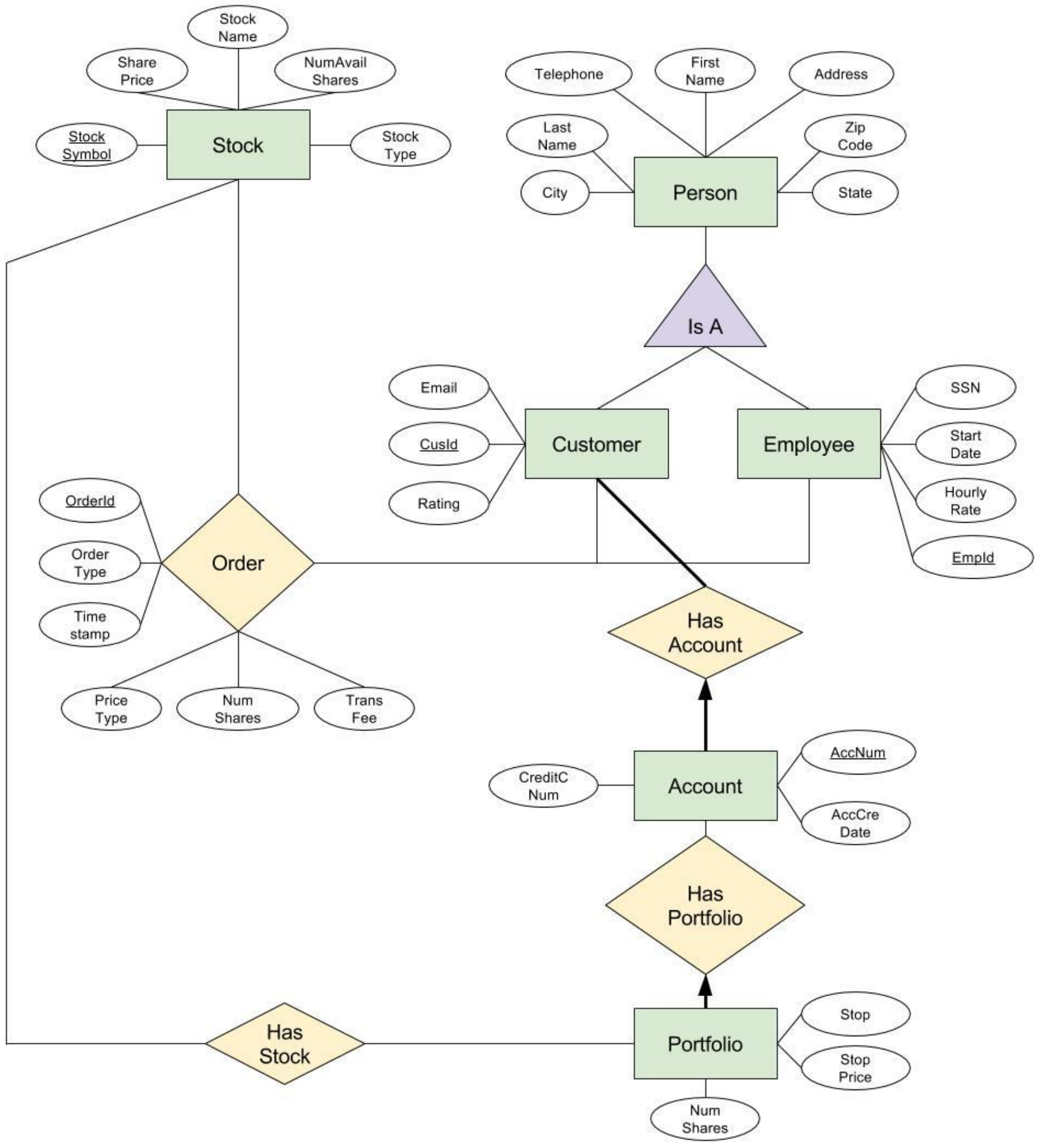
LIVESTOCK

CSE 305: Assignment 1

Wendy Zheng: wendy.zheng@stonybrook.edu
Ryan Goepfert: ryan.goepfert@stonybrook.edu
Yuliya Astapova: yuliya.astapova@stonybrook.edu

E-R Diagram



E-R Diagram Rationale

We created an entity titled Person, which is a superclass of both Customer and Employee. Person contains the attributes common to both of these classes, helping to prevent the duplication of data. Customer and Employee have attributes unique to themselves. CusId and EmpId are the primary keys to their respective entities, as each customer and employee have a unique identification number.

A customer can have any number of accounts, but must have at least one, while each account must be linked to only one customer. Therefore, Customer to HasAccount has a bold line indicating a participation constraint to Account, while Account to HasAccount has a bolded arrow indicating a participation and key constraint to Customer.

Each account is represented by a unique account number, which is the primary key. Each account has a stock portfolio, so we created an entity Portfolio that has stocks. An account with no stocks does not have a stock portfolio, so there are no constraints. A portfolio is linked to exactly one account, so there is a participation and key constraint from Portfolio to Account, indicated by a bolded arrow to HasPortfolio.

