TradingView

Product Dissection for TradingView

Company Overview:

TradingView is a leading financial charting and social networking platform. Founded in 2011 by Konstantin Ivanov, Denis Globa, and Stan Bokov, the company is headquartered in New York. TradingView provides traders and investors with powerful charting tools, real-time data, and a vibrant community for sharing trading ideas. The company's mission is to make financial markets more accessible and understandable for everyone, regardless of their experience level

Problem 1: Data Volume and Velocity

Real-World Challenge:

Financial markets generate an immense volume of data every second, including price quotes, trade volumes. The rapid pace of trades and updates necessitates a robust system capable of handling high-frequency data without latency. Combining data from multiple sources, such as different exchanges and market data providers, into a cohesive and real-time feed can be challenging.

TradingView's Solution:

Broker Account Integration: TradingView allows users to connect their broker accounts to the platform, enabling direct access to real-time market data provided by their brokers. This ensures that users receive accurate and timely information tailored to their trading activities.

Advanced Infrastructure: TradingView employs a highly scalable and resilient infrastructure to manage and process large volumes of data efficiently. This includes using distributed systems and cloud technologies to handle the load and ensure minimal latency.

Problem 2: Operational Efficiency

Real-World Challenge:

Maintaining efficient operations in the context of high-frequency trading and rapid market changes is critical for a platform like TradingView.

TradingView's Solution:

Scalable Infrastructure: TradingView uses a highly scalable cloud-based infrastructure that can handle large volumes of data and high traffic. This ensures that the platform remains responsive and reliable even during peak trading times.

Real-Time Data Processing: TradingView processes real-time data feeds from multiple sources, using advanced algorithms to ensure data accuracy and timeliness. This is crucial for traders who rely on up-to-the-second information to make decisions.

Problem 3: Technological Complexity

Real-World Challenge:

Integrating and managing advanced technologies for real-time monitoring presents a significant challenge for TradingView.

TradingView's Solution:

Streaming Data: TradingView uses streaming data processing frameworks like Apache Kafka and Apache Flink. These frameworks handle the ingestion, processing, and delivery of real-time data streams.

Event-Driven Architecture: An event-driven architecture allows TradingView to process and respond to real-time events, such as price updates, efficiently.

Conclusion:

TradingView has established itself as a leading platform for traders and investors by addressing the real-time challenges of data volume, operational efficiency, compliance, and technological complexity. Its advanced charting tools, real-time data integration, and strong community make it a valuable resource for financial analysis and trading.

Top Features of TradingView:

Advanced Charting Tools: Users can create and customize charts with various technical indicators, drawing tools, and chart types (e.g., candlestick, line, bar charts).

Real-Time Data: Access to real-time data for various financial instruments, including stocks, forex, cryptocurrencies, and commodities.

Social Network for Traders: Users can publish and share their trading ideas, strategies, and analyses with the community. Traders can follow each other, comment on ideas, and participate in discussions.

Alerts and Notifications: Users can set up alerts based on price movements, technical indicators, and custom conditions. Alerts can be delivered via email, SMS, or in-app notifications.

Cross-Platform Access: TradingView is available on web, desktop (Windows and Mac), and mobile apps (iOS and Android), ensuring users can access their tools and data anytime, anywhere.

Schema Description:

TradingView's schema comprises numerous entities, each responsible for a specific aspect of the platform's functionality. These entities correspond to the Users, Accounts, Data, Indicators

Alerts, Trades, Watchlists, SocialInteractions, Messages and more. Each entity has specific attributes that describe its properties and relationships with other entities.

Users Entity:

This Entity stores essential information about each user, including their login credentials, roles, and subscription details.

- **user_id:** INTEGER (Primary Key) - Unique identifier for each user.
- **username**: The username chosen by the user.
- **Email:** The user's email address.
- **password:** The user's hashed password**.**
- **Role:** The role of the user (e.g., trader, admin).
- **subscription_plan:** The type of subscription plan the user has (free, pro).
- **created_at:** The date and time when the user account was created.

Accounts Entity:

This Entity links users to their brokerage accounts, detailing the broker and account type.

- **account_id:** INTEGER (Primary Key) - Unique identifier for each account.
- **user_id:** INTEGER (Foreign Key referencing Users.user_id) - Identifier for the user who owns the account.
- **broker_name:** The name of the brokerage linked to the account.
- **account_type:** The type of brokerage account (e.g., cash, margin).
- **created_at:** The date and time when the account was created.

Data Entity:

This Entity stores various types of financial data linked to user accounts, including real-time price and volume data.

- **data_id**: INTEGER (Primary Key) - Unique identifier for each data record.
- **account_id**: INTEGER (Foreign Key referencing Accounts.account_id) - Identifier for the account to which the data belongs.
- **data_type**: The type of data (e.g., price, volume).
- **timestamp**: The date and time when the data was recorded.
- **value**: The value of the data point.

Indicators Entity:

This Entity stores user-created indicators and their associated scripts and parameters.

- **indicator_id**: INTEGER (Primary Key, Auto Increment) - Unique identifier for each indicator.
- **user_id**: INTEGER (Foreign Key referencing Users.user_id) - Identifier for the user who created the indicator.
- **script_name**: The name of the script used for the indicator.
- **parameters**: The parameters used in the script.
- **created_at:** The date and time when the indicator was created.

Alerts Entity:

This Entity manages alerts set by users, including the conditions that trigger them and the actions taken.

- **alert_id**: INTEGER (Primary Key) - Unique identifier for each alert.
- **user_id**: INTEGER (Foreign Key referencing Users.user_id) - Identifier for the user who created the alert.
- **condition**: The condition that triggers the alert.
- **action**: The action to be taken when the alert is triggered.
- **created_at**: The date and time when the alert was created.

