

Project Step 3

Siyuan Qi
Yaodan Zhang
Jiajun Lin

May 21, 2024

1 Descriptions

Conceptual Database Design

This is a conceptual database design for a trading system, detailing the relationships between stocks, ETFs, indices, users, and trading sessions. In this system, users engage in trading sessions dealing with a variety of financial instruments. A user's portfolio is managed over time, with asset allocation adjusted as needed. The Sharpe Ratio is used to assess the risk-adjusted return of the investments. It includes the following specifications:

- Stocks are identified by a stock ID (sId) and each stock record includes its ticker symbol, open price, high price, low price, close price, and the date of these prices.
- Futures are identified by a future ID (fId) and each future record includes its ticker symbol, open price, high price, low price, close price, and the date.
- Indices are identified by an index ID (iId) and each index record includes its ticker symbol, open price, high price, low price, close price, and the date.
- Stocks, ETFs, and indices are interconnected through the “contain” relationships indicating their composition.
- User trading sessions are identified by a session ID and include the start date, end date, the amount of trade, the position (long/short), and an underlying ID that connects to stocks, futures, or indices.
- Users are identified by a user ID (uId) and records include the first name, last name, email, phone, and trading preferences.
- The trading result is recorded with risk and return metrics to calculate the Sharpe Ratio.
- Each user manages a portfolio, identified by the user ID, which records the start time, end time, and asset allocation.
- User trading sessions are managed by users and connected to the assets being traded.

