STOCK EXCHANGE

DATABASE PROJECT

Presented To

SHAHIDUL SHAKIB

Lecturer, Computer Science and Engineering, Khulna University of Engineering & Technology

Presented By

MAHIR AZMAIN HAQUE

Roll:2007118

Problem Statement

Most major corporations are publicly owned. Companies offer shares of ownership called stock. The price of stock changes by the minute depending on the value investors place on the stock. Corporations sell stock to raise money and grow the business.

A share is a single unit of ownership in a corporation. A shareholder, also known as a stockholder, is a person who owns shares of a certain company or organization and is thus a part-owner of the company. If there are a million shares and you own 1000, you own 0.001 of the company. As owners, the holders are entitled to elect the corporation's directors and vote on major issues. These votes typically take place at the corporation’s annual meeting, which shareholders are invited to attend.

In the fascinating world of the stock market, millions of transactions occur every second. This project aims to maintain a database system consisting of information about the current state of the stock market.

Every company has its name, the industry to which it belongs, the total number of shares
it holds and the current price of its shares. Every user (shareholder) has a name and unique ID. Also, different currencies and commodities are specified with their current prices.

Introduction

The term stock refers to the share of ownership of a company. In giant companies, one person can't generate all the capital and take all the risks. So the ownership is divided and sold as a stock.

A stock exchange is a marketplace where these stocks are sold and bought. By buying a stock, one becomes a part of the owner of a company.

The stock exchange must maintain a database to work properly. It should contain stock price, user info, company information and user-owned Equity.

Objectives

1. To create a database for the stock exchange.
2. To automate the data storage through it.
3. To provide accurate database services.
4. To manage and store various types of data related to the stock exchange, including information about listed companies, user accounts, stock prices, transactions, and historical data.
5. To know about ER Diagrams and Relational Schema.

Table Creation and Modification

```
create table Stock_Exchange(
exchange_id varchar(20) not null,
exchange_name varchar(40),
opening_time varchar(10) DEFAULT '10:00 AM',--USED DEFAULT
closing_time varchar(10));
-- adding Primary Key
alter table Stock_Exchange ADD PRIMARY KEY (exchange_id);

create table users(
user_id varchar(20) not null,
ac_no number(20),
balance number(20,2) CHECK (balance>0),--USED CHECK
extra varchar(12),
reg_date Date,
primary key(user_id));
-- ADDING NEW COLUMN
Alter table users ADD name varchar(20);
-- Drop A column
Alter table users drop COLUMN extra;

create table company_stock(
company_id varchar(20) NOT NULL,
company_name varchar(20),
PRIMARY KEY (company_id));

create table offer_to_sell(
exchange_id varchar(20) NOT NULL,
company_id varchar(20) NOT NULL,
Price varchar(20),
volume number(10),
PRIMARY KEY (exchange_id, company_id),
foreign key (company_id) REFERENCES company_stock (company_id) on DELETE CASCADE,
foreign key (exchange_id) REFERENCES stock_exchange (exchange_id)
on DELETE CASCADE);
-- MODIFY PRICE COLUMN
Alter table offer_to_sell MODIFY Price number(10,3);
```

```
create table buy_sell(
user_id varchar(20) NOT NULL,
exchange_id varchar(20) NOT NULL,
primary key (user_id,exchange_id),
foreign key (exchange_id) REFERENCES stock_exchange (exchange_id)
on DELETE CASCADE);
--ADDED FOREIGN KEY
Alter table buy_sell ADD foreign key (user_id) REFERENCES users (user_id) on DELETE
CASCADE;

create table yearly_revenue(
amount number(20,3),
user_id varchar(20) NOT NULL,
company_id varchar(20)NOT NULL,
exchange_id varchar(20) NOT NULL,
PRIMARY KEY (user_id, company_id,exchange_id),
foreign key (user_id) REFERENCES users (user_id) on DELETE CASCADE,
foreign key (company_id) REFERENCES company_stock (company_id)
on DELETE CASCADE,
foreign key (exchange_id) REFERENCES stock_exchange(exchange_id) on DELETE
CASCADE);

Create Table Owned_stock(
company_name varchar(20),
quantity number(20),
buy_price number(20),
user_id varchar(20) NOT NULL,
exchange_id varchar(20) NOT NULL,
company_id varchar(20) NOT NULL,
PRIMARY KEY (user_id, company_id,exchange_id),
foreign key (user_id) REFERENCES users (user_id) on DELETE CASCADE,
foreign key (company_id) REFERENCES company_stock (company_id)
on DELETE CASCADE,
foreign key (exchange_id) REFERENCES stock_exchange (exchange_id)
on DELETE CASCADE);
```

