DATABASE SPECIFICATIONS

Mock Market
Dylan Porter

10/10/2021

DOCUMENT CONTROL

Work carried out by:

Name	Email Address	Other
Dylan Porter	**N/A**	N/A

Revision Sheet

Release No.	Date	Revision Description
1	**10/10/2021**	Added General Data requirements
2	10/17/2021	Finalized all general data activities

DATABASE SPECIFICATIONS

TABLE OF CONTENTS

MILESTONE 1: DATA REQUIREMENTS

Mock Market

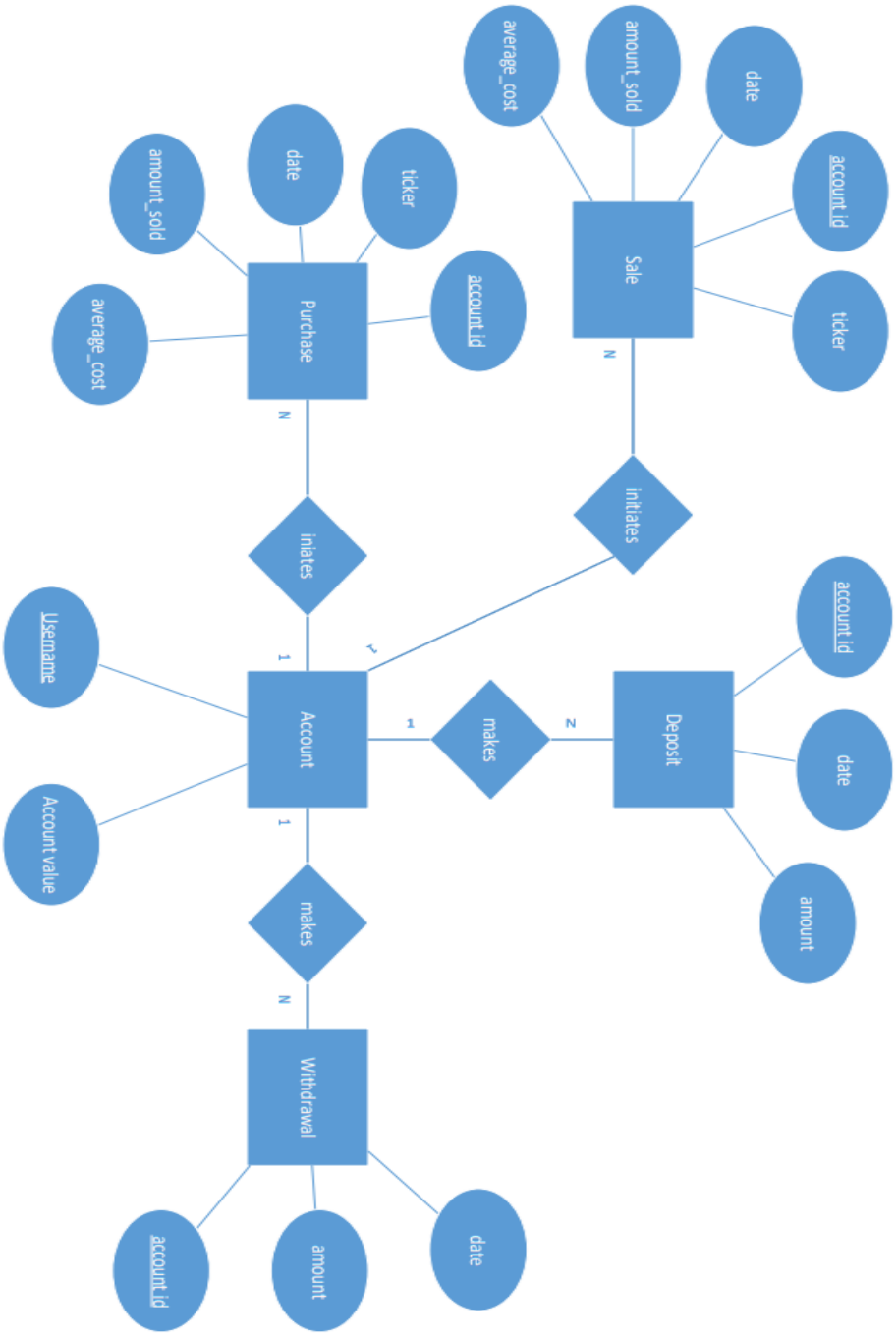
Online mocked market trading system.

