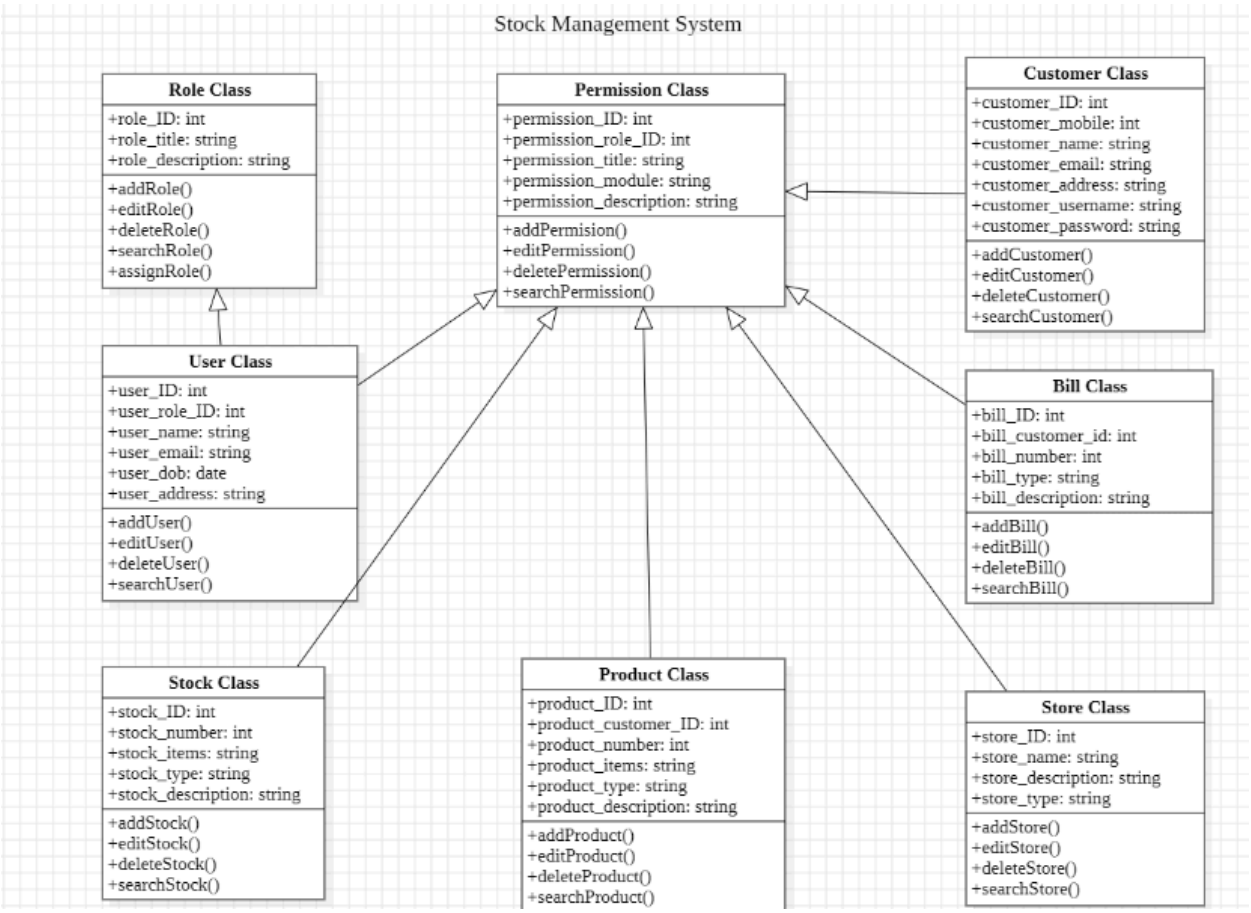
Stock Market Database Management System

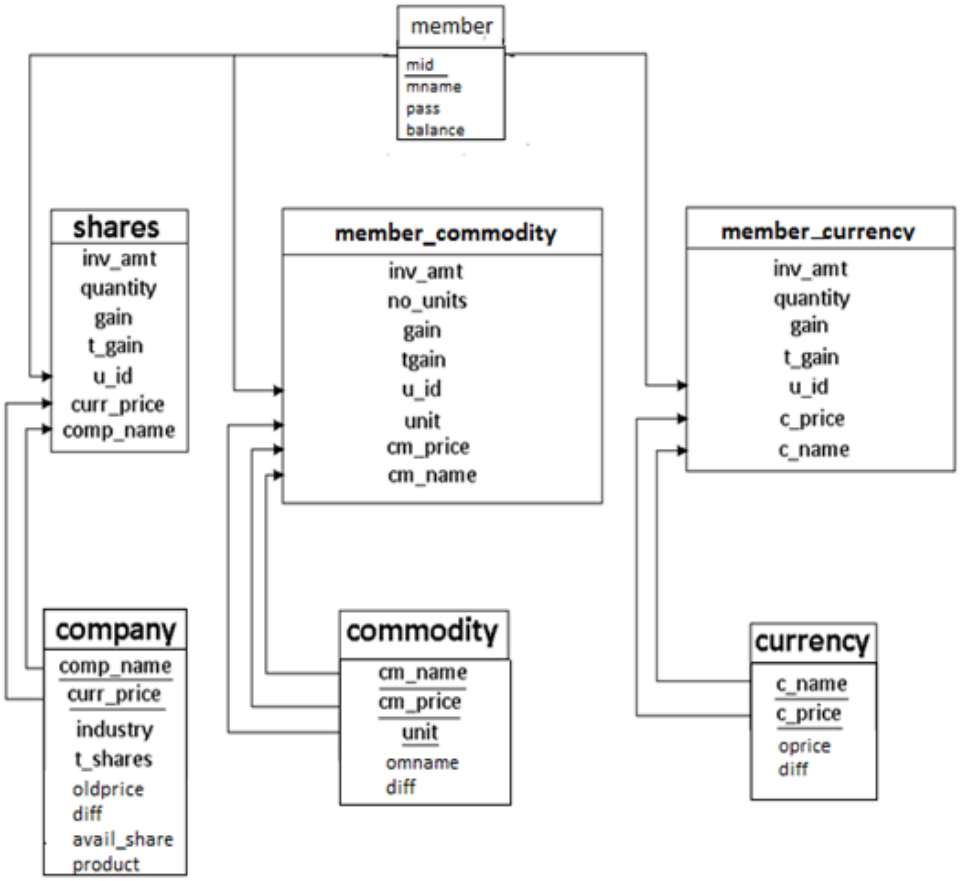
M Bhuvan Anand : PES1UG22CS311
M Mohan Gowda : PES1UG22CS313

ER diagram :



Relationship schema :

RELATIONAL SCHEMA



Queries:

Triggers :

1.

```
DELIMITER ;;
CREATE TRIGGER `after_company_index_insert`
AFTER INSERT ON `company_indexes`
FOR EACH ROW
BEGIN
    INSERT INTO company_index_log (cuid, action)
    VALUES (NEW.cuid, 'INSERT');
END ;;
DELIMITER ;
```

CREATE TRIGGER: Defines a new trigger named `after_company_index_insert`.
AFTER INSERT ON `company_indexes`: Specifies that the trigger should execute after a row is inserted into the `company_indexes` table.
FOR EACH ROW: Indicates that the trigger applies to each row that is inserted.
BEGIN ... END: The block of SQL code that will be executed when the trigger fires.
INSERT INTO `company_index_log`: Inserts a new row into the `company_index_log` table with the `cuid` from the newly inserted row in `company_indexes` and the action `'INSERT'`.

This code ensures that every time a new record is added to the `company_indexes` table, a corresponding entry is made in the `company_index_log` table, tracking the insert action.

2.

```
DELIMITER ;;
CREATE TRIGGER `after_company_index_update`
AFTER UPDATE ON `company_indexes`
FOR EACH ROW
BEGIN
    INSERT INTO company_index_log (cuid, action)
    VALUES (OLD.cuid, 'UPDATE');
END ;;
DELIMITER ;
```

A **trigger for updates** on the `company_indexes` table, which was not present in the previously uploaded code.