Entity-Relationship Diagram

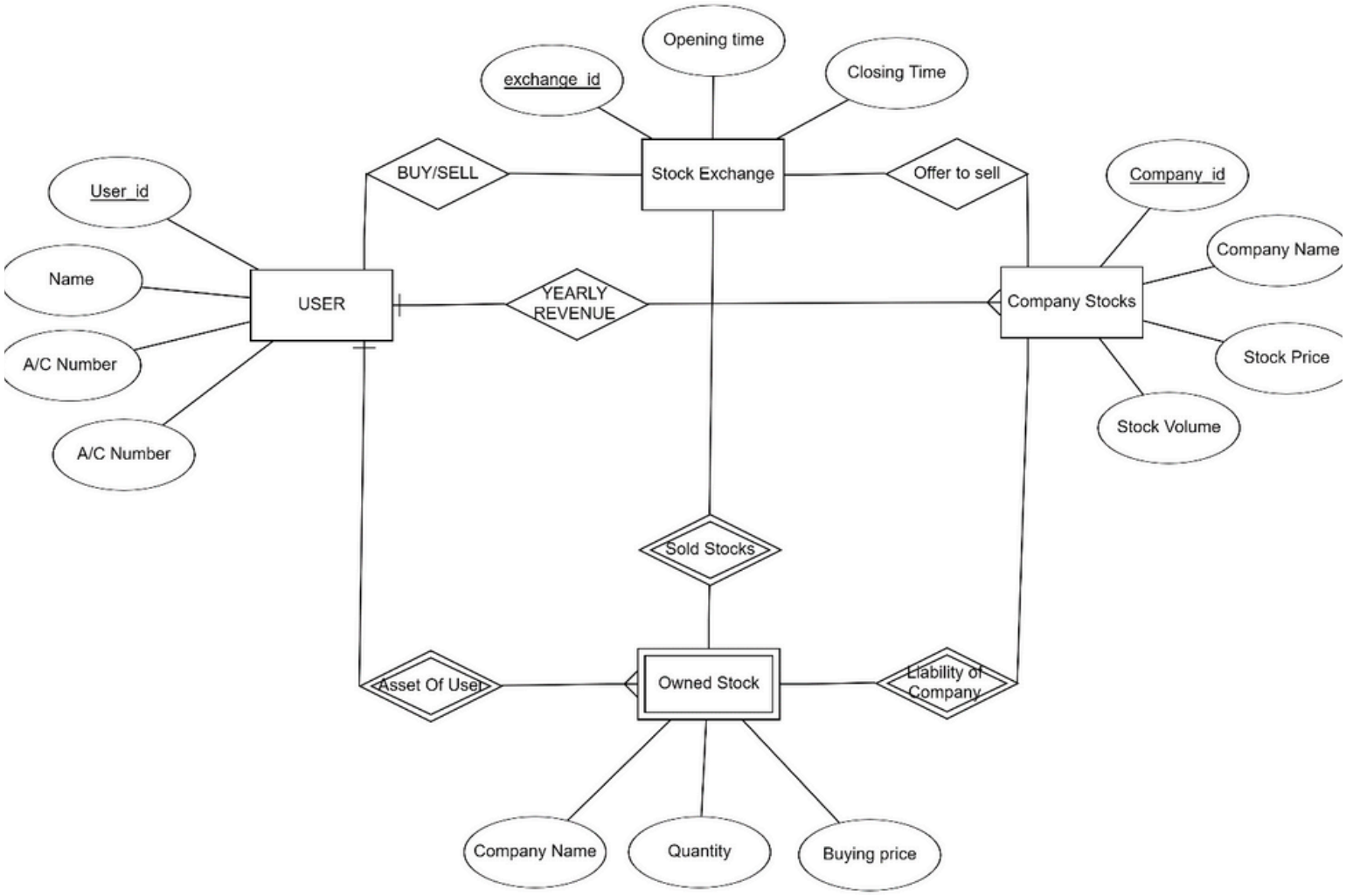

Fig: Entity Relationship Diagram of a Stock-Exchange Database

Relational Schema

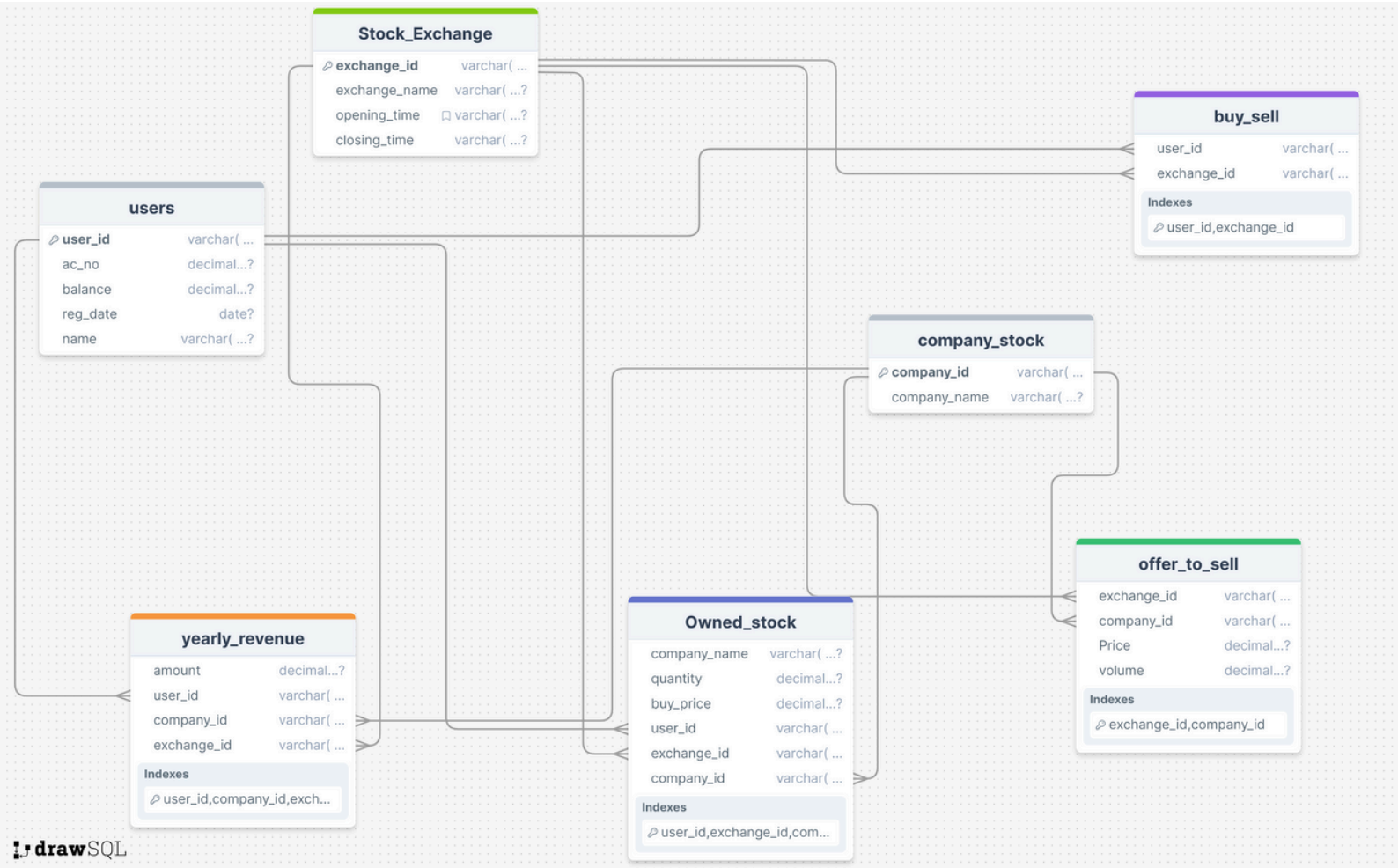

Fig:Relational Schema of a Stock-Exchange Database

Displaying table data using SELECT command

```
--DESCRIBE COMMAND
DESCRIBE Stock_Exchange;
DESCRIBE USERS;
DESCRIBE company_stock;
DESCRIBE owned_stock;
DESCRIBE buy_sell;
DESCRIBE offer_to_sell;
DESCRIBE yearly_revenue;

--SELECT * COMMAND
select * from Stock_Exchange;
select * from USERS;
select * from company_stock;
select * from owned_stock;
select * from buy_sell;
select * from offer_to_sell;
select * from yearly_revenue;
```

