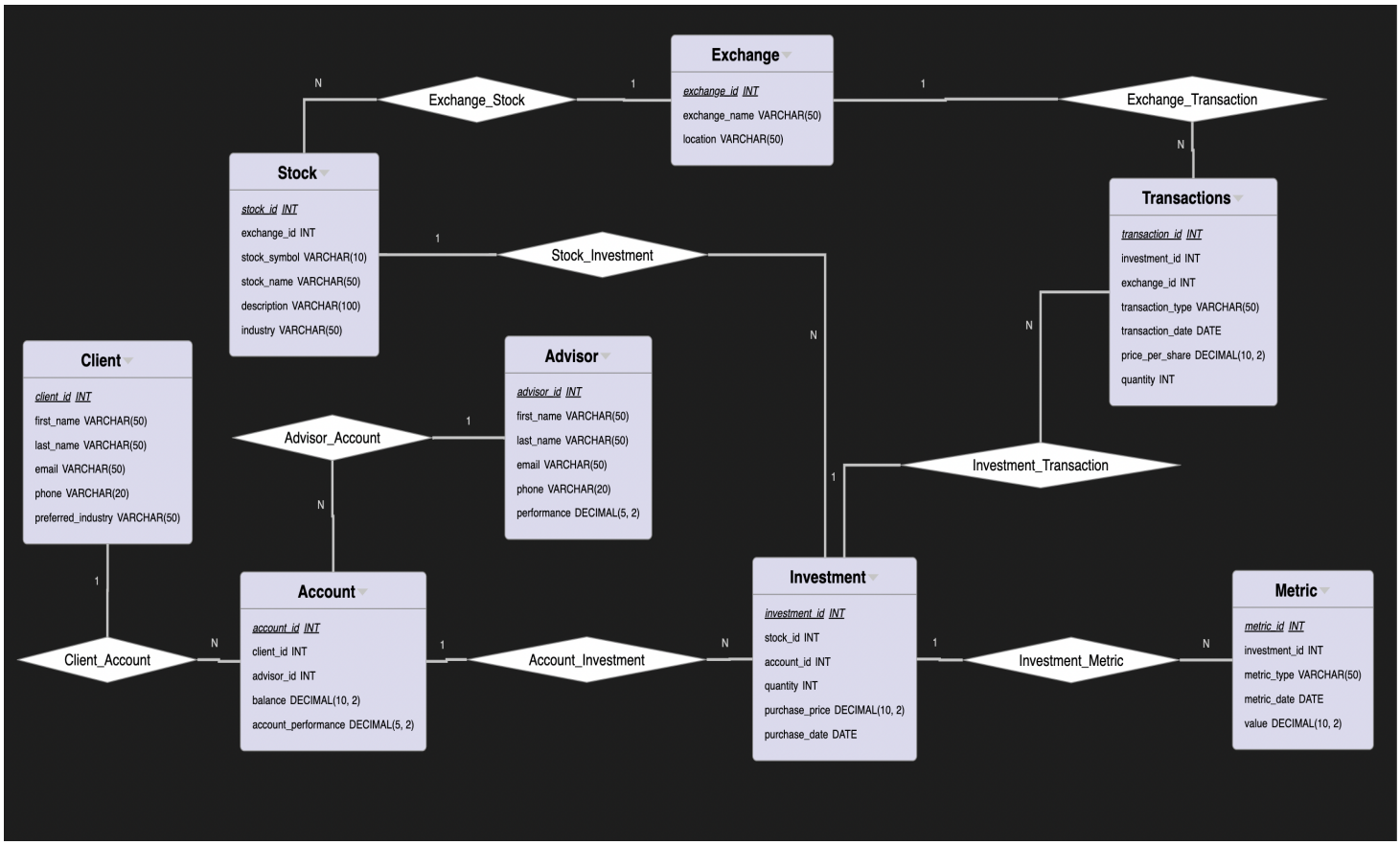
Transactions: This entity will contain information about individual transactions made by clients, including the transaction_id, investment_id (foreign key), exchange_id (foreign key), transaction_type, transaction_date, price_per_share, and quantity. Each transaction will be assigned a unique identifier, transaction_id, to ensure that its information can be easily tracked and accessed.