Trades Entity:

This Entity logs all trading activities, including trade details and timestamps.

- **trade_id**: INTEGER (Primary Key, Auto Increment) - Unique identifier for each trade.
- **user_id**: INTEGER (Foreign Key referencing Users.user_id) - Identifier for the user who executed the trade.
- **symbol**: The trading symbol of the instrument.
- **quantity**: The quantity of the instrument traded.
- **price**: The price at which the instrument was traded.
- **trade_date**: The date and time when the trade was executed.

Watchlists Entity:

This Entity contains user-created watchlists, allowing users to track specific stocks or financial instruments.

- **watchlist_id**: INTEGER (Primary Key, Auto Increment) - Unique identifier for each watchlist.
- **user_id**: INTEGER (Foreign Key referencing Users.user_id) - Identifier for the user who created the watchlist.
- **Name**: The name of the watchlist.
- **created_at**: The date and time when the watchlist was created.

SocialInteractions Entity:

This Entity records social interactions on the platform, such as posts and comments.

- **interaction_id**: INTEGER (Primary Key, Auto Increment) - Unique identifier for each social interaction.
- **user_id**: INTEGER (Foreign Key referencing Users.user_id) - Identifier for the user who created the interaction.
- **interaction_type**: The type of interaction (e.g., post, comment).
- **content**: The content of the interaction.
- **created_at**: The date and time when the interaction was created.

Messages Entity:

This Entity manages private messages between users on the platform.

- **message_id**: INTEGER (Primary Key) - Unique identifier for each message.

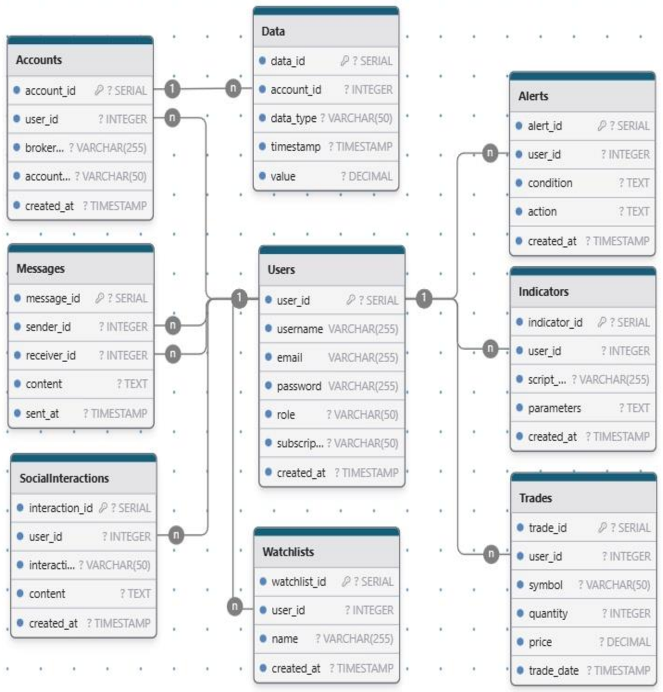
- **sender_id**: INTEGER (Foreign Key referencing Users.user_id) - Identifier for the user who sent the message.
- **receiver_id**: INTEGER (Foreign Key referencing Users.user_id) - Identifier for the user who received the message.
- **content**: The content of the message.
- **sent_at**: The date and time when the message was sent.

Relationship Details:

- **Users** can have multiple **Accounts**.
- **Accounts** are linked to multiple **Data** records.
- **Users** can create multiple **Indicators**, **Alerts**, **Trades**, **Watchlists**, and **SocialInteractions**.
- **Messages** can be sent from one user to another, involving both sender and receiver relationships.

ER Diagram:

Let's create an ER diagram to clearly depict the relationships and attributes of the entities within the TradingView schema. This visual representation will highlight the key components of the data model, helping you understand the complex interactions and connections that characterize the platform's structure. The diagram will serve as an illustrative tool, making it easier to comprehend how different entities interact and relate to each other within the system.



Accounts
account_id ? SERIAL
user_id ? INTEGER
broker... ? VARCHAR(255)
account... ? VARCHAR(50)
created_at ? TIMESTAMP
Data
data_id ? SERIAL
account_id ? INTEGER
data_type ? VARCHAR(50)
timestamp ? TIMESTAMP
value ? DECIMAL
Alerts
alert_id ? SERIAL
user_id ? INTEGER
condition ? TEXT
action ? TEXT
created_at ? TIMESTAMP
Messages
message_id ? SERIAL
sender_id ? INTEGER
receiver_id ? INTEGER
content ? TEXT
sent_at ? TIMESTAMP
Users
user_id ? SERIAL
username VARCHAR(255)
email VARCHAR(255)
password VARCHAR(255)
role ? VARCHAR(50)
subscrip... ? VARCHAR(50)
created_at ? TIMESTAMP
Indicators
indicator_id ? SERIAL
user_id ? INTEGER
script_... ? VARCHAR(255)
parameters ? TEXT
created_at ? TIMESTAMP
SocialInteractions
interaction_id ? SERIAL
user_id ? INTEGER
interacti... ? VARCHAR(50)
content ? TEXT
created_at ? TIMESTAMP
Watchlists
watchlist_id ? SERIAL
user_id ? INTEGER
name ? VARCHAR(255)
created_at ? TIMESTAMP
Trades
trade_id ? SERIAL
user_id ? INTEGER
symbol ? VARCHAR(50)
quantity ? INTEGER
price ? DECIMAL
trade_date ? TIMESTAMP
1
n