NATIONAL UNIVERSITY OF LESOTHO

CS4430
PRINCIPLES OF DISTRIBUTED DATABASE SYSTEMS

CLUSTERCORE

201902624	*L LETSIE*
201902131	*M SEEQELA*
201902703	*F MOTSU*
202004214	*N TLALI*

MMB SYSTEM IMPLEMENTATION

MALOTI MOUNTAIN BREWERY

DISTRIBUTED DATABASE MANAGEMENT SYSTEM

PURPOSE:

The purpose of brewery distribution is to get the beer produced by the brewery to consumers in various locations. Distribution can involve selling beer to bars, restaurants, and retail stores or delivering beer directly to consumers through online orders or a taproom. Some breweries may have their own distribution network, while others may use third-party distributors to help get their beer to different markets. The distribution process involves managing inventory, shipping and logistics, as well as complying with legal regulations and obtaining necessary licenses and permits. By distributing their beer, breweries can expand their customer base, increase sales, and build their brand awareness in new markets.

SCOPE AND SPECIAL REQUIREMENTS:

A brewery DDBMS (Distributed Database Management System) is a complex system that handles the data management needs of a brewery. The system needs to handle a large amount of data from various sources, including inventory management, production scheduling, quality control, and sales data. Here are some of the scope and special requirements for a brewery DDBMS:

Data integration: A brewery DDBMS should be able to integrate data from various sources, including production, inventory, and sales data. This requires the system to have a flexible data model that can handle different data types and formats.

Distributed data storage: A brewery DDBMS should be able to store data in a distributed manner, allowing multiple users to access data from different locations. This requires the system to have a distributed database architecture that can handle high volumes of data and provide fast access to data.

Real-time data processing: A brewery DDBMS should be able to process data in real-time, allowing brewery operators to monitor production, inventory, and sales data in real-time. This requires the system to have a fast data processing engine that can handle real-time data streams.

Analytics and reporting: A brewery DDBMS should be able to provide analytics and reporting features, allowing brewery operators to analyze data and generate reports. This requires the system to have advanced analytics capabilities, including data visualization and predictive analytics.

Security and data privacy: A brewery DDBMS should be able to ensure the security and privacy of data. This requires the system to have advanced security features, including data encryption, access control, and auditing.

Scalability: A brewery DDBMS should be able to scale up or down depending on the needs of the brewery. This requires the system to have a scalable architecture that can handle a growing volume of data and users.

Integration with other systems: A brewery DDBMS should be able to integrate with other systems, including accounting software, ERP systems, and production planning software. This requires the system to have an open architecture that can easily integrate with other systems.

TERMINOLOGY:

