ER Diagram to Relational Model

LEGEND: (Schema = ER Diagram)
InstitutionID = Inst_ID
Debt-Equity_ratio = D/E Ratio
T_id = Trans_id

Manager (UserId:char(20), Password:char(15), Name:char(15), Phone:Integer, Email:char(30), PortfoliosManaged:Integer)
Primary Key: UserId
Alternate Key: (Phone, Name)

Customer (UserId:Integer, Password:char(15), Name:char(15), Phone:Integer, Email:char(30), FICO_Score:Integer, **InstitutionId:Char(30)**)
Primary Key: UserId
Alternate Key: (Phone, Name)
Foreign Key: InstitutionId REFERENCES CREDITOR

Creditor (InstitutionId:Char(30), Amount_Issued:Float, Institution:char(30))
Primary Key: InstitutionId

Leverage (CreditId:Integer, **InstitutionId:Char(30), UserId:Integer**, Interest_Rate:Float, Amount:Integer, Safety_Margin:Float, Debt-Equity_ratio:Float)
Primary Key: (CreditId, InstitutionId, UserId)
Foreign Key: InstitutionId REFERENCES CREDITOR
UserId REFERENCES CUSTOMER

Portfolio (Pid:Integer, **CustomerId:Integer**, Date:Date, Balance:Float, Since:Date, **ManagerId:Integer**)
Primary Key: (Pid, CustomerId)
Foreign Key: CustomerId REFERENCES CUSTOMER
ManagerId REFERENCES MANAGER

Contains (**Pid:Integer**, **UserId:Integer**, **Ticker: char(4)**)
Primary Key: (Pid, UserId, Ticker)
Foreign Key: (Pid, CustomerId) REFERENCES PORTFOLIO
CustomerId REFERENCES CUSTOMER
Ticker REFERENCES STOCKS

Stocks (Ticker:char(4), Price:Float, Trade_Index:char(15), **Industry:char(16)**, **Name:char(16),** report_Id:Integer, EPS:Float, Date:Date, ROI:Float, P/E_Ratio:Float)
Primary Key: Ticker
Foreign Key: (Industry, Name) REFERENCES COMPANY

Company (Name:char(16), Industry:char(16), Shares_Outstanding:Integer, Market_Cap:Integer, **Ticker:char(4)**)
Primary Key: (Name, Industry)
Foreign Key: Ticker REFERENCES STOCKS

Dividends(T_id:Integer, Div_Yield:Float, **Pid:Integer**, **UserId:Integer, Industry:char, Name:char**)
Primary Key: (Tid, Pid, UserId, Industry, Name)
Foreign Key: (Industry, Name) REFERENCES COMPANY
Pid REFERENCES PORTFOLIO
UserId REFERENCES CUSTOMER

FUNCTIONAL DEPENDENCIES & NORMALIZATION

Manager:
UserId -> Password, Name, Phone
(Phone, Name) -> UserId

Normalization:
1. UserId -> Password, Name, Phone:

FD is non-trivial and UserId is a PK & SK, so this FD does not violate BCNF

2. (Phone, Name) -> UserId:

FD is non-trivial
Since (Phone, Name) is an alternate key, it is an SK of the table
So this FD does not violate BCNF

Since both FDs do not violate BCNF, **Manager** Table is already in BCNF.

Customer:
UserId -> Password, Name, Phone, FICO_Score
(Phone, Name) -> UserId

Normalization:
1. UserId -> Password, Name, Phone, FICO_Score:

FD is non-trivial, and UserId is a PK & SK, so this FD does not violate BCNF

2. (Phone, Name) -> UserId:

FD is non-trivial
Since (Phone, Name) is an alternate key, it is an SK of the table
So this FD does not violate BCNF

Since both FDs do not violate BCNF, **Customer** Table is already in BCNF.

Creditor:
InstitutionId -> Institution (eg. JP-PrivateEquity will determine JP-Morgan)

Normalization:
1. InstitutionId -> Institution:

FD is non-trivial, and InstitutionId is a PK & SK, so this FD does not violate BCNF

Since there is only one FD and it does not violate BCNF, **Creditor** Table is already in BCNF.

