HW4 - Fast Food Chain Website

Group 24

1.Database Design

Main tables:

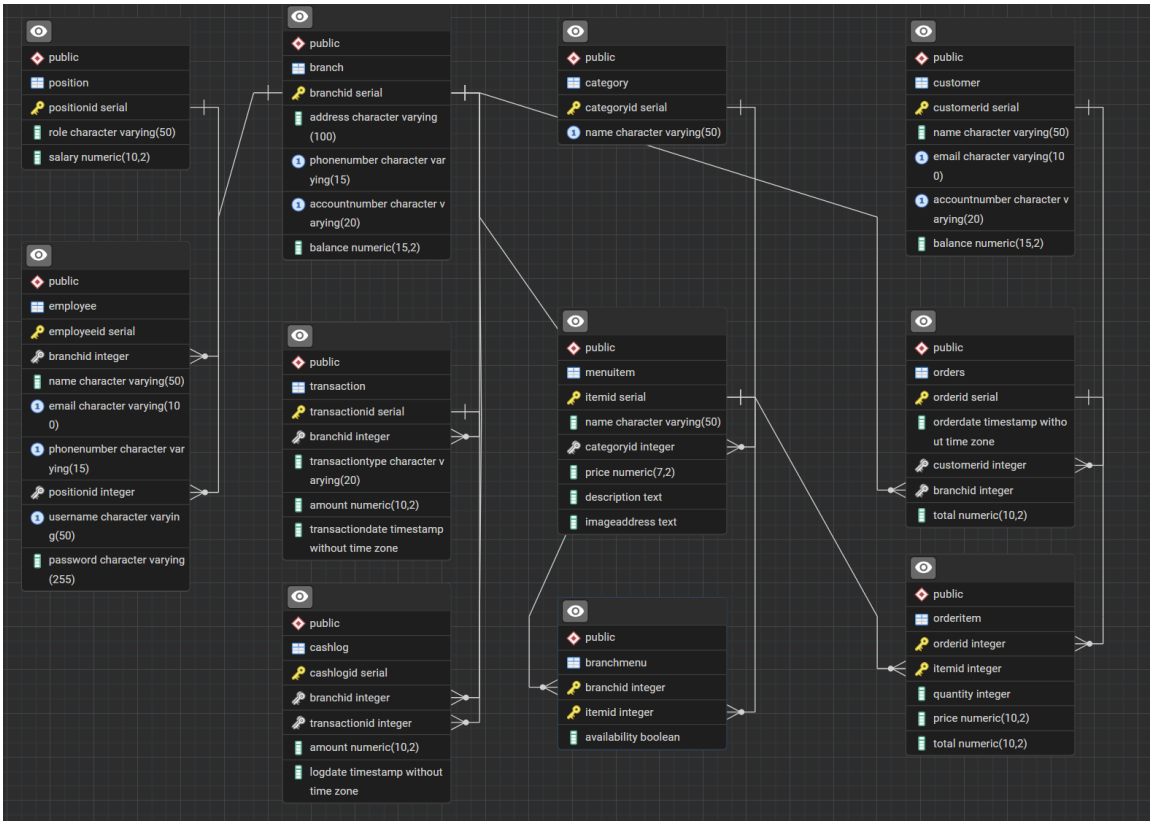
- Branch
- Customer
- Category
- MenuItem
- BranchMenu
- Orders
- OrderItem
- Transaction
- Cashlog
- Position
- Employee

All tables satisfy 3NF/BCNF.

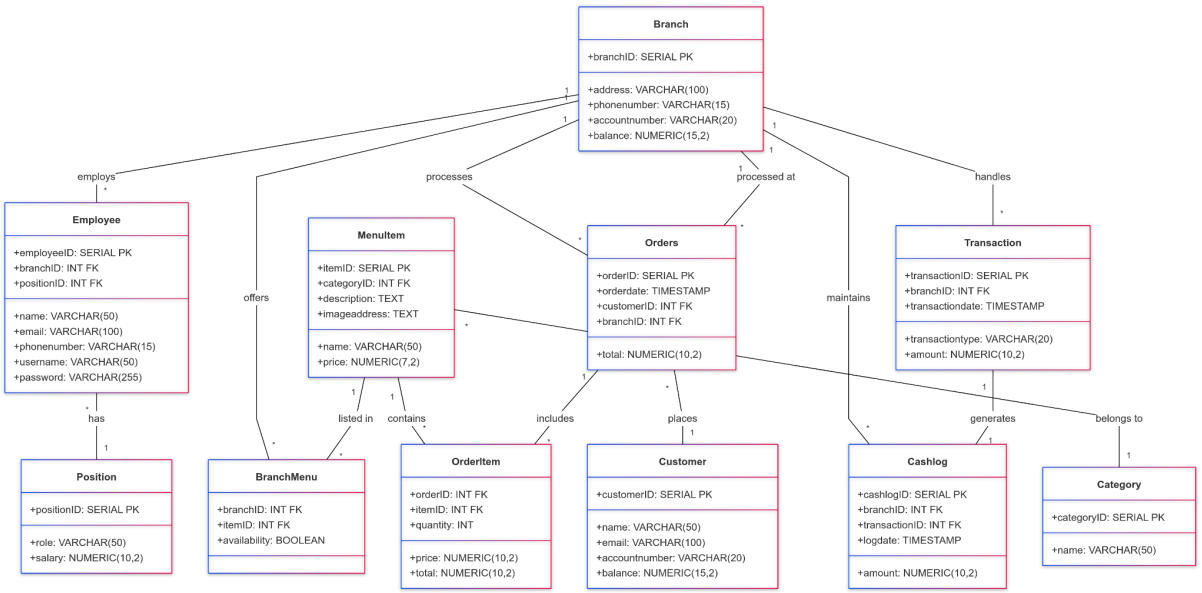
OrderItem table has a *composite key* form by **orderID** (from Order table) and **itemID** (from MenuItem table).