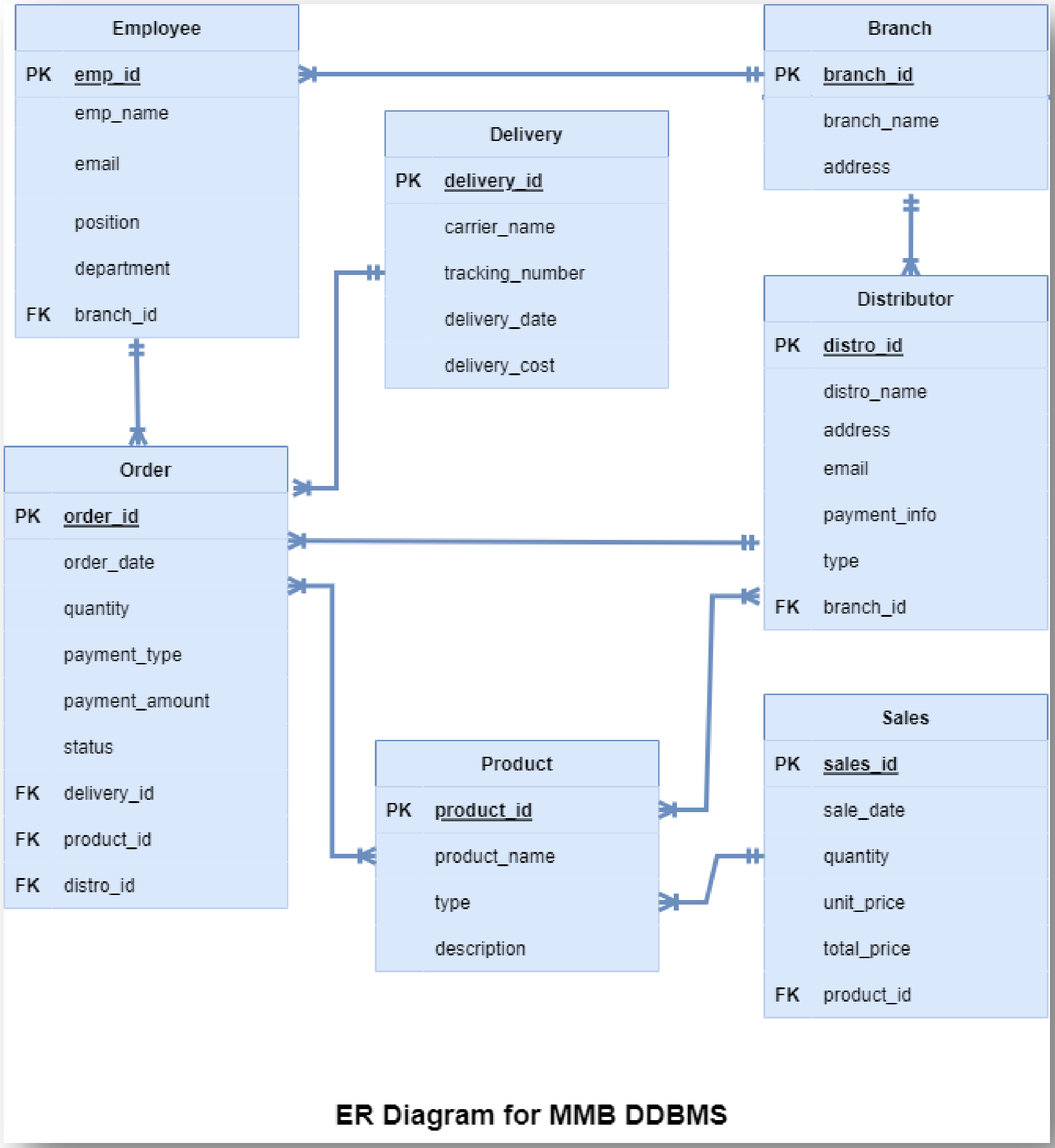
DDBMS: A centralized software system that managed a distributed database in a manner as if it were all stored in a single location.
BREWERY: The factory where beer is made.
ERP SYSTEM: A type of software system that helps organization automates and manages core business processes for optimal performance.

DATABASE DESCRIPTION:

A brewery distribution database is a system that stores and manages information related to the distribution of beer and other beverages produced by a brewery. The database typically includes information about the brewery's production facilities, the products they produce, their distribution network, and their customers. The brewery distribution database is designed to help the brewery manage their inventory, sales, and distribution processes more efficiently and effectively, and to provide insights into their customers and the market.

ENTITIES AND THEIR ATTRIBUTES:

ER diagram



POSGRES STATEMENTS:

CREATE:

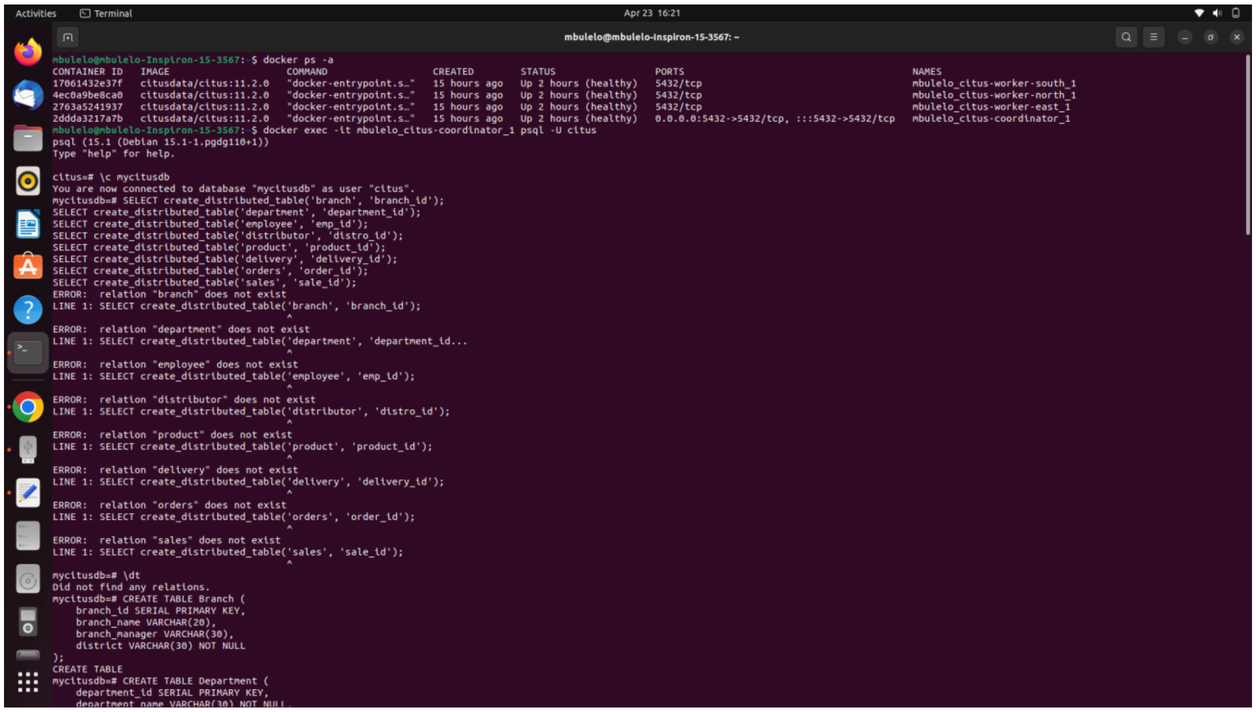
```
CREATE TABLE Branch (
   branch_id SERIAL PRIMARY KEY,
   branch_name VARCHAR(20),
   branch_manager VARCHAR(30),
   district VARCHAR(30) NOT NULL
```

```
);

CREATE TABLE Department (
    department_id SERIAL PRIMARY KEY,
    department_name VARCHAR(30) NOT NULL,
    H0D VARCHAR(50),
    branch_id INT NOT NULL,
    FOREIGN KEY (branch_id) REFERENCES Branch(branch_id)
);

CREATE TABLE Employee (
    emp_id SERIAL PRIMARY KEY,
    emp_firstname VARCHAR(20),
    emp_lastname VARCHAR(20),
    email VARCHAR(50),
    mobile_no INT UNIQUE,
    emo_position CHAR,
    department_id INT NOT NULL,
    FOREIGN KEY (department_id) REFERENCES Department(department_id)
);

CREATE TABLE Distributor (
    distro_id SERIAL PRIMARY KEY,
    distro_name VARCHAR(50) NOT NULL,
    address VARCHAR(50) NOT NULL,
    email VARCHAR(50),
    mobile_no INT NOT NULL UNIQUE,
    distro_type VARCHAR(50) NOT NULL
);

CREATE TABLE Product (
    product_id SERIAL PRIMARY KEY,
    product_name VARCHAR(50) NOT NULL,
    product_type VARCHAR(50) NOT NULL,
    description VARCHAR(50)
);

CREATE TABLE Delivery (
    delivery_id SERIAL PRIMARY KEY,
    carrier_name VARCHAR(50) NOT NULL,
    delivery_date DATE,
    delivery_cost INT
);

CREATE TABLE Orders (
    order_id SERIAL PRIMARY KEY,
    order_date DATE DEFAULT CURRENT_DATE,
    quantity INT,
    payment_type VARCHAR(50),
    payment_amount INT,
    delivery_id INT,
    product_id INT,
    distro_id INT,
    status VARCHAR(30),
    FOREIGN KEY (product_id) REFERENCES Product (product_id),
    FOREIGN KEY (distro_id) REFERENCES Distributor (distro_id),