Portfolio may contain any number of stocks or none at all, so there are no constraints from Portfolio to entity Stock. A stock can belong to any number of portfolios or none at all, so there are no constraints from Stock to Portfolio. Each stock has a unique stock symbol, which is Stock's primary key.

Both customers and employees can place an order for a stock, but may also never place one. We created relationship Order to represent the customers and employees making orders for stocks. Each Order has a unique Id number, so OrderId is its primary key.

Relational Model

```
CREATE TABLE ORDER (
	OrderId			INTEGER AUTO_INCREMENT,
	StockSymbol		VARCHAR(5) NOT NULL,
	OrderType		OrderTypes NOT NULL,
	NumShares		INTEGER,
	CusAccNum		INTEGER DEFAULT 0 NOT NULL,
	Timestamp		DATETIME NOT NULL,
	TransFee		FLOAT(2),
	PriceType		PriceTypes,
	EmpId			INTEGER DEFAULT 0 NOT NULL,
	PRIMARY KEY (OrderId),
	UNIQUE (StockSymbol, Timestamp, CusAccNum, EmpId),
	FOREIGN KEY (StockSymbol) REFERENCES STOCK
		ON DELETE SET NULL
		ON UPDATE CASCADE,
	FOREIGN KEY (CusAccNum) REFERENCES ACCOUNT(AccNum)
		ON DELETE SET DEFAULT
		ON UPDATE CASCADE,
	FOREIGN KEY (EmpId) REFERENCES EMPLOYEE
		ON DELETE SET NULL
		ON UPDATE CASCADE,
	CHECK (NumShares > 0),
	CHECK (TransFee > 0)
)

CREATE DOMAIN OrderTypes VARCHAR(4)
	CHECK (VALUE IN ('Buy', 'Sell'))

CREATE DOMAIN PriceTypes VARCHAR(15)
	CHECK (VALUE IN ('Market', 'Market on Close', 'Trailing Stop', 'Hidden Stop'))
```

```
CREATE ASSERTION ValidNumShares
        CHECK (NOT EXISTS
                (SELECT * FROM Order, Stock
                WHERE ORDER.NumShares <= STOCK.NumAvailShares
                AND ORDER.StockSymbol = STOCK.StockSymbol))

CREATE TABLE STOCK (
        StockSymbol             VARCHAR(5),
        StockName               VARCHAR(20) NOT NULL,
        StockType               VARCHAR(20),
        SharePrice              FLOAT(2) NOT NULL,
        NumAvailShares          INTEGER NOT NULL,
        PRIMARY KEY (StockSymbol)
        UNIQUE (StockName)
        CHECK (SharePrice > 0),
        CHECK (NumAvailShares > -1)
)

CREATE TABLE CUSTOMER (
        LastName                VARCHAR(20) NOT NULL,
        FirstName               VARCHAR(20) NOT NULL,
        Address                 VARCHAR(50),
        City                    VARCHAR(20),
        State                   VARCHAR(20),
        ZipCode                 CHAR(5),
        Telephone               CHAR(10),
        Email                   VARCHAR(50),
        Rating                  Ratings,
        CusId                   INTEGER AUTO_INCREMENT,
        PRIMARY KEY (CusId)
)

CREATE DOMAIN Ratings INTEGER
        CHECK(VALUE > -1 AND VALUE < 11)

CREATE TABLE EMPLOYEE (
        SSN                     CHAR(9) NOT NULL,
        LastName                VARCHAR(20),
        FirstName               VARCHAR(20),
        Address                 VARCHAR(50),
        City                    VARCHAR(20),
        State                   VARCHAR(20),
        ZipCode                 CHAR(5),
        Telephone               CHAR(10),
        StartDate               DATETIME,
        HourlyRate              FLOAT(2),
        EmpId               INTEGER AUTO_INCREMENT,
        PRIMARY KEY (EmpId),
        UNIQUE (SSN)
)

CREATE TABLE PORTFOLIO (
        AccNum              INTEGER,
        StockSymbol         CHAR(5),
        NumShares           INTEGER,
        Stop                Stops,
        StopPrice           FLOAT(2),
        PRIMARY KEY (AccNum, StockSymbol)
        FOREIGN KEY (AccNum) REFERENCES ACCOUNT
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY (StockSymbol) REFERENCES STOCK
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        CHECK (NumShares > -1)
)

CREATE TRIGGER DeleteOnZeroShares
        AFTER UPDATE OF NumShares ON PORTFOLIO
        FOR EACH STATEMENT
                DELETE FROM PORTFOLIO
                    WHERE
                        EXISTS (SELECT * FROM PORTFOLIO P
                            WHERE P.NumShares = 0)

CREATE DOMAIN Stops
        CHECK(VALUE IN ('Trailing', 'Hidden'))

CREATE TABLE ACCOUNT (
        AccNum              INTEGER AUTO_INCREMENT,
        AccCreDate          DATETIME,
        CreditCNum          VARCHAR(16) NOT NULL,
        CusId               INTEGER NOT NULL,
        PRIMARY KEY (AccNum),
        FOREIGN KEY (CusId) REFERENCES CUSTOMER
                ON DELETE NO ACTION
                ON UPDATE CASCADE
)
```