Leverage:
CreditId -> InstitutionId, UserId, Interest_Rate, Safety_Margin, D-E_Ratio, Amount
Safety_Margin -> Interest_Rate, D-E_Ratio

Normalization:
1. CreditId -> InstitutionId, UserId, Interest_Rate, Safety_Margin, D-E_Ratio, Amount:

FD is non-trivial, and CreditId is a PK & SK, so this FD does not violate BCNF

2. Safety_Margin -> Interest_Rate, D-E_Ratio

FD is non-trivial
Since Safety_Margin is not a SK
This FD violates BCNF

Decomposition:
R1= (Safety_Margin, Interest_Rate, D-E_Ratio)
R2=(CreditId, **InstitutionId**, **UserId**, Amount, Safety_Margin)

R1:
Safety_Margin -> Interest_Rate, D-E_Ratio, and Safety_Margin is the PK & SK
So R1 is now in BCNF

R2:
CreditId -> InstitutionId, UserId, Safety_Margin, Amount
CreditId is PK, and hence SK, so this FD does not violate BCNF
So R2 is now in BCNF

So **Leverage** Table is decomposed to:
R1 = (Safety_Margin, Interest_Rate, D-E_Ratio)
R2 =(CreditId, **InstitutionId**, **UserId**, Amount, Safety_Margin)

Lossless Check:
Since Interest_Rate and D-E_Ratio are not in R2,
CreditId does not directly determine Interest_Rate and D-E_Ratio
However, CreditId -> Safety_Margin and Safety_Margin -> Interest_Rate, D-E_Ratio
By transitivity, CreditId -> Interest_Rate, D-E_Ratio
Since the above FD is preserved and no other FDs are affected,
The decomposition is a lossless join

Portfolio:
Pid -> Date, Balance, Since, ManagerId

Normalization:
Pid -> Date, Balance, Since, ManagerId:
FD is non-trivial
Since the PK is (Pid, CustomerId), Pid by itself is not a SK
This FD violates BCNF

Decomposition:
R1 = (Pid, Date, Balance, Since, **ManagerId**)
R2 = (Pid, **CustomerId**)

R1:
Pid -> Date, Balance, Since, ManagerId
FD is non-trivial, and Pid is the SK, and hence SK, so R1 is in BCNF

R2:
Since there are no FD and the PK is all the attributes, which is (Pid, CustomerId)
R2 is in BCNF

So **Portfolio** Table is decomposed to:
R1 = (Pid, Date, Balance, Since, **ManagerId**)
R2 = (Pid, **CustomerId**)

Lossless Check:
Pid -> Date, Balance, Since, ManagerId is preserved in R1 and there is no other FD
So the decomposition is a lossless join

Contains:
No FD

Stocks:
Ticker - > Price, Trade_Index, reportId, ROI, P/E_Ratio, Date, EPS, Name, Industry
reportId -> EPS, Date, ROI, P/E_Ratio
(Price, EPS) -> P/E_Ratio
(Name, Industry) -> Trade_Index

Normalization:
Since Ticker -> reportId and reportId -> EPS, Date, ROI, P/E_Ratio,
By transitivity, Ticker -> EPS, Date, ROI, P/E_Ratio
Hence, Ticker -> EPS, Date, ROI, P/E_Ratio is redundant
The first FD can be reduced to:
Ticker - > Price, Trade_Index, Report_Id, Name, Industry

Ticker - > Price, Trade_Index, Report_Id, Name, Industry:
The FD is non-trivial
Since Ticker is PK, and hence SK, this FD does not violate BCNF

reportId -> EPS, Date, ROI, P/E_Ratio:
The FD is non-trivial
Since the PK is Ticker, reportId is not an SK, so this FD violates BCNF

Decomposition:
R1 = (<u>reportId</u>, EPS, Date, ROI, P/E_Ratio)
R2 = (<u>Ticker</u>, Price, Trade_Index, Industry, Name, reportId)

R1:
reportId -> EPS, Date, ROI, P/E_Ratio:
The FD is not-trivial, and reportId is PK & SK, so R1 is in BCNF

R2:
Ticker - > Price, Trade_Index, Report_Id, Name, Industry:
This FD is checked and does not violate BCNF

(Name, Industry) -> Trade_Index:
The FD is non-trivial
Since the CK is Ticker, (Name, Industry) is not a SK, so this FD violates BCNF

R3 = (<u>Name</u>, <u>Industry</u>, Trade_Index)
R4 = (<u>Ticker</u>, Price, **Industry**, **Name**, reportId)

R3:
(Name, Industry) -> Trade_Index:
The FD is non-trivial, but (Name, Industry) is PK, so this FD does not violate BCNF

R4:
Ticker - > Price, reportId, Name, Industry:
This FD is checked and does not violate BCNF