    FOREIGN KEY (delivery_id) REFERENCES Delivery (delivery_id)
);

CREATE TABLE Sales (
    sale_id SERIAL PRIMARY KEY,
    sale_date DATE NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10, 2) NOT NULL,
    total_price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
);


INSERT:

-- Populating Branch Table
INSERT INTO Branch (branch_name, branch_manager, district) VALUES
            ('Central Branch', 'Tomas Semethe', 'Maseru'),
            ('North Branch', 'Lefu Koleke', 'Leribe'),
            ('East Branch', 'Jobo Silase', 'Qachas Nek'),
            ('South Branch', 'Bobo Masei', 'Mohales Hoek');

-- Populating Department Table
INSERT INTO Department (department_name, H0D, branch_id) VALUES
            ('Sales', 'Head of Sales', 1),
            ('Marketing', 'Head of Marketing', 2,,
            ('Human Resources', 'Head of HR', 3),
            ('Finance', 'Head of Finance', 4);

-- Populating Employee Table
INSERT INTO Employee (emp_firstname, emp_lastname, email, mobile_no, emo_position, department_id)
VALUES
            ('Tokelo', 'Semethe', 'tokelo.smith@email.com', 57515248, 'Manager', 1),
            ('Lefu', 'Kompone', 'l.kompone@email.com', 62584568, 'Assistant Manager', 1),
            ('Botle', 'Leqe', 'botleleqe@email.com', 58764289, 'Manager', 2),
            ('Alice', 'Lebelo', 'alice.lee@email.com', 51242715, 'Assistant Manager', 2);

-- Populating Distributor Table
INSERT INTO Distributor (distro_name, address, email, mobile_no, distro_type) VALUES
            ('Lakeside', 'Roma, Maseru', 'lakesideoffsale@email.com', 50521545, 'Off-sales'),
            ('Bothobapelo', 'Berea', 'thobas@email.com', 59456666, 'Tarvern');
```