Updating the data in a table and Deleting row from a table

```
--UPDATING A DATA/ ROW with UPDATE
select * from Stock_Exchange where exchange_id='101';
UPDATE  Stock_Exchange
SET exchange_name='Dhaka Stock Exchange' where exchange_id='101';
select * from Stock_Exchange where exchange_id='101';

--DELETED ONE ENTRY/DATA
DELETE from company_stock
where company_id='C-100';
```

Union, Intersect, and Except

```
select company_name from company_stock
where company_id in (select company_id from offer_to_sell
where price>500)
UNION --USAGE OF UNION
select company_name from company_stock where company_id in (select company_id from
offer_to_sell
where volume<12000);
--Shows the stocks that have a price greater than 500 or a volume is less than 12000
```

```
select company_name from company_stock
where company_id in (select company_id from offer_to_sell
where price>500)
INTERSECT --USAGE OF INTERSECT
select company_name from company_stock where company_id in (select company_id from
offer_to_sell
where volume<12000);
--Shows the stocks who have a price greater than 500 AND volume is less than 12000
```

```
select company_name from company_stock where company_id in (select company_id from
offer_to_sell
where volume<12000)
MINUS
select company_name from company_stock
where company_id in (select company_id from offer_to_sell
where price>500);
--First Query Returns ford, tesla, Exim. Second Returns Ford and Tesla. So the Final Result is Exim.
```

```
select company_name from company_stock
where company_id in (select company_id from offer_to_sell
where price>500)
UNION
select company_name from company_stock where company_id in
(select company_id from offer_to_sell where volume<12000)
INTERSECT
select company_name from company_stock where company_id in
(select company_id from offer_to_sell where exchange_id='101');
--Here Union would be Executed At first as It is on the left of the Intersect
```

With clause

Show the user Name and Joining Year Of Each User. What is the count of users in the yearly_revenue table whose yearly revenue exceeds the average revenue?

```
select name, EXTRACT(YEAR from reg_date) AS JOINING_YEAR from users;
with high_revenue_users as (
    select user_id
    from yearly_revenue
where amount > (select avg(amount) from yearly_revenue)
)
select count(*) as high_revenue_users_count
from high_revenue_users;
```

Aggregate function

```
select count(*) from users;--users in the database

select count(*) from offer_to_sell where exchange_id = '101';--companies have listed their stocks for sale on the "Dhaka Stock Exchange"

select count(distinct user_id) from buy_sell where exchange_id in ('101', '103');--users have made transactions in both the "Dhaka Stock Exchange" and the "New York Stock Exchange"

select sum(volume) from offer_to_sell;--total quantity of stocks listed for sale across all stock exchanges

select (amount/12) from yearly_revenue;
```

Group by, Having and Nested subquery

```
select company_id, sum(volume)
from offer_to_sell group by company_id having sum(volume)>1000;--GROUP BY HAVING

select exchange_id ,sum(volume) from offer_to_sell group by exchange_id;
--Shows the Total Volume Each Stock Exchange Currently Selling

select company_id, avg(Price) from offer_to_sell group by company_id;
--Show On average Stock Price of a company

select company_id, avg(Price) from offer_to_sell group by company_id
having avg(price)>500;
--Shows which company has a price of more than $500

select company_name from company_stock where company_id in
(select DISTINCT company_id from offer_to_sell);--USAGE OF DISTINCT AND NESTED QUERY
```

Set Membership(AND, OR, NOT), some/all/exists/unique, String Operations

```
select name from users where balance between 100000 and 200000;--between
command
--showing who has a balance between 100000 and 200000

select user_id from buy_sell where exchange_id IN ('101', '103');
--This shows who has purchased history in stock_exchange 101 and 103

select * from owned_stock order by quantity desc;
--Shows all users based on their quantity of stock in descending order
```

```
select * from users u JOIN owned_stock o
USING (user_id);--Show full details of the users who have stocks

select * from users
where balance > all (select balance from stock_exchange);--Retrieve all users who have a balance greater than all balances in the Stock_Exchange table

select * from users
where balance < some (select balance from stock_exchange);--Retrieve all users who have a balance less than some balances in the Stock_Exchange table

select * from users
where exists (select 1 from stock_exchange where users.ac_no =
stock_exchange.exchange_id);--Check if there exists any user whose account number matches with any exchange ID.

select name from users where  name LIKE 'S%';
--SHOWS THE USER WHO HAS 'S' at First OF their Name
```