2 ER Diagram

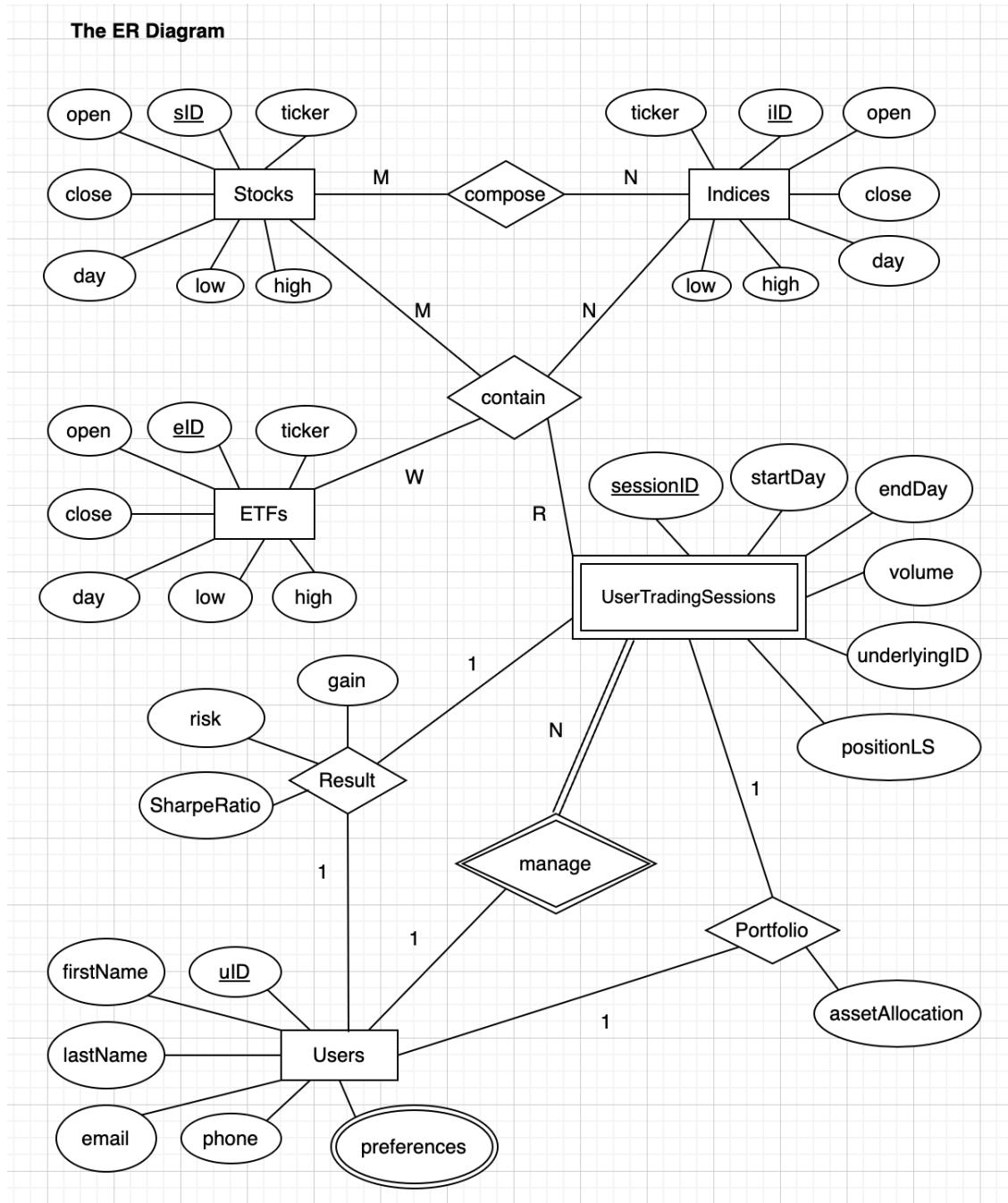


Figure 1: ER Diagram

3 Relational Schema

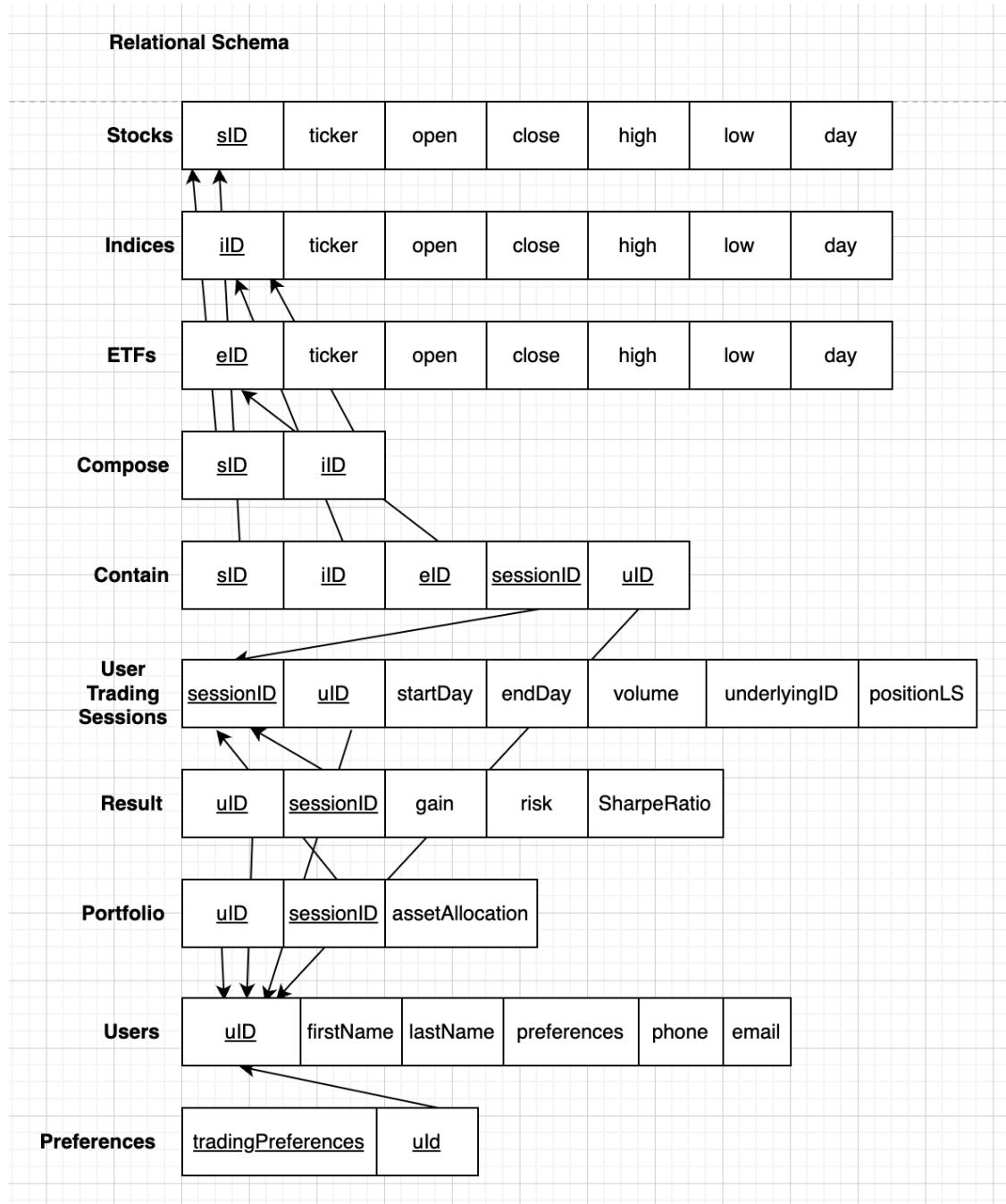


Figure 2: Relational Schemas

4 Prepare Datasets

The following details the procedure for retrieving real-world financial data to populate the trading system's database. The data acquisition will be performed using the Python 'yfinance' module, which allows us to fetch historical market data from Yahoo Finance. Additionally, for the purpose of testing and demonstration, we will craft mock data for non-financial entities, simulating realistic trading scenarios. This allows us to assume the role of traders and create a dynamic dataset that reflects various user interactions and trading behaviors within the system.

Retrieving Financial Data

The following Python code utilizes the 'yfinance' library to download historical data for stocks. The 'get_stock_data' function retrieves data points such as 'Open', 'Close', 'High', 'Low', and 'Date' for a specified stock ticker. These data points are in line with the attributes defined in the Relational Schema for the 'Stock' entity. Similar operations can be performed for other entities.

```
1 import yfinance as yf
2
3 def get_stock_data(ticker):
4
5     # Downloads the stock's historical data using yfinance.
6     data = yf.download(ticker, period="max")
7
8     # Filters the dataframe to include only the required columns.
9     data = data[["Open", "Close", "High", "Low"]]
10
11    # Resets the index to convert the "Date" index into a column.
12    data.reset_index(inplace=True)
13
14    # Renames the columns to adhere to the relational schemas.
15    data.columns = ["date", "open", "close", "high", "low"]
16    return data
17
18 # Demonstrates the function usage with Apple's stock ticker 'AAPL'.
19 ticker = "AOS"
20 stock_data = get_stock_data(ticker)
21 print(stock_data.head())
```

Listing 1: An Example to Retrieve Data

Integration with the Database

The output of the 'get_stock_data' function is a pandas DataFrame, which can be easily exported to various formats, such as CSV, to integrate with a database.

Sourcing Additional Data

For the non-financial entities, we will generate these data assuming the role of a mock user. Essentially, we can simulate the trading activity as if we were the trader. This approach allows us to create realistic yet fictitious data sets that can be used to test and demonstrate the functionality of the trading system database.

5 How the App Works

Let us start with a description of the app.

6 Queries and Results

6.1 Queries 1