**Greed’s Food Productions**

**Database Design Project, Spring 2021**

**Webster University Thailand**

**Cyrus, cyrusbj@gmail.com**

# PROJECT STATEMENT

As the database expert for a Database Management Company, you have been assigned to create a database for (Random) Food Supplier/ Manufacturing Company to manage scheduling needs for delivering, stocking, and storing items, raw materials, and products.

# DATA DESCRIPTION

From interview with multiple employees working at the food company as well as few of their clients, the following information has been gathered.

1. The company not only produces products but also supplies these products to their clients. Having need of separate tables for clients, products, and employee data.
2. Clients are usually registered with client’s name, id, address, email, phone number.
3. Product information such as product name, id, price, source, and stock duration are necessary for the company to function.
4. Employee data such as name, position, id, address, email, phone number are stored in the company database.
5. There is an order table(quantity, delivery date, total cost) that draws relation from client table (client id) and product table(product id, price, stock duration) to keep track of individual orders.
6. Employees are assigned to work for client order in teams. Employees in team can be assigned more than one order at a time.
7. Total cost of order is derived from product cost and quantity after order has been created.

# QUERIES REQUIRED

The completed system must be able to satisfy the following queries:

**For Employee**

1. List the employee’s name, position, id, and contact information.

SELECT \* FROM Employee;

1. List employees by team.

SELECT Employee.e\_name, assigned\_to.team FROM Employee, assigned\_to INNER JOIN Employee ON Employee.employee\_id=assigned\_to.employee\_id ORDER BY team;

**For Order**

1. List the order name, total, quantity, delivery date.

SELECT \* FROM Order;

1. Find the total cost using price from Product and quantity from Orders.

SELECT SUM(Product.price + Orders.quantity) AS total FROM Product, Orders WHERE Orders.o\_name=(‘home’);

1. Teams are assigned to each order.

SELECT assigned\_to.team, Order.o\_name FROM assigned\_to, Orders INNER JOIN Orders ON assigned\_to.order\_id=Orders.order\_id

**For Client/Customer**

1. List the client’s name, id, and contact information.

SELECT \* FROM Order;

1. Display and find all the customers’ orders.

SELECT \* FROM Order WHERE client\_id = (SELECT client\_id FROM client WHERE c\_name = [name];

**For Product**

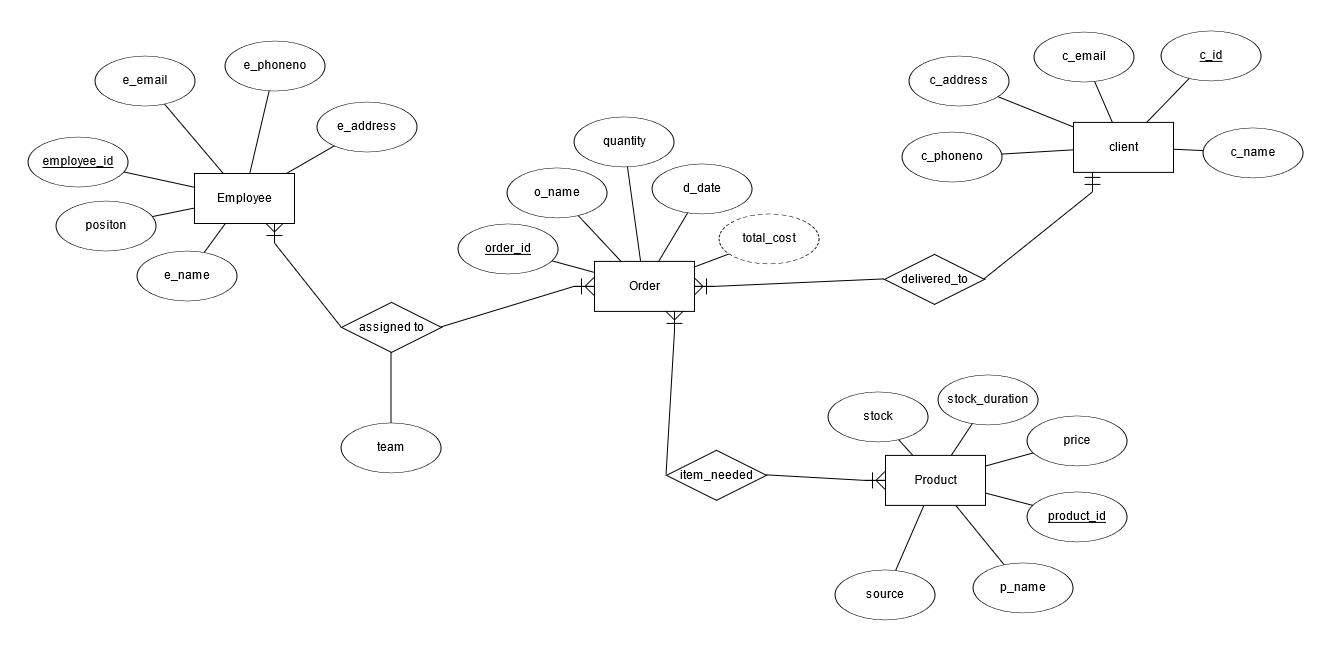
1. List the product name, price, stock duration, id, source.