Join operations

```
select * from users u JOIN owned_stock o
USING (user_id);-- USING KEYWORD USED HERE
--QUERY show full details of the users who have stocks.

select u.name from
users, u JOIN owned_stock o
on u.user_id=o.user_id;
--THE QUERY SHOWS THE USER WHO HAVE STOCKS

select name, company_name from users natural JOIN OWNED_STOCK;
--USE OF NATURAL JOIN

select exchange_name,user_id,company_name from Stock_Exchange s cross JOIN owned_stock
o
where s.exchange_id=o.exchange_id;
--USED CROSS JOIN WITH where Condition.
```

```
select name,exchange_id from users u left outer JOIN buy_sell b on u.user_id=b.user_id;
--USED LEFT OUTER JOIN
--QUERY RETURNS ALL the user names and Exchange IDs for the users who have made the transaction
--also returns the name of users who have not made any transaction yet

select s.exchange_name,b.exchange_id from buy_sell b
right outer JOIN stock_exchange s on s.exchange_id=b.exchange_id;
--RIGHT OUTER JOIN
--QUERY RETURNS THE stock exchange who has already sold some stocks
--Also returns the exchange name and ID who has not sold any stock.

select s1.exchange_name from
stock_exchange s1 JOIN stock_exchange s2
on s1.opening_time=s2.closing_time;
--SELF JOIN
--RETURNS THE Stock Exchanges if Some stock Exchange's
--Closing time is another one's opening time
```

PL/SQL

```
SET SERVEROUTPUT ON
DECLARE
open stock_exchange.opening_time%type;
lim varchar(40);
BEGIN
lim:='Dhaka Stock Exchange';
select opening_time into open from Stock_Exchange
where exchange_name=lim;
DBMS_OUTPUT.PUT_LINE('OPENING TIME OF '||lim||' is '||open);
END;
/
--This Query Returns the opening time of DSE
--Usages of PL SQL block
```

```
DECLARE
CURSOR star_customer IS select name from users
where user_id in( select user_id from owned_stock group by user_id
having sum(quantity)>=1000);
s_cur star_customer%ROWTYPE;
c number;
BEGIN
OPEN star_customer;
c:=1;
LOOP
FETCH star_customer into s_cur;
EXIT when star_customer%NOTFOUND;
DBMS_OUTPUT.PUT_LINE ('Star Customer '||c||': '||s_cur.name);
c:=c+1;
END LOOP;
CLOSE star_customer;
END;
/
--If any customer has more than 1000 Stocks, he/she is a star customer
--This Query Returns the name of Star Customers
-- Usage of CURSOR


create or replace view Revenue_View AS
select u.name,user_id, sum(amount) AS TOTAL_REVENUE from users u natural JOIN
yearly_revenue y group by (user_id,name);
--This view shows the total revenue of each user;


select name, (TOTAL_REVENUE*.25) AS TAX_PAYABLE from Revenue_View where
TOTAL_REVENUE>5000;
--Shows the user name and Total Tax of Each user who has earned more than 5000 in
the previous year.
```

```
SET SERVEROUTPUT ON
BEGIN
dbms_output.put_line('Total REVENUE OF USER ABRAR IS: '|| show_revenue('U-24'));
dbms_output.put_line('Total REVENUE OF USER SADMAN IS: '|| show_revenue('U-20'));
END;
/
--Prints the total Revenue against user ID;



BEGIN
calculate_profit_loss(300,'U-24','C-3','103');
END;
/
--Returns the Profit/Loss if this stock is sold

--USE OF savepoint
SAVEPOINT s1;

insert into company_stock values ('C-1234','DEMO COMPANY' );

select * from company_stock;

ROLLBACK to s1;

select * from company_stock;

SET SERVEROUTPUT ON
DECLARE
v_num1 Number;
v_num2 Number;
v_sum Number;

BEGIN
V_num1 := &Number1;
V_num2 := &Number2;

V_sum := v_num1 + v_num2 ;

Dbms_Output.Put_Line ('The Sum of number is :' || v_sum);

END;
/
```

```
--THIS FUNCTION SHOWS THE YEARLY REVENUE OF AN USER
CREATE OR REPLACE FUNCTION show_revenue
(uid yearly_revenue.user_id%type)
RETURN yearly_revenue.amount%type IS
ret_value yearly_revenue.amount%type;
BEGIN
select sum(amount) into ret_value
from yearly_revenue where yearly_revenue.user_id=uid;
RETURN ret_value;
END show_revenue;
/

--This Procedure Returns The Amount Of Profit OR Loss Of a User against His stock
SET SERVEROUTPUT ON
create or REPLACE PROCEDURE calculate_profit_loss
(
quan owned_stock.quantity%type,
uid owned_stock.user_id%type,
cid owned_stock.company_id%type,
eid owned_stock.exchange_id%type)IS
cur_price offer_to_sell.Price%type;
b_price owned_stock.buy_price%type;
q1 owned_stock.quantity%type;
BEGIN
select price into cur_price from offer_to_sell
where company_id=cid and exchange_id=eid;

select buy_price,quantity into b_price,q1 from owned_stock
where user_id=uid and company_id=cid and exchange_id=eid;

if b_price>cur_price then
DBMS_OUTPUT.PUT('LOSS: ');
DBMS_OUTPUT.PUT_LINE(quan*b_price-quan*cur_price);
elsif b_price<cur_price then
DBMS_OUTPUT.PUT_LINE(b_price);
DBMS_OUTPUT.PUT_LINE(cur_price);
DBMS_OUTPUT.PUT('PROFIT :');
DBMS_OUTPUT.PUT_LINE(quan*cur_price-quan*b_price);
else
DBMS_OUTPUT.PUT_LINE('No profit or Loss');
end if;
END calculate_profit_loss;
/
```

```
SET SERVEROUTPUT ON
create or replace procedure add_yearly_revenue(
am1 yearly_revenue.amount%type,
uid users.user_id%type,
cid company_stock.company_id%type,
eid stock_exchange.exchange_id%type) IS
--<ADDED LOCAL VARIABLE TO PL/SQL PROCEDURE>
c number;
BEGIN
--<THIS IS FOR CHECKING IF THE USER HAS STOCK FROM THIS COMPANY>--
select count(1) into c from owned_stock where cid=company_id and
eid=owned_stock.exchange_id
and uid=owned_stock.user_id;
if (c=1) then
INSERT into yearly_revenue(amount,user_id,company_id,exchange_id)
values(am1,uid,cid,eid);
else
RAISE_APPLICATION_ERROR(-122,'User does not have stock');
end if;
END add_yearly_revenue;
/

--THIS TRIGGER WILL AUTOMATICALLY ADD DATA TO buy_sell table when someone
buys/sells stocks;
CREATE OR REPLACE TRIGGER check_buy_sell BEFORE INSERT OR UPDATE ON
owned_stock
FOR EACH ROW
DECLARE
c number;
BEGIN
select count(1) into c from buy_sell
where :new.user_id=buy_sell.user_id and :new.exchange_id=buy_sell.exchange_id;
if(c=0) then
INSERT INTO buy_sell(user_id,exchange_id) values (:new.user_id,:new.exchange_id);
end if;
END;
/
```