Transaction table with **transactionID** as *primary key*.

ERD diagram



UML diagram



2.Relationship between tables

One to Many: Branch and Order, MenuItem and Category, Customer and Order, Order and OrderItem, Branch and Transaction, Branch and Employee, Position and Employee.

One relation violates one to many: Transaction and Cashlog **one to one**, separation between Transaction and Cashlog in order to control the cash flow into each branch, which is different from payment made by card.

3. Queries for Report and Transaction

Sample queries we use to get data for the report and transaction:

Get available items in a specific branch:

```
SELECT mi.itemID, mi.name, mi.price, mi.description, mi.imageaddress
FROM MenuItem mi
JOIN BranchMenu bm ON mi.itemID = bm.itemID
WHERE bm.branchID = $1 AND bm.availability = TRUE;
```

Retrieve all orders sort by date:

```
SELECT o.orderID, COALESCE(c.name, 'Guest') AS customerName,
STRING_AGG(mi.name, ', ') AS itemNames,SUM(oi.price * oi.quantity)
AS orderTotal, o.orderdate AS orderTime, b.address AS branchName
FROM Orders o
LEFT JOIN Customer c ON o.customerID = c.customerID
INNER JOIN OrderItem oi ON o.orderID = oi.orderID
INNER JOIN MenuItem mi ON oi.itemID = mi.itemID
INNER JOIN Branch b ON o.branchID = b.branchID
GROUP BY o.orderID, c.name, o.orderdate, b.address
ORDER BY o.orderdate;
```

Transaction Logical:

Check if customer exist or not:

- If yes, customers can proceed to order and transaction.
- Else, ask customers to register.

Order and Transaction:

- Create Order in Order table
- Insert into OrderItems
- Update balance of customer and restaurant
- Insert new record in transaction table

Queries for transaction:

Register customer

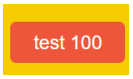
```
DO $$
DECLARE
    customerId INT;

BEGIN
    -- Check if account exists
    SELECT customerid
    INTO customerId
    FROM Customer
    WHERE accountnumber = 'accountNumberPlaceholder';

    -- If account does not exist, register a new customer and set
initial balance
    IF customerId IS NULL THEN
        INSERT INTO customer (name, email, accountnumber)
        VALUES ('namePlaceholder', 'emailPlaceholder',
'accountNumberPlaceholder')
        RETURNING customerid INTO customerId;

        UPDATE customer
        SET balance = 'initialBalancePlaceholder'
        WHERE accountnumber = 'accountNumberPlaceholder';
    END IF;
END $$;

COMMIT;
```

Transaction Handling

```
BEGIN;

-- Check if account exists and retrieve balance
DO $$
DECLARE
    customerId INT;
    customerBalance NUMERIC;
BEGIN
    SELECT customerid, balance
    INTO customerId, customerBalance
    FROM Customer
    WHERE accountnumber = 'accountNumberPlaceholder';

    IF customerId IS NULL THEN
        RAISE EXCEPTION 'Customer does not exist.';
```

```
    ELSIF customerBalance < 'totalAmountPlaceholder' THEN
        RAISE EXCEPTION 'Insufficient balance.';
    END IF;
END $$;

-- Create order and insert order items
DO $$
DECLARE
    orderId INT;
BEGIN
    INSERT INTO orders (customerid, branchid, total)
    VALUES (
        (SELECT customerid FROM Customer WHERE accountnumber =
'accountNumberPlaceholder'),
        'branchIDPlaceholder',
        'totalAmountPlaceholder'
    )
    RETURNING orderid INTO orderId;

    -- Insert order items
    INSERT INTO orderitem (orderid, itemid, quantity, price)
    SELECT
        orderId,
        item.itemid,
        item.quantity,
        item.price
    FROM UNNEST('cartItemsPlaceholder'::jsonb) AS item(itemid,
quantity, price);
END $$;

-- Update balances
UPDATE customer
SET balance = balance - 'totalAmountPlaceholder'
WHERE accountnumber = 'accountNumberPlaceholder';

UPDATE branch
SET balance = balance + 'totalAmountPlaceholder'
WHERE branchid = 'branchIDPlaceholder';

-- Record transaction
INSERT INTO transaction (branchid, transactiontype, amount)
VALUES ('branchIDPlaceholder', 'Card', 'totalAmountPlaceholder');

COMMIT;
```

4.Testing

We create a test 100 button 

, when you click the button there will be 100 random transactions going into the restaurant.

5. Video

The video including how to setup and test our project:
https://drive.google.com/file/d/1Ig1inPFYXyiyljyptyOG7tKNHo8AijAs/view?usp=sharing
HW4-dbs24.mp4