Core requirements

No.	Requirement	Referenced page in SRS	Referenced Section in SRS	Referenced Paragraph in Section
1	Investor is a type of user in the system	3	1.2.	N/A
2	The system should store information about a purchase including: Time purchased, amount purchased, average cost, stock ticker, date.	3	1.2.	N/A
3	The system should store information about a sale including: Time sold, amount sold, average cost, stock ticker, date	3	1.2.	N/A
4	The system should store information about a deposit including: time of deposit, amount deposited, date.	3	1.3.	N/A
5	The system should store information about a withdrawal including: time of withdrawal, amount withdrawn,.	3	1.3.	N/A
6	The system should store information about Account including: username, funds available.	3	1.3.	N/A

MILESTONE 2: CONCEPTUAL DESIGN

Diagram

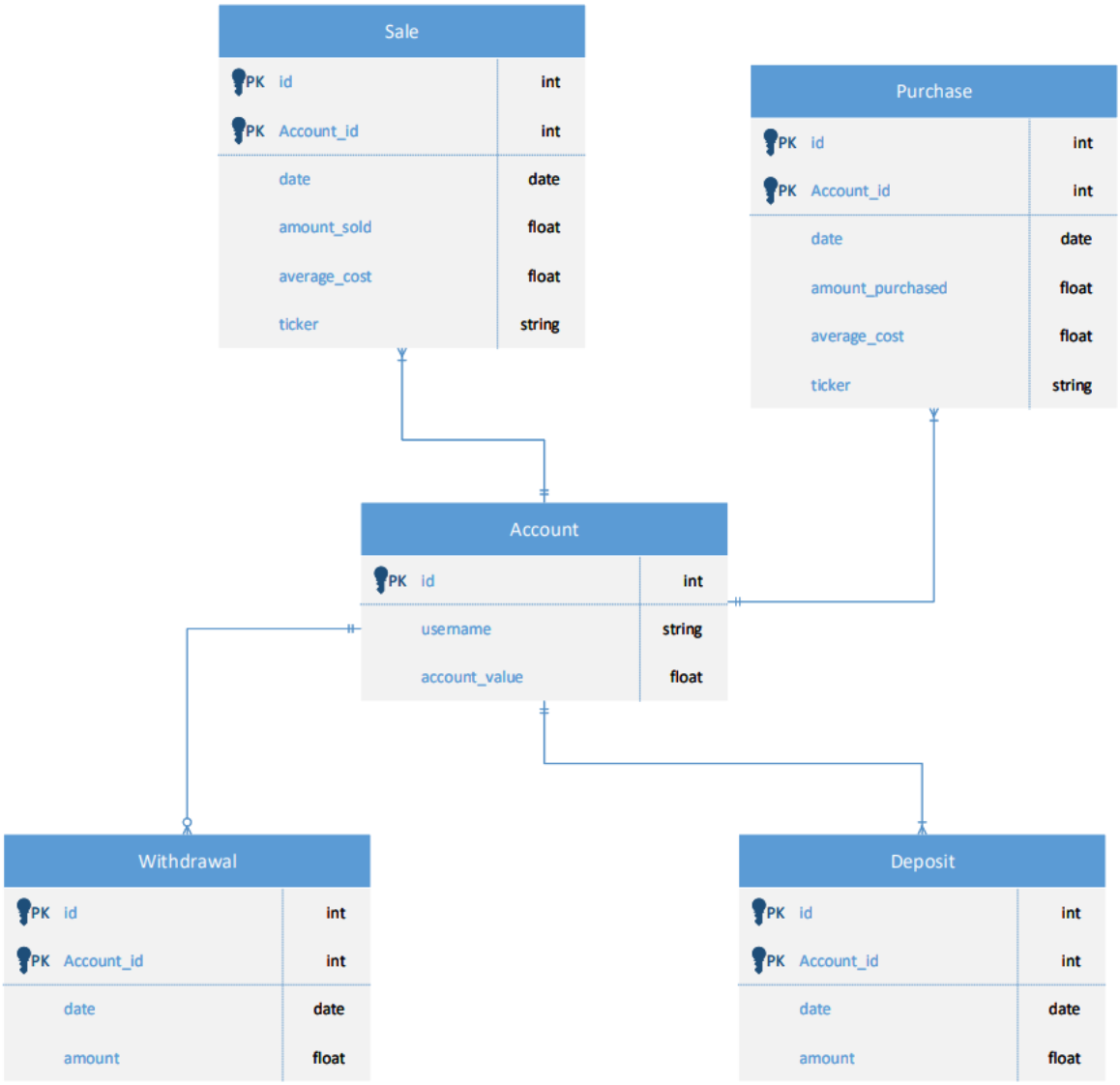


Assumptions and Constraints