3.

```
CREATE TABLE `fund_transaction_log` (
    `log_id` int NOT NULL AUTO_INCREMENT,
    `tuid` varchar(22) DEFAULT NULL,
    `uuid` varchar(22) DEFAULT NULL,
    `action` varchar(50) DEFAULT NULL,
    `action_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`log_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

The creation of a **new table** `fund_transaction_log` to store logs for fund transactions.

4.

```
DELIMITER ;;
CREATE TRIGGER `after_fund_transaction_insert`
AFTER INSERT ON `fund_transactions`
FOR EACH ROW
BEGIN
    INSERT INTO fund_transaction_log (tuid, uuid, action)
    VALUES (NEW.tuid, NEW.uuid, 'INSERT');
END ;;
DELIMITER ;
```

A **trigger for insertions** on the `fund_transactions` table, which logs the action into `fund_transaction_log`.

Procedures:-

1.

```
DELIMITER ;;

CREATE PROCEDURE `AddCompanyIndex`(
    IN p_cuid VARCHAR(15),
    IN p_sl_no SMALLINT,
    IN p_name VARCHAR(52),
    IN p_symbol VARCHAR(10),
    IN p_market VARCHAR(2),
    IN p_no_equity VARCHAR(15),
    IN p_price_yesterday DECIMAL(6,2),
    IN p_price_today DECIMAL(6,2),
    IN p_change_percentage DECIMAL(3,2)
)
BEGIN
    INSERT INTO COMPANY_INDEXES (
        cuid,
        sl_no,
        name,
        symbol,
        market,
        no_equity,
        price_yesterday,
        price_today,
        change_percentage
    ) VALUES (
        p_cuid,
        p_sl_no,
        p_name,
        p_symbol,
        p_market,
        p_no_equity,
        p_price_yesterday,
        p_price_today,
        p_change_percentage
    );
END ;;

DELIMITER ;
```

This SQL stored procedure allows you to insert new records into the `COMPANY_INDEXES` table using the specified parameters, ensuring that the process is standardized and reusable.

2.

```
DELIMITER ;;

CREATE PROCEDURE `AddFundTransaction`(
    IN p_tuid VARCHAR(22),
    IN p_uuid VARCHAR(22),
    IN p_type VARCHAR(8),
    IN p_amount DECIMAL(10,2),
    IN p_date VARCHAR(10)
)
BEGIN
    INSERT INTO fund_transactions (
        tuid,
        uuid,
        type,
        amount,
        date
    ) VALUES (
        p_tuid,
        p_uuid,
        p_type,
        p_amount,
        p_date
    );
END ;;

DELIMITER ;
```

This procedure adds a new fund transaction to the `fund_transactions` table. It takes several parameters for the transaction details and inserts them into the table

3.

```
DELIMITER ;;

CREATE PROCEDURE `UpdateCompanyIndex`(
    IN p_cuid VARCHAR(15),
    IN p_sl_no SMALLINT,
    IN p_name VARCHAR(52),
    IN p_symbol VARCHAR(10),
    IN p_market VARCHAR(2),
    IN p_no_equity VARCHAR(15),
    IN p_price_yesterday DECIMAL(6,2),
    IN p_price_today DECIMAL(6,2),
    IN p_change_percentage DECIMAL(3,2)
)
BEGIN
    UPDATE company_indexes
    SET
        sl_no = p_sl_no,
        name = p_name,
        symbol = p_symbol,
        market = p_market,
        no_equity = p_no_equity,
        price_yesterday = p_price_yesterday,
        price_today = p_price_today,
        change_percentage = p_change_percentage
    WHERE cuid = p_cuid;
END ;;

DELIMITER ;
```

This procedure updates an existing company index in the `company_indexes` table. It uses parameters to specify which fields to update and the new values for those fields.

Functions:-

1.

```
DELIMITER ;;

CREATE FUNCTION `CalculateTotalValue`(p_uuid VARCHAR(22))
RETURNS DECIMAL(10,2)
BEGIN
    DECLARE total_value DECIMAL(10,2);

    SELECT SUM(t.qty * t.price) INTO total_value
    FROM transactions t
    WHERE t.uuid = p_uuid
    AND t.order_type = 'buy';

    RETURN total_value;
END ;;

DELIMITER ;
```

Function to calculate the total value of a User's portfolio

3.