So **Stocks** Table is decomposed to (BCNF):
R1 = (<u>reportId</u>, EPS, Date, ROI, P/E_Ratio)
R3 = (<u>**Name**</u>, <u>**Industry**</u>, Trade_Index)
R4 = (<u>Ticker</u>, Price, **Industry**, **Name**, reportId)

Lossless Check:
Ticker - > Price, Report_Id, Name, Industry is preserved in R4

Ticker cannot directly determine Trade_Index
However, Since Ticker -> Name, Industry and (Name, Industry) -> Trade_Index
By transitivity, Ticker -> Trade_Index
Hence, this FD is not lost

reportId -> EPS, Date, ROI, P/E_Ratio is perserved in R1
(Name, Industry) -> Trade_Index is preserved in R3

However, (Price, EPS) -> P/E_Ratio is lost during the BCNF decomposition
Hence, a new relation is needed:
R5 = (<u>Price</u>, <u>EPS</u>, P/E_Ratio)

So **Stocks** Table needs to be decomposed to 3NF to be lossless:
R1 = (<u>reportId</u>, EPS, Date, ROI, P/E_Ratio)
R3 = (<u>**Name**</u>, <u>**Industry**</u>, Trade_Index)
R4 = (<u>Ticker</u>, Price, **Industry**, **Name**, reportId)
R5 = (<u>Price</u>, <u>EPS</u>, P/E_Ratio)

Company:
(Industry, Name) - > Market_Cap, Shares_Outstanding

Normalization:
(Industry, Name) - > Market_Cap, Shares_Outstanding:
The FD is non-trivial
Since (Industry, Name) is PK, and hence SK, this FD does not violate BCNF

Dividends:
T_id -> Dividend_Yield

Normalization:
The FD is non-trivial
Since the PK is (T_id, Pid, UserId, Industry, Name), T_id by itself is not an SK,
This FD violates BCNF

Decomposition:
R1 = (<u>**T_id**</u>, Dividend_Yield)
R2 = (<u>T_id</u>, <u>**Pid, UserId, Industry, Name**</u>)

R1:
T_id -> Dividend_Yield:
The FD is non-trivial, and T_id is PK, and hence SK, so R1 is in BCNF

R2:
There is no FD in R2, hence R2 is in BCNF

So **Dividends** Table is decomposed to:
R1 = (<u>**T_id**</u>, Dividend_Yield)
R2 = (<u>T_id</u>, <u>**Pid, UserId, Industry, Name**</u>)

Lossless Check:
T_id -> Dividend_Yield is preserved in R1

So the decomposition is lossless.

SQL DDL TABLES

Manager

```
CREATE TABLE Manager
	(UserId:		CHAR(20),
	Password:		CHAR(15),
	Name:			CHAR(15),
	Phone:				INTEGER,
	Email:			CHAR(30),
	Portfolio_Manager:	INTEGER,
	PRIMARY KEY		(UserId),
	UNIQUE			(Phone, Email));
```

Customer

```
CREATE TABLE Customer
	(UserId:		INTEGER,
	Password:		CHAR(15),
	Name:			CHAR(15),
	Phone:				INTEGER,
	Email:			CHAR(30),
	FICO_Score:	INTEGER,
	InstitutionId:	CHAR(30),
	PRIMARY KEY		(UserId),
	UNIQUE			(Phone, Email),
	FOREIGN KEY		(InstitutionId)		REFERENCES Creditor,
			ON DELETE SET NULL //If a credit is lost, we want to be able to say
					// there is no credit instead of rejecting that credit
					// update/delete and also not set default because
					// that means there is still a credit when there isn't
			ON UPDATE CASCADE);
```

Creditor

```
CREATE TABLE Creditor(
	InstitutionId:		CHAR(30),
	Amount_Issued:		FLOAT,
	Institution:		CHAR(30),
	PRIMARY KEY			(InstitutionId));
```

Leverage

```
CREATE TABLE Leverage_R1(
	Safety_Margin:		FLOAT,
	Interest_Rate:		FLOAT,
	D-E_Ratio:		FLOAT,
	PRIMARY KEY (Safety_Margin));


CREATE TABLE Leverage_R2(
	CreditId:		INTEGER,
	InstitutionId:		CHAR(30),
```

```
        UserId:             INTEGER,
        Amount:             INTEGER,
        Safety_Margin:      FLOAT,
        PRIMARY KEY(CreditId, InstitutionId, UserId),
        FOREIGN KEY(InstitutionId)
                REFERENCES CREDITOR,
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
        FOREIGN KEY(UserId)
                REFERENCES CUSTOMER
                        ON DELETE CASCADE
                        ON UPDATE CASCADE);
```