- Primary key: account_id in Sale
- Primary key: account_id in Purchase
- Primary key: username in Account
- Primary key: account_id in Withdrawal
- Primary key: account_id in Deposit
- Deposit cannot be greater than $1,000,000
- Withdrawal cannot be greater than account value
- Deposit cannot be a non-positive number.
- Withdrawal cannot be a non-positive number.
- Purchase cannot be greater than account value.
- Sale cannot be greater than amount in securities.

MILESTONE 3: LOGICAL DESIGN

Entity Relationship Diagram



Assumptions and Constraints

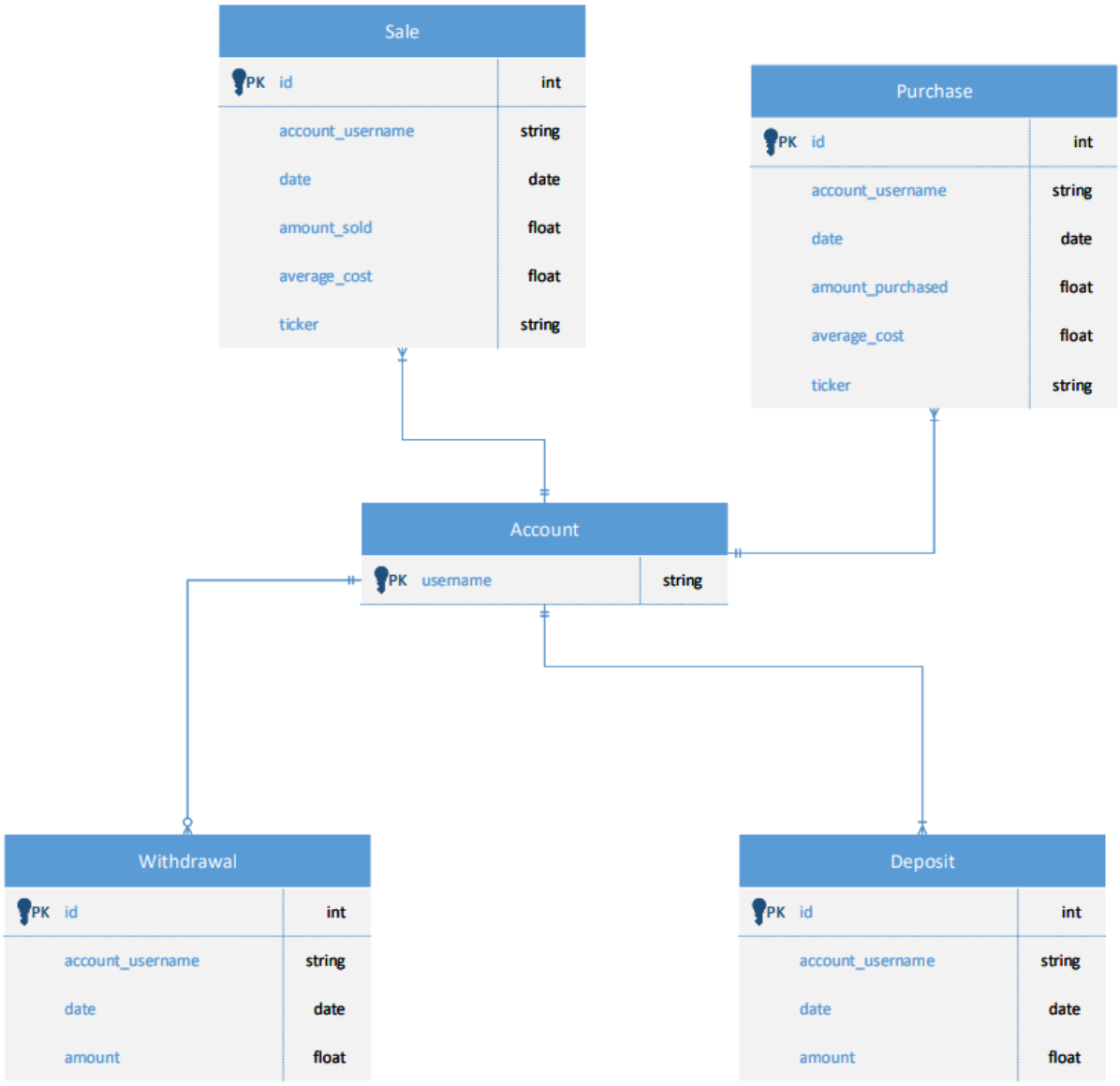
- There can be multiple withdrawals per account.
- There can be multiple deposits per account.
- There can be multiple sales per account.
- There can be multiple purchases per account.
- Account_id is a part of the composite key for Withdrawal, Deposit, Sale, and Purchase.