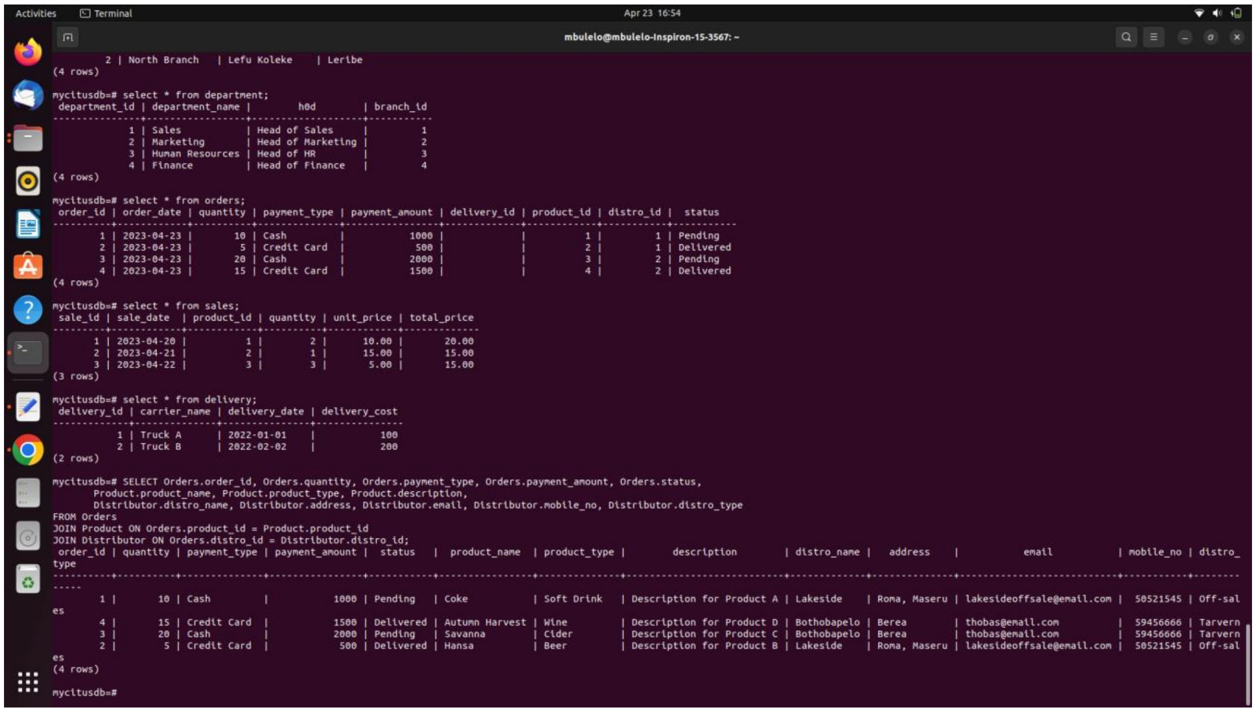
```
-- Populating Product Table
INSERT INTO Product (product_name, product_type, description) VALUES
            ('Coke', 'Soft Drink', 'Description for Product A'),
            ('Hansa', 'Beer', 'Description for Product B'),
            ('Savanna', 'Cider', 'Description for Product C'),
            ('Autumn Harvest', 'Wine', 'Description for Product D');

-- Populating Delivery Table
INSERT INTO Delivery (carrier_name, delivery_date, delivery_cost) VALUES
            ('Truck A', '2022-01-01', 100),
            ('Truck B', '2022-02-02', 200);

-- Populating Orders Table
INSERT INTO Orders (quantity, payment_type, payment_amount, product_id, distro_id, status) VALUES
            (10, 'Cash', 1000, 1, 1, 'Pending'),
            (5, 'Credit Card', 500, 2, 1, 'Delivered'),
            (20, 'Cash', 2000, 3, 2, 'Pending'),
            (15, 'Credit Card', 1500, 4, 2, 'Delivered');

--Populate Sales Table
INSERT INTO Sales (sale_date, product_id, quantity, unit_price, total_price)
VALUES
    ('2023-04-20', 1, 2, 10.00, 20.00),
    ('2023-04-21', 2, 1, 15.00, 15.00),
    ('2023-04-22', 3, 3, 5.00, 15.00);

TRIGGERS:

--Trigger generate receipt for a sale
CREATE TRIGGER generate_receipt
AFTER INSERT ON Sales
FOR EACH ROW
BEGIN
    DECLARE product_name VARCHAR(50);
    SELECT product_name INTO product_name FROM Product WHERE product_id = NEW.product_id;
    SELECT CONCAT('--------------------------\n',
                'Sale ID: ', NEW.sale_id, '\n',
                'Sale Date: ', NEW.sale_date, '\n',
                'Product: ', product_name, '\n',
                'Quantity: ', NEW.quantity, '\n',
                'Unit Price: $', NEW.unit_price, '\n',
                'Total Price: $', NEW.total_price, '\n',
                '--------------------------\n') AS 'Receipt';
END;



--Trigger to prevent a distributor from being deleted if they have pending orders:
```

```
CREATE TRIGGER prevent_delete_distributor
BEFORE DELETE ON Distributor
FOR EACH ROW
BEGIN
    DECLARE orders_count INT;
    SELECT COUNT(*) INTO orders_count FROM Orders WHERE distro_id = OLD.distro_id AND status <>
'Delivered';
    IF orders_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot delete distributor with pending orders';
    END IF;
END;

--Trigger to update the status column in the Orders table when a delivery is made:
CREATE TRIGGER update_order_status
AFTER INSERT ON Delivery
FOR EACH ROW
BEGIN
    UPDATE Orders
    SET status = 'Delivered'
    WHERE order_id = NEW.order_id;
END;

--Trigger to update the H0D (Head of Department) column in the Department table when a new employee is
added:
CREATE TRIGGER update_hod
AFTER INSERT ON Employee
FOR EACH ROW
BEGIN
    UPDATE Department
    SET H0D = CONCAT(NEW.emp_firstname, ' ', NEW.emp_lastname)
    WHERE department_id = NEW.department_id;
END;

--trigger to update order status at delivery
 CREATE TRIGGER update_order_status
AFTER INSERT ON Delivery
FOR EACH ROW
BEGIN
    UPDATE Orders
    SET status = 'Delivered'
    WHERE order_id = NEW.order_id;
END;
```