```
DELIMITER ;;

CREATE FUNCTION `CalculateCompanyIndexChange`(p_cuid VARCHAR(15))
RETURNS DECIMAL(3,2)
BEGIN
    DECLARE price_yesterday DECIMAL(6,2);
    DECLARE price_today DECIMAL(6,2);
    DECLARE change_percentage DECIMAL(3,2);

    SELECT price_yesterday, price_today
    INTO price_yesterday, price_today
    FROM company_indexes
    WHERE cuid = p_cuid;

    SET change_percentage = ((price_today - price_yesterday)
    / price_yesterday) * 100;

    RETURN change_percentage;
END ;;

DELIMITER ;
```

Function To Calculate Change Percentage Of A Company Index

nested query:-

```
DELIMITER ;;

CREATE PROCEDURE `GetUserTotalFundTransactions`(
    IN p_uuid VARCHAR(22)
)
BEGIN
    SELECT uuid, SUM(amount) AS total_amount
    FROM fund_transactions
    WHERE uuid = p_uuid
    GROUP BY uuid;
END ;;

DELIMITER ;
```

```
const getUserTotalFundTransactions = async (uuid) => {
    const [result] = await db.promise().query(
        'CALL GetUserTotalFundTransactions(?)',
        [uuid]
    );
    console.log('Total fund transactions for user:', result[0]);
};

// Example usage
getUserTotalFundTransactions('iDwf2cfe7ZjgXBTTB3cgCA');
```

aggregated query:-

```
DELIMITER ;;

CREATE PROCEDURE `GetTotalPortfolioValueForAllUsers`()
BEGIN
    SELECT uuid, SUM(t.qty * t.price) AS total_value
    FROM transactions t
    WHERE t.order_type = 'buy'
    GROUP BY uuid;
END ;;

DELIMITER ;
```

insert query:-

```
DELIMITER ;;

CREATE PROCEDURE `AddUser`(
    IN p_uuid VARCHAR(22),
    IN p_first_name VARCHAR(7),
    IN p_last_name VARCHAR(6),
    IN p_email VARCHAR(24),
    IN p_password_hash VARCHAR(60),
    IN p_dp_uri VARCHAR(26)
)
BEGIN
    INSERT INTO users (
        uuid, first_name, last_name, email,
        password_hash, dp_uri
    )
    VALUES (
        p_uuid, p_first_name, p_last_name, p_email,
        p_password_hash,
        p_dp_uri
    );
END ;;

DELIMITER ;
```

```
const addUser = async (
    uuid,
    first_name,
    last_name,
    email,
    password_hash,
    dp_uri
) => {
    const [result] = await db.promise().query(
        'CALL AddUser(?, ?, ?, ?, ?, ?)',
        [
            uuid,
            first_name,
            last_name,
            email,
            password_hash,
            dp_uri
        ]
    );
    console.log('User added:', result);
};

// Example usage
addUser(
    'newUuid',
    'John',
    'Doe',
    'john.doe@example.com',
    'hashed_password',
    'dp_uri_now_not_implemented'
);
```

update query:-

```
DELIMITER ;;

CREATE PROCEDURE `UpdateUser`(
    IN p_uuid VARCHAR(22),
    IN p_first_name VARCHAR(7),
    IN p_last_name VARCHAR(6),
    IN p_email VARCHAR(24),
    IN p_password_hash VARCHAR(60),
    IN p_dp_uri VARCHAR(26)
)
BEGIN
    UPDATE users
    SET
        first_name = p_first_name,
        last_name = p_last_name,
        email = p_email,
        password_hash = p_password_hash,
        dp_uri = p_dp_uri
    WHERE uuid = p_uuid;
END ;;

DELIMITER ;
```

```
const updateUser = async (
    uuid,
    first_name,
    last_name,
    email,
    password_hash,
    dp_uri
) => {
    const [result] = await db.promise().query(
        'CALL UpdateUser(?, ?, ?, ?, ?, ?)',
        [
            uuid,
            first_name,
            last_name,
            email,
            password_hash,
            dp_uri
        ]
    );
    console.log('User updated:', result);
};