Stocks

```
CREATE TABLE Stocks_R1(
        Report_Id:      INTEGER,
        EPS:            FLOAT,
        Date:           DATE,
        ROI:            FLOAT,
        P/E_Ratio:      FLOAT,
        PRIMARY KEY:            (Report_Id));

CREATE TABLE Stocks_R3(
        Name:           CHAR(16),
        Industry:       CHAR(16)
        Trade_Index:    CHAR(15));
        PRIMARY KEY             (Name, Industry),
        FOREIGN  KEY            (Name, Industry)
                        REFERENCES Company
                                ON DELETE CASCADE
                                ON UPDATE CASCADE);

CREATE TABLE Stocks_R4
        (Ticker         CHAR(4),
        Price           FLOAT,
        Industry        CHAR(16),
        Name            CHAR(16),
        Report_id       INTEGER,
        PRIMARY KEY             (Ticker)
        FOREIGN KEY             (Name, Industry)
                        REFERENCES Company
                                ON DELETE NO ACTION
                                ON UPDATE CASCADE);

CREATE TABLE Stocks_R5
        (Price          FLOAT,
        EPS             FLOAT,
        P/E_Ratio       FLOAT,
        PRIMARY KEY             (Price, EPS));
```

Contains

```
CREATE TABLE Contains
	(Pid:		INTEGER,
	UserId:		INTEGER,
	Ticker:		CHAR(4),
	PRIMARY KEY	(Pid, UserId, Ticker),
	FOREIGN KEY	(Pid, CustomerId)
			REFERENCES PORTFOLIO,
				ON DELETE
				ON UPDATE
	FOREIGN KEY	 (CustomerId)
			REFERENCES CUSTOMER,
				ON DELETE CASCADE
				ON UPDATE CASCADE
	FOREIGN KEY (Ticker)
			REFERENCES STOCKS
				ON DELETE NO ACTION
				ON UPDATE CASCADE);
```

Company

```
CREATE TABLE Company
	(Name:			CHAR(16)	NOT NULL,
	Industry:		CHAR(16)	NOT NULL,
	Shares_Outstanding:	INTEGER,
	Market_Cap:		INTEGER,
	Ticker:				CHAR(4),
	PRIMARY KEY		(Name, Industry)
	UNIQUE			(Name, Ticker)
	FOREIGN KEY(Ticker)
		REFERENCES STOCKS
			ON DELETE NO ACTION
			// we wouldnt delete a company if the ticker is deleted, reject the deletion
			ON UPDATE CASCADE)
```

Dividends

```
CREATE TABLE Dividends_R1(
	T_id:		INTEGER,
	Dividend_Yield: FLOAT,
	PRIMARY KEY	(T_id));

CREATE TABLE Dividends_R2(
	T_id: INTEGER,
	Pid: INTEGER,
	UserId: INTEGER,
	Industry: CHAR(30),
	Name: CHAT(15),
	PRIMARY KEY(T_id, Pid, UserId, Industry, Name),
	FOREIGN KEY(Pid)
		REFERENCES PORTFOLIO
```

```
                ON DELETE CASCADE
                ON UPDATE CASCADE
    FOREIGN KEY(Industry, Name)
        REFERENCES COMPANY
                ON DELETE CASCADE
                ON UPDATE CASCADE
    FOREIGN KEY(UserId)
        REFERENCES CUSTOMER
                ON DELETE CASCADE
                ON UPDATE CASCADE);
```

Portfolio

```
CREATE TABLE Portfolio_R1(
    Pid:            INTEGER ,
    Date:           DATE,
    Balance:        FLOAT,
    Since:          DATE,
    ManagerId:      INTEGER,
    PRIMARY KEY         (Pid),
    FOREIGN KEYS        (ManagerId),
        REFERENCES Manager
                ON DELETE SET NULL,
                ON UPDATE CASCADE);

CREATE TABLE Portfolio_R2(
    Pid:                INTEGER ,
    CustomerId:         INTEGER,
    PRIMARY KEY         (Pid, CustomerId),
    FOREIGN KEY         (CustomerId)
        REFERENCES Customer
                ON DELETE CASCADE,
                ON UPDATE CASCADE);
```