Metrics: This entity will contain information about individual metrics for each investment, including the metric_id, investment_id (foreign key), metric_type, metric_date, and value. Each metric will be assigned a unique identifier, metric_id, to ensure that its information can be easily tracked and accessed.

Exchanges: This entity will contain information about individual exchanges, including the exchange_id and exchange_name. Each exchange will be assigned a unique identifier, exchange_id, to ensure that its information can be easily tracked and accessed. It will also contain an attribute called location, which shows where the exchange is located.

Stocks: This entity will contain information about individual stocks, including the stock_id, exchange_id (foreign key), stock_symbol, stock_name, description, and industry. Each stock will be assigned a unique identifier, stock_id, to ensure that its information can be easily tracked and accessed. It will also contain an attribute called industry, which shows which industry the stock is in.

Entity-Relationship Diagram



Relational Schema

Client (client_id, first_name, last_name, email, phone, preferred_industry(FK))

Account (account_id, client_id (FK), advisor_id (FK), balance, account_performance)

Investment (investment_id, stock_id (FK), account_id (FK), quantity, purchase_price, purchase_date)

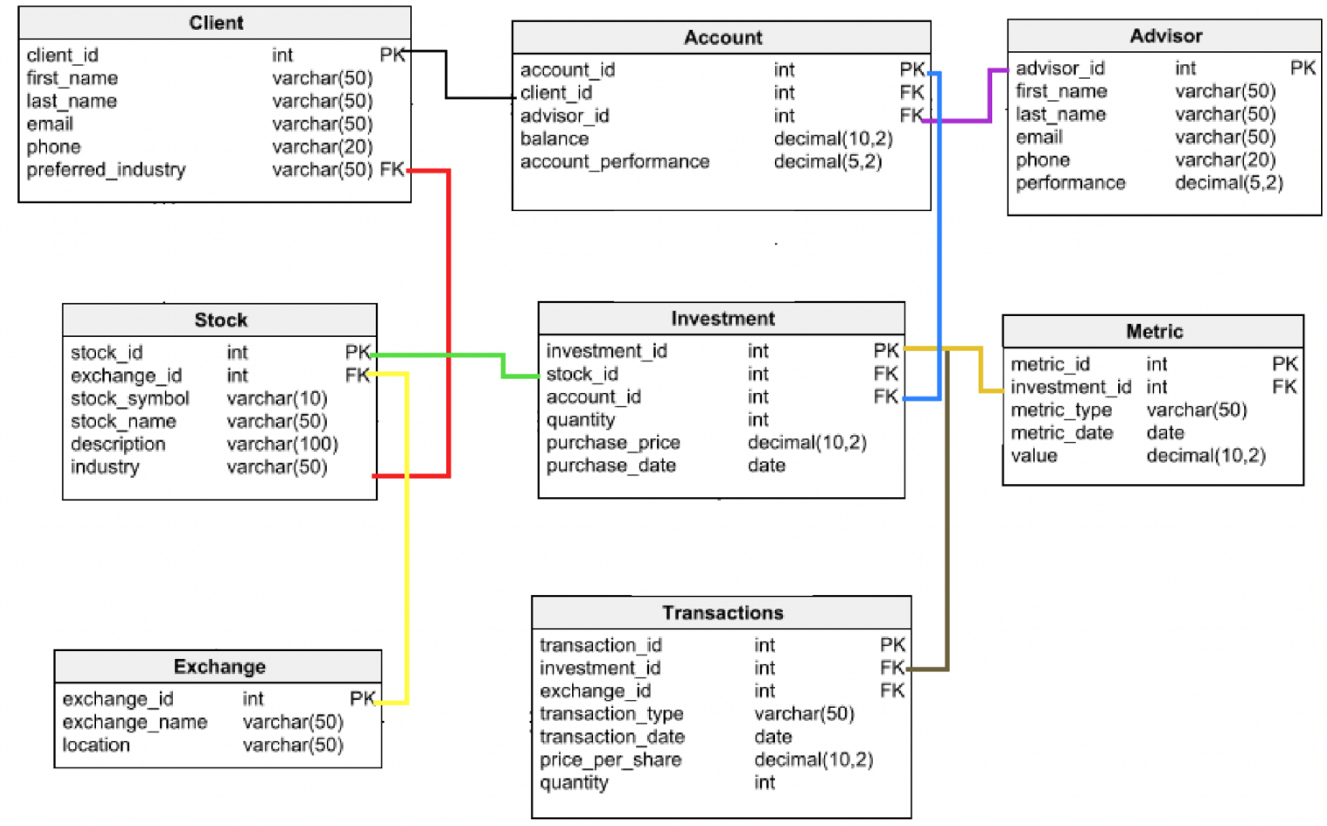
Advisor (advisor_id, first_name, last_name, email, phone, performance)