Relational Model Rationale

In our relational model, we decided to create a total of six tables: Order, Stock, Customer, Employee, Account, and Portfolio. The primary key of Order is the OrderId, which we set to auto-increment each time an order is created. One candidate key of Order is the tuple containing the attributes StockSymbol, Timestamp, CusAccNum, and EmpId, because it would not make sense for someone to order the same stock multiple times, but at the exact same time. We set the default value of EmpId to 0 to account for the cases in which an employee is not part of the transaction. The foreign keys of Order are StockSymbol (from Stock), CusAccNum (AccNum from Account), and EmpId (from Employee). We chose the trigger in StockSymbol to be ON DELETE SET NULL, because setting it to a default would not work reliably, since a new stock may accidentally have the default name, and no action would not work because a Stock must be able to be deleted in the case that a company goes private or bankrupt. We chose ON DELETE SET DEFAULT for CusAccNum instead of no action because since customers can close accounts, accounts must therefore be deletable, but set default allows us to easily refer to all closed accounts. We chose ON DELETE SET NULL for EmpId because no action would not work in case an employee leaves the company and must be deleted, and set default would not work because we want to differentiate between when an Employee is deleted and when they are not present for a transaction. The OrderType attribute has a domain associated with it, which only includes “Buy” and “Sell” as values. PriceTypes has a domain with four possible price types in it. We have also created an assertion to ensure that the NumShares of an Order does not exceed the number of available shares for that Stock.

In the Stock table, we have StockSymbol as the primary key and StockName as the candidate key. We chose this candidate key because a company cannot have two different stocks. Also, we have decided that the number of shares for a stock should refer to the number of shares that are currently available for purchase, reflected in the NumAvailShares attribute.

The Customer table is very straightforward; it includes all of a customer’s information, along with an auto-incrementing CusId as the primary key. We did not include a candidate key because customers can have the same first and last name or the same contact information, but still be different people. The domain included for the Rating simply ensures that it is a number between 1 and 10.

The Employee table is also straightforward; it includes all of an employee’s information, with an auto-incrementing EmpId as the primary key. We have also made SSN a candidate key, because no two people can have the same social security number. We did not create other candidate keys, because people can have the same names but be different people, as with Customer. For both Employee and Customer, we assume that all users have a United States format phone number.

Our Portfolio table represents the ownership of shares of a certain Stock by a certain customer’s Account. The primary key of Portfolio is the tuple of AccNum and StockSymbol, because each account only has one possible entry for each Stock, though the number of shares it owns can change. AccNum and StockSymbol are also both foreign keys. We chose the trigger ON DELETE NO ACTION for AccNum, because we have decided that an Account cannot be deleted if it still has Portfolios attached to it. For StockSymbol, we chose the trigger ON DELETE CASCADE because if a Stock is taken off the market, that should be reflected in Portfolios. We have also included a domain for Stop, which includes “Trailing” and “Hidden” as values. We created a trigger, DeleteOnZeroShares, for the Portfolio table to delete a portfolio when the number of shares it contains hits zero.

Our final table is Account, which includes CusId as a foreign key in order to refer to the customer that owns the account. We chose ON DELETE NO ACTION for this foreign key because we decided that a Customer cannot be deleted if he/she still has open accounts, because this means that he/she still has stocks in the portfolios associated with it. The primary key of Account is the auto-incrementing AccNum, which counts all accounts in our database, not just those of a particular Customer.

Throughout our relational model, we have made all CHAR attributes into VARCHARs because they preserve space when not filled to capacity. Also, in many cases, we chose VARCHAR rather than INTEGER for an attribute, such as with credit card numbers, IDs, and phone numbers, because they need to have a certain number of digits. In those cases, it is our intent to check whether or not correct values are entered through the form in which users add data.

Collaboration Plan

In order to best determine the representation of the tables that would become the foundation of our project, we each came up with a collection of tables that could loosely be used to define the backbone of our E-R Diagram. By comparing and contrasting each of our tables, we were able to see the flaws while picking out the strengths of our respective tables. By combining the best of our interpretations, we were able to map out the E-R diagram as depicted by our document.

Now that we had an E-R diagram for reference, we began our quest of translation. Following the usual algorithm for translation was no easy task. Ryan wrote a first draft of the relational model, including all of the tables and their attributes, types, keys, and several constraints. He also listed several concerns that needed to be addressed that he himself was unsure of. Wendy and Yuliya then went over Ryan's work and made changes to attribute types, added triggers, domains, some checks, fixed some of the keys and constraints, and addressed the issues that Ryan listed.

Upon the completion of the translation of our E-R Diagram to the respective relation schemas, we decided to compare both documents to make sure that nothing was lost in the translation process. Finally, in writing the rationales, Ryan wrote the E-R Diagram Rationale, Yuliya wrote the Relational Model Rationale, Wendy wrote the Collaboration Plan, and we each proofread and edited each other's writing.