MILESTONE 4: NORMALIZATION AND

MILESTONE 5: PHYSICAL DESIGN

Assumptions and Constraints

Updated Diagram based on normalization:



Naming Conventions

Discuss the naming standards and conventions that you have used for table creation.

Tables

	Name of the table	***Account***			
	Description	An Account is the basis of the application. It is what we use to allow someone to login and make transactions.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	Username	The username of the account.	String	'burghporter'	Cannot be null. Must be unique.
	…				
	Functional Dependencies and Keys				
	Functional dependencies	Username → username			
	Candidate keys	**Username; Reason: {username}+ = {username}**			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	All the non-key attributes depend on the entire key.		
	3NF	**Yes**	All the non-key attributes depend only on the key.		
	BCNF	**Yes**	All the attributes depend only on a key.		
	Physical Design				
	Primary Key	Username			
	Foreign Keys	**none**			
	SQL Code	CREATE TABLE ACCOUNT(username VARCHAR(50) NOT NULL, PRIMARY KEY(username));			

	Name of the table	***Sale***			
	Description	A sale is when an account decides to sell a security in their portfolio.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	Id	Id of the sale.	Integer	Between 1 and 9999999999	Cannot be null. Must be unique.
	Account_username	The username of the account making the sale.	String	'burghporter'	-
	Date	The date that the sale took place	Varchar2(50)	10-06-26	Cannot be null; must be a valid date.
	Amount_sold	The amount of the security that was sold	Integer	Between 1 and 99999999999	Cannot be null.
	Average_cost	The average cost	Float	5.50	Cannot be null.

		when the security was sold.			
	Ticker	The name of the stock that was sold	String	'AMC'	Cannot be null.
	Functional Dependencies and Keys				
	Functional dependencies	Id → account_username id → date Id → amount_sold Id → average_cost Id → ticker			
	Candidate keys	**id; Reason: {id}+ = {id, account_username, date, amount_sold, average_cost, ticker}**			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	All the non-key attributes depend on the entire key.		
	3NF	**Yes**	All the non-key attributes depend only on the key.		
	BCNF	**Yes**	All the attributes depend only on a key.		
	Physical Design				
	Primary Key	Id			
	Foreign Keys	**Account_username**			
	SQL Code	CREATE TABLE SALE(id SERIAL PRIMARY KEY, account_username VARCHAR(50) NOT NULL, date TIMESTAMP NOT NULL, amount_sold INTEGER NOT NULL, average_cost FLOAT NOT NULL, ticker VARCHAR(20) NOT NULL, FOREIGN KEY(account_username) REFERENCES ACCOUNT(username));			