Transactions (transaction_id, investment_id (FK), exchange_id (FK), transaction_type, transaction_date, price_per_share, quantity)

Metric (metric_id, investment_id (FK), metric_type, metric_date, value)

Exchange (exchange_id, exchange_name, location)

Stock (stock_id, exchange_id (FK), stock_symbol, stock_name, description, industry)



Boyce–Codd Normal Form Decomposition

First we identify all functional dependencies in the schema.
Client_id → {first_name, last_name, email, phone, preferred_industry}
Account_id → {client_id, advisor_id, balance, account_performance}
Investment_id → {stock_id, account_id, quantity, purchase_price, purchase_date}
Advisor_id → {first_name, last_name, email, phone, performance}
Investment_id {transaction_id, exchange_id, transaction_type, transaction_date, price_per_share, quantity}
Investment_id → {metric_id, metric_type, metric_date, value}
Exchange_id → {exchange_name, location}
Stock_id → {exchange_id, stock_symbol, stock_name, description, industry}

Then we check whether any non-prime attribute is functionally dependent on a proper subset of a candidate key.
The candidate keys:
{client_id}
{account_id}
{investment_id}
{advisor_id}
{transaction_id}
{metric_id}
{exchange_id}
{stock_id}

The non-prime attributes:
{first_name, last_name, email, phone, preferred_industry} (from the Client relation)
{balance, account_performance} (from the Account relation)
{quantity, purchase_price, purchase_date} (from the Investment relation)
{performance} (from the Advisor relation)
{transaction_type, transaction_date, price_per_share, quantity} (from the Transactions relation)
{metric_type, metric_date, value} (from the Metric relation)
{exchange_name, location} (from the Exchange relation)
{stock_symbol, stock_name, description, industry} (from the Stock relation)
We can see that all non-prime attributes are fully dependent on the candidate keys, and there are no partial dependencies. Therefore, the schema is in 2NF.

Then we check whether any non-prime attribute is transitively dependent on a candidate key.
There are no non-prime attributes that are transitively dependent on a candidate key. Therefore, the schema is in 3NF.
Lastly we check whether any non-prime attribute is functionally dependent on another non-prime attribute.

There are no non-prime attributes that are functionally dependent on another non-prime attribute. Therefore, the schema is in BCNF.

Transaction and Query Executions

```sql
--1: Find the total balance of all accounts grouped by preferred industry of clients.
SELECT c.preferred_industry, SUM(a.balance) as total_balance
FROM Client c
INNER JOIN Account a ON c.client_id = a.client_id
GROUP BY c.preferred_industry;
```

preferred_industry	total_balance
Energy	15000
Finance	34000
Healthcare	30500
Real Estate	68000
Retail	12000
Technology	40000

```sql
--2: Find the advisors whose performance is greater than the average performance of all
advisors and their associated clients.
SELECT DISTINCT ad.first_name, ad.last_name, ad.performance
FROM Advisor ad
WHERE ad.performance > (
    SELECT AVG(performance)
    FROM Advisor
) ORDER BY ad.performance DESC;
```