The entities were created and populated:



```
mbulelo@mbulelo-Inspiron-15-3567:~$ docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS                  PORTS                                       NAMES
17061432e37f   citusdata/citus:11.2.0 "docker-entrypoint.s…"   15 hours ago   Up 2 hours (healthy)    5432/tcp                                    mbulelo_citus-worker-south_1
4ec0a9be8ca0   citusdata/citus:11.2.0 "docker-entrypoint.s…"   15 hours ago   Up 2 hours (healthy)    5432/tcp                                    mbulelo_citus-worker-north_1
2763a5241937   citusdata/citus:11.2.0 "docker-entrypoint.s…"   15 hours ago   Up 2 hours (healthy)    5432/tcp                                    mbulelo_citus-worker-east_1
2ddda3217a7b   citusdata/citus:11.2.0 "docker-entrypoint.s…"   15 hours ago   Up 2 hours (healthy)    0.0.0.0:5432->5432/tcp, :::5432->5432/tcp   mbulelo_citus-coordinator_1
mbulelo@mbulelo-Inspiron-15-3567:~$ docker exec -it mbulelo_citus-coordinator_1 psql -U citus
psql (15.1 (Debian 15.1-1.pgdg110+1))
Type "help" for help.

citus=# \c mycitusdb
You are now connected to database "mycitusdb" as user "citus".
mycitusdb=# SELECT create_distributed_table('branch', 'branch_id');
SELECT create_distributed_table('department', 'department_id');
SELECT create_distributed_table('employee', 'emp_id');
SELECT create_distributed_table('distributor', 'distro_id');
SELECT create_distributed_table('product', 'product_id');
SELECT create_distributed_table('delivery', 'delivery_id');
SELECT create_distributed_table('orders', 'order_id');
SELECT create_distributed_table('sales', 'sale_id');
ERROR:  relation "branch" does not exist
LINE 1: SELECT create_distributed_table('branch', 'branch_id');
                                        ^
ERROR:  relation "department" does not exist
LINE 1: SELECT create_distributed_table('department', 'department_id...
                                        ^
ERROR:  relation "employee" does not exist
LINE 1: SELECT create_distributed_table('employee', 'emp_id');
                                        ^
ERROR:  relation "distributor" does not exist
LINE 1: SELECT create_distributed_table('distributor', 'distro_id');
                                        ^
ERROR:  relation "product" does not exist
LINE 1: SELECT create_distributed_table('product', 'product_id');
                                        ^
ERROR:  relation "delivery" does not exist
LINE 1: SELECT create_distributed_table('delivery', 'delivery_id');
                                        ^
ERROR:  relation "orders" does not exist
LINE 1: SELECT create_distributed_table('orders', 'order_id');
                                        ^
ERROR:  relation "sales" does not exist
LINE 1: SELECT create_distributed_table('sales', 'sale_id');
                                        ^
mycitusdb=# \dt
Did not find any relations.
mycitusdb=# CREATE TABLE Branch (
    branch_id SERIAL PRIMARY KEY,
    branch_name VARCHAR(20),
    branch_manager VARCHAR(30),
    district VARCHAR(30) NOT NULL
);
CREATE TABLE
mycitusdb=# CREATE TABLE Department (
    department_id SERIAL PRIMARY KEY,
    department_name VARCHAR(30) NOT NULL,
```



```
       2 | North Branch   | Lefu Koleke    | Leribe
(4 rows)

mycitusdb=# select * from department;
 department_id | department_name |        h0d        | branch_id
---------------+-----------------+-------------------+-----------
             1 | Sales           | Head of Sales     |         1
             2 | Marketing       | Head of Marketing |         2
             3 | Human Resources | Head of HR        |         3
             4 | Finance         | Head of Finance   |         4
(4 rows)

mycitusdb=# select * from orders;
 order_id | order_date | quantity | payment_type | payment_amount | delivery_id | product_id | distro_id |  status
----------+------------+----------+--------------+----------------+-------------+------------+-----------+-----------
        1 | 2023-04-23 |       10 | Cash         |           1000 |             |          1 |         1 | Pending
        2 | 2023-04-23 |        5 | Credit Card  |            500 |             |          2 |         1 | Delivered
        3 | 2023-04-23 |       20 | Cash         |           2000 |             |          3 |         2 | Pending
        4 | 2023-04-23 |       15 | Credit Card  |           1500 |             |          4 |         2 | Delivered
(4 rows)

mycitusdb=# select * from sales;
 sale_id | sale_date  | product_id | quantity | unit_price | total_price
---------+------------+------------+----------+------------+-------------
       1 | 2023-04-20 |          1 |        2 |      10.00 |       20.00
       2 | 2023-04-21 |          2 |        1 |      15.00 |       15.00
       3 | 2023-04-22 |          3 |        3 |       5.00 |       15.00
(3 rows)

mycitusdb=# select * from delivery;
 delivery_id | carrier_name | delivery_date | delivery_cost
-------------+--------------+---------------+---------------
           1 | Truck A      | 2022-01-01    |           100
           2 | Truck B      | 2022-02-02    |           200
(2 rows)

mycitusdb=# SELECT Orders.order_id, Orders.quantity, Orders.payment_type, Orders.payment_amount, Orders.status,
       Product.product_name, Product.product_type, Product.description,
       Distributor.distro_name, Distributor.address, Distributor.email, Distributor.mobile_no, Distributor.distro_type
FROM Orders
JOIN Product ON Orders.product_id = Product.product_id
JOIN Distributor ON Orders.distro_id = Distributor.distro_id;
 order_id | quantity | payment_type | payment_amount |  status   |  product_name  | product_type |        description        | distro_name |   address    |            email             | mobile_no | distro_
type
----------+----------+--------------+----------------+-----------+----------------+--------------+---------------------------+-------------+--------------+------------------------------+-----------+--------
-----
        1 |       10 | Cash         |           1000 | Pending   | Coke           | Soft Drink   | Description for Product A | Lakeside    | Roma, Maseru | lakesideoffsale@email.com    |  50521545 | Off-sal
es
        4 |       15 | Credit Card  |           1500 | Delivered | Autumn Harvest | Wine         | Description for Product D | Bothobapelo | Berea        | thobas@email.com             |  59456666 | Tarvern
        3 |       20 | Cash         |           2000 | Pending   | Savanna        | Cider        | Description for Product C | Bothobapelo | Berea        | thobas@email.com             |  59456666 | Tarvern
        2 |        5 | Credit Card  |            500 | Delivered | Hansa          | Beer         | Description for Product B | Lakeside    | Roma, Maseru | lakesideoffsale@email.com    |  50521545 | Off-sal
es
(4 rows)

mycitusdb=#
```