// Example usage
updateUser(
    'iDwf2cfe7ZjgXBTTB3cgCA',
    'Rabeeh',
    'Ta',
    'rabeeh.ta@example.com',
    'new_hashed_password',
    'dp_uri_now_not_implemented'
);
```

delete query:-

```
DELIMITER ;;

CREATE PROCEDURE `DeleteUser`(
    IN p_uuid VARCHAR(22)
)
BEGIN
    DELETE FROM users
    WHERE uuid = p_uuid;
END ;;

DELIMITER ;
```

Users:-

uuid	first_name	last_name	email	password_hash	dp_uri
iDwf2cfe7ZjgXBTTB3cgCA	rabeeh	ta	rabeeh@gmail.com	$2b$10$NDFpwsTgqxD0LQJFNMha..D.0t4FW4YJALQdWG7Pnd9JLAJrSn4Ue	dp_uri now not implemented
7Wrp2ULtX1QSSg7mpCnDe9	sinu	pedli	sinu@gmail.com	$2b$10$OUF/lvpmcBOwYwrutS20GOdK6mwSdOvMHAu8yb440ffdX3.pyT7H6	dp_uri now not implemented
rG2ftwYrrysUE8ed2XUdUJ	Muzzu	koli	muzzu@gmail.com	$2b$10$oQhTFORh3fgkt2kBql6b5eAQMgEZNQAmiw8RKJbcMfVPgsmUy/YxW	dp_uri now not implemented
1MMymeMkpwdsRJhwwbAR43	muneebu	beeb	muni@gmail.com	$2b$10$5iEO6hDGXNqTDQYC3I1MceCXMK0JFf0TUPO/a4y3ykHh8XLcKNeGa	dp_uri now not implemented
d4DeUdbXpxaqD91bmMLUht	niyaz	mohd	niyaz@gmail.com	$2b$10$2RQxp2eOKbMbdPEgeXaGEeHb22hWJQMwB42iRkXjVDlHmpY06YlUa	dp_uri now not implemented
qbYLANEQAKtTyK9L6Vi5ex	muneeb	mujeeb	muneeb@gmail.com	$2b$10$YNTYI86k32Xgad6NK3TQuusSGAD9rhxX1WAKuf04waZmGwWCUr/d.	dp_uri now not implemented
a8Az7ZPrGYuycJWWm3tGij	raziq	mk	raziq@gmail.com	$2b$10$UN7ebwBbic.FyRr3wj5s0O4THqW3p0VJHPeofzbqVL1Mh5zfyrcd2	dp_uri now not implemented
qH5ZzVrbajCVABqxfFkQhf	sulthan	Paris	sulthan@gmail.com	$2b$10$.jsMN8GBJyLL1gfIZOa9M.YBM4E2EIze2obzvGgnP9zEFsBJpxd0e	dp_uri now not implemented
o2dxHJbHeMD4EpB4wnCutQ	faiz	jaleel	faiz@gmail.com	$2b$10$FmDeoLmdIh0EuybyN5QYhOhATPhQF5H.J/D98dv7pJjMH4MeQGrau	dp_uri now not implemented

join query:-

```
DELIMITER ;;

CREATE PROCEDURE `GetUserTransactionsWithCompanyDetails`(
    IN p_uuid VARCHAR(22)
)
BEGIN
    SELECT t.tuid, t.uuid, t.symbol, t.order_type, t.date, t.qty, t.price,
           c.name, c.market
    FROM transactions t
    JOIN company_indexes c ON t.symbol = c.symbol
    WHERE t.uuid = p_uuid;
END ;;

DELIMITER ;
```

```
const getUserTransactionsWithCompanyDetails = async (uuid) => {
    const [result] = await db.promise().query(
        'CALL GetUserTransactionsWithCompanyDetails(?)',
        [uuid]
    );
    console.log('User transactions with company details:', result);
};

// Example usage
getUserTransactionsWithCompanyDetails(
    'iDwf2cfe7ZjgXBTTB3cgCA'
);
```