first_name	last_name	performance
Cathie	Wood	9.5
David	Swensen	9.4
Larry	Fink	9.3
Paul	Tudor Jones	9.2
Ray	Dalio	9.1

```sql
--3: Give the first and last name of clients and their advisors who invested in the Real
Estate industry and have an account performance greater than 4.0.
SELECT c.first_name, c.last_name, a.first_name, a.last_name
FROM Client c
INNER JOIN Account ac ON c.client_id = ac.client_id
INNER JOIN Advisor a ON ac.advisor_id = a.advisor_id
WHERE c.preferred_industry = 'Real Estate'
AND ac.account_performance > 4.0;
```

first_name	last_name	first_name	last_name
Not	Sure	Paul	Tudor Jones

```sql
--4: Give the first and last name of advisors who have no clients.
SELECT a.first_name, a.last_name
FROM Advisor a
LEFT OUTER JOIN Account ac ON a.advisor_id = ac.advisor_id
WHERE ac.account_id IS NULL;

NO RESULT

--5: Find the first and last name of the clients and the advisors who have performance
greater than 4.0
SELECT c.first_name AS client_first_name, c.last_name AS client_last_name,
       adv.first_name AS advisor_first_name, adv.last_name AS advisor_last_name
FROM Client c
JOIN Account a ON c.client_id = a.client_id
JOIN Advisor adv ON a.advisor_id = adv.advisor_id
WHERE a.account_performance > 4.0;
```

client_first_name	client_last_name	advisor_first_name	advisor_last_name
Mehmet	Turhan	Jamie	Dimon
Lincoln	Chapata	Larry	Fink
Nick	Price	Abigail	Johnson
Rawleigh	Pollock	David	Swensen
Abdulrahman	Nassar	Warren	Buffett
Not	Sure	Paul	Tudor Jones

```sql
--6: Find the advisor's first and last name and their performance who have clients with
account balance less than 15000.
SELECT DISTINCT adv.first_name, adv.last_name, adv.performance
FROM Advisor adv
JOIN Account a ON adv.advisor_id = a.advisor_id
WHERE a.balance < 15000;
```

first_name	last_name	performance
Abigail	Johnson	9
Howard	Marks	8.8
Ken	Griffin	8.9

```sql
--7: Find the purchase_price of all the investments made in NASDAQ and purchase_price of
more than 3000$.
SELECT s.stock_name, i.purchase_price
FROM Stock s
INNER JOIN Investment i ON s.stock_id = i.stock_id
WHERE s.exchange_id = (SELECT exchange_id FROM Exchange WHERE exchange_name = 'NASDAQ')
AND i.purchase_price > 3000;
```

stock_name	purchase_price
Tesla, Inc.	6240
INDUS Realty Trust, Inc.	8945
Biogen Inc.	9230

```sql
--8: Find all investments and their associated metrics.
SELECT Investment.investment_id, Investment.purchase_price, Metric.metric_type, Metric.value
FROM Investment
LEFT OUTER JOIN Metric ON Investment.investment_id = Metric.investment_id;
```

investment_id	purchase_price	metric_type	value
1	12550	Dividend Yield	0.005
1	12550	PE ratio	25.7

investment_id	purchase_price	metric_type	value
2	17640	Dividend Yield	0.008
2	17640	PE ratio	35.6
3	14560	Dividend Yield	0.015
3	14560	PE ratio	20.9
4	6240	Dividend Yield	0.025
4	6240	PE ratio	15.3
5	8945	Dividend Yield	0.03
5	8945	PE ratio	12.8
6	9230	Dividend Yield	0.02
6	9230	PE ratio	18.4
7	7812.5	Dividend Yield	0.012
7	7812.5	PE ratio	21.6
8	5820	Dividend Yield	0.007
8	5820	PE ratio	30.2
9	6502.5	Dividend Yield	0.02
9	6502.5	PE ratio	16.5
10	5280	Dividend Yield	0.008
10	5280	PE ratio	28.6