RELATIONSHIPS:

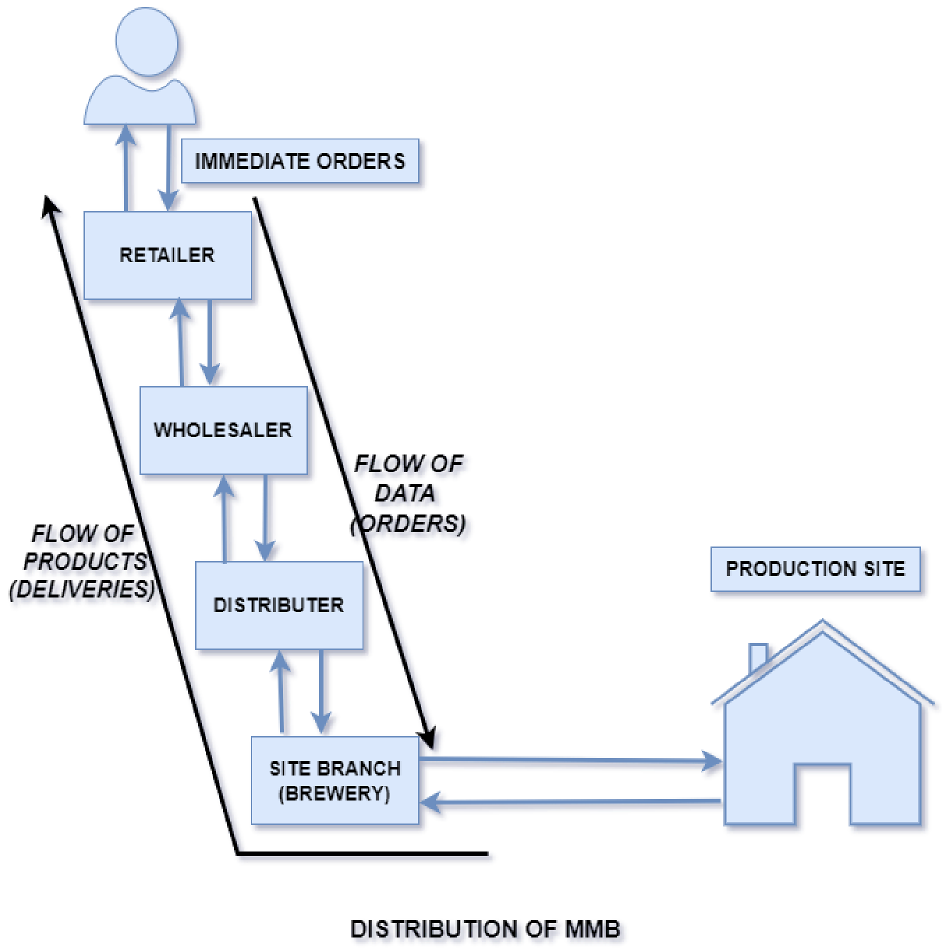
Employee: exist to be linked to others entities and track who do what, when, etc.