Discussion

The ER diagram shows the brief idea of the project. Here are entities users, Stock Exchange, company _stock and Owned_stock.

Users entity has the attribute user_id, Name, A/C number and A/C balance.

Stock Exchange has attributes exchange_id, opening time and closing time.

Company_stock entity has attribute company_id,company_name, stock price, and stock volume.

Lastly, Owned_stock has a company name, quantity and buying price attribute. This is a weak entity set.

Here users and the Stock Exchange are in a many-to-many relation. Users can buy stock from more than one Stock Exchange market place and each marketplace may have more than one user. Again Stock Exchange and the company are in many-to-many relation. Each company may sell stocks in different stock exchanges and each stock exchange may have more than one company.

Users and Company are in many to many relation. This is because one user gets yearly revenue from more than one company and each company may have more than one stock holder.

The Owned_stock entity has a relation with the user, stock exchange and company. So its primary key would be user_id,exchange_id and company_id. Users see his/her owned stock from this. The Stock Exchange knows the sold amount and the company can know the liability amount from this table.

Conclusion

In conclusion, the stock exchange management system is a vital component of the global financial system. It has evolved significantly over time and will continue to adapt to new technologies and market dynamics. Regulatory oversight remains crucial to maintaining trust and stability in these markets. As the financial industry continues to evolve, the stock exchange management system will play a central role in connecting investors with opportunities and shaping the global economy.

From this project, we would be able to design a database that is suitable for the Stock Exchange. It will also lessen the data redundancy a lot. So, these were the primary goals of this project which would be achieved after completing the project.