	Name of the table	***Purchase***			
	Description	A purchase is when an account decides to buy a security in their portfolio.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	Id	Id of the sale.	Integer	Between 1 and 9999999999	Cannot be null. Must be unique.
	Account_username	The username of the account making the purchase.	String	'burghporter'	-
	Date	The date that the purchase took place	Varchar2(50)	10-06-26	Cannot be null; must be a valid date.
	Amount_purcha	The amount of the	Integer	Between 1 and	Cannot be null.

	sed	security that was purchase		99999999999	
	Average_cost	The average cost when the security was bought.	Float	5.50	Cannot be null.
	Ticker	The name of the stock that was bought	String	'AMC'	Cannot be null.
	Functional Dependencies and Keys				
	Functional dependencies	Id → account_username id → date Id → amount_purchased Id → average_cost Id → ticker			
	Candidate keys	**id; Reason: {id}+ = {id, account_username, date, amount_purchased, average_cost, ticker}**			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	All the non-key attributes depend on the entire key.		
	3NF	**Yes**	All the non-key attributes depend only on the key.		
	BCNF	**Yes**	All the attributes depend only on a key.		
	Physical Design				
	Primary Key	Id			
	Foreign Keys	**Account_username**			
	SQL Code	CREATE TABLE PURCHASE(id SERIAL PRIMARY KEY, account_username VARCHAR(50) NOT NULL, date TIMESTAMP NOT NULL, amount_purchased INTEGER NOT NULL, average_cost FLOAT NOT NULL, ticker VARCHAR(20) NOT NULL, FOREIGN KEY(account_username) REFERENCES ACCOUNT(username));			

	Name of the table	***Withdrawal***			
	Description	A withdrawal is when an account decides to withdrawal money from their account.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	Id	Id of the withdrawal.	Integer	Between 1 and 9999999999	Cannot be null. Must be unique.
	Account_username	The username of the account making the withdrawal.	String	'burghporter'	-
	Date	The date that the	Varchar2(50)	10-06-26	Cannot be null;

		withdrawal took place			must be a valid date.
	Amount	The amount that was withdrawn	Float	5.50	Cannot be null.
	Functional Dependencies and Keys				
	Functional dependencies	Id → account_username id → date Id → amount			
	Candidate keys	**id; Reason: {id}+ = {id, account_username, date, amount}**			
	Normalization				
	1NF	**Yes**	All cells contain a unique value		
	2NF	**Yes**	All the non-key attributes depend on the entire key.		
	3NF	**Yes**	All the non-key attributes depend only on the key.		
	BCNF	**Yes**	All the attributes depend only on a key.		
	Physical Design				
	Primary Key	Id			
	Foreign Keys	**Account_username**			
	SQL Code	CREATE TABLE WITHDRAWAL(id SERIAL PRIMARY KEY, account_username VARCHAR(50) NOT NULL, date TIMESTAMP NOT NULL, amount INTEGER NOT NULL, FOREIGN KEY(account_username) REFERENCES ACCOUNT(username));			

	Name of the table	***Deposit***			
	Description	A deposit is when an account decides to deposit money into their account.			
	Attribute	**Description**	**Type**	**Examples of values**	**Notes**
	Id	Id of the deposit.	Integer	Between 1 and 9999999999	Cannot be null. Must be unique.
	Account_username	The username of the account making the deposit.	String	'burghporter'	-
	Date	The date that the deposit took place	Varchar2(50)	10-06-26	Cannot be null; must be a valid date.
	Amount	The amount that was deposited	Float	5.50	Cannot be null.
	Functional Dependencies and Keys				
	Functional dependencies	Id → account_username id → date Id → amount			
	Candidate keys	**id; Reason: {id}+ = {id, account_username, date, amount}**			

	Normalization		
	1NF	**Yes**	All cells contain a unique value
	2NF	**Yes**	All the non-key attributes depend on the entire key.
	3NF	**Yes**	All the non-key attributes depend only on the key.
	BCNF	**Yes**	All the attributes depend only on a key.
	Physical Design		
	Primary Key	Id	
	Foreign Keys	**Account_username**	
	SQL Code	CREATE TABLE DEPOSIT(id SERIAL PRIMARY KEY, account_username VARCHAR(50) NOT NULL, date TIMESTAMP NOT NULL, amount INTEGER NOT NULL, FOREIGN KEY(account_username) REFERENCES ACCOUNT(username));	