Branch: site that distribute the product.

Distributer: stores the data about who the distributor purchases

Product: stores the type, description and the name of the product
Sales: stores the information about the inventory sales
Delivery: Stores information about the deliveries made from

SYSTEM WORKFLOW:



DEVELOPMENT STATUS:

In progress……

MEMBERS:

201902624	***L LETSIE***
201902131	***M SEEQELA***
201902703	***F MOTSU***
202004214	***N TLALI***

Bibliography

Food and beverage. (2023). Retrieved 2023, from Rockwell Automation: https://www.rockwellautomation.com/en-us/industries/food-beverage/brewery-production-automation.html

Barton, D. (2021). Opening a Microbrewery. *From Brewmaster to Profitable Brewery Operations Management*.

BEGG, C. E., & Connoly, T. M. (2014). Database Systems, A Practical Approach to Design, Implementation, and Managemen. *Introduction to Distributed Database Management Systems*.

Burns, B. (2018). *Designing Distributed Systems.* Sebastopol: O'Reilly Media, Inc.

Fadoua, H., & Grissa , T. A. (n.d.). Intelligent Implementation Processor Design for Oracle. *Distributed Databases System*, 12.

Fogel, S. (2010). *Oracle Database Administrator's Guide.* Oracle.

Howley, E., Jim , D., & Hongliang , L. (2014). A Coevolutionary Approach to the Beer Game. *Individual Versus Group Rationality*, 19.

Jayashree, J. (2013). Distributed Database Management System and Query Processing. *International Journal of Computer Science And Technology*, 5.

M, L., & R, E. (2012). On the Move to Meaningful Internet Systems. *Information Infrastructures for Utilities Management in the Brewing Industr*.

MAGAZINE, A. O. (2022). *Brewering at its peak(Maluti Mountain Brewery).* Maseru Lesotho: AFRICA OUTLOOK MAGAZINE.

Melanie. (2023). The Beer Supply Chain in 2023 and Beyond. *unleashed inventory management software*.

Muriu, S. (2019). *studocu.* Retrieved from https://www.studocu.com/row/document/chuka-university/computer-science/database-project-proposal/5736303

Ozsu, T. M., & Valduries, P. (2011). *Principle of Distributed Database Systems.* London: Pearson Edufcation, Inc.

Winnie, N. (2011). *A public monitoring and evaluation web-application for the breweries.* Kampala: Kampala International University.