SELECT \* FROM Product;

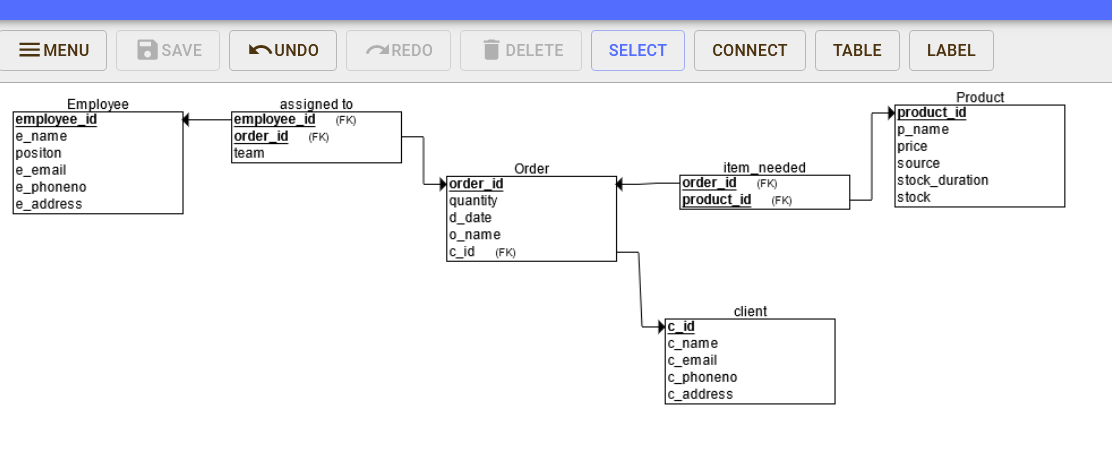
1. Sort product information based on duration, price, or order demand.

SELECT \* FROM Product ORDER BY price;

# ENTITY-RELATIONSHIP DIAGRAM



# QUERIES



# IMPLEMENTED QUERIES (Generated SQL)

CREATE TABLE Employee

(

e\_name VARCHAR(150) NOT NULL,

employee\_id INT NOT NULL,

positon VARCHAR(150) NOT NULL,

e\_email VARCHAR(150) NOT NULL,

e\_phoneno NUMERIC NOT NULL,

e\_address VARCHAR(150) NOT NULL,

PRIMARY KEY (employee\_id)

);

CREATE TABLE Product

(

p\_name VARCHAR(150) NOT NULL,

product\_id INT NOT NULL,

price FLOAT NOT NULL,

source VARCHAR(150) NOT NULL,

stock\_duration DATE NOT NULL,

stock INT NOT NULL,

PRIMARY KEY (product\_id)

);

CREATE TABLE client

(

c\_name VARCHAR(150) NOT NULL,

c\_id INT NOT NULL,

c\_email VARCHAR(150) NOT NULL,

c\_phoneno INT NOT NULL,

c\_address VARCHAR(150) NOT NULL,

PRIMARY KEY (c\_id)

);

CREATE TABLE Orders

(

quantity INT NOT NULL,

d\_date DATE NOT NULL,

o\_name VARCHAR(150) NOT NULL,

order\_id INT NOT NULL,

c\_id INT NOT NULL,

PRIMARY KEY (order\_id),

FOREIGN KEY (c\_id) REFERENCES client(c\_id)

);

CREATE TABLE assigned\_to

(

team CHAR(50) NOT NULL,

employee\_id INT NOT NULL,

order\_id INT NOT NULL,

PRIMARY KEY (employee\_id, order\_id),

FOREIGN KEY (employee\_id) REFERENCES Employee(employee\_id),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id)

);

CREATE TABLE item\_needed

(

order\_id INT NOT NULL,

product\_id INT NOT NULL,

PRIMARY KEY (order\_id, product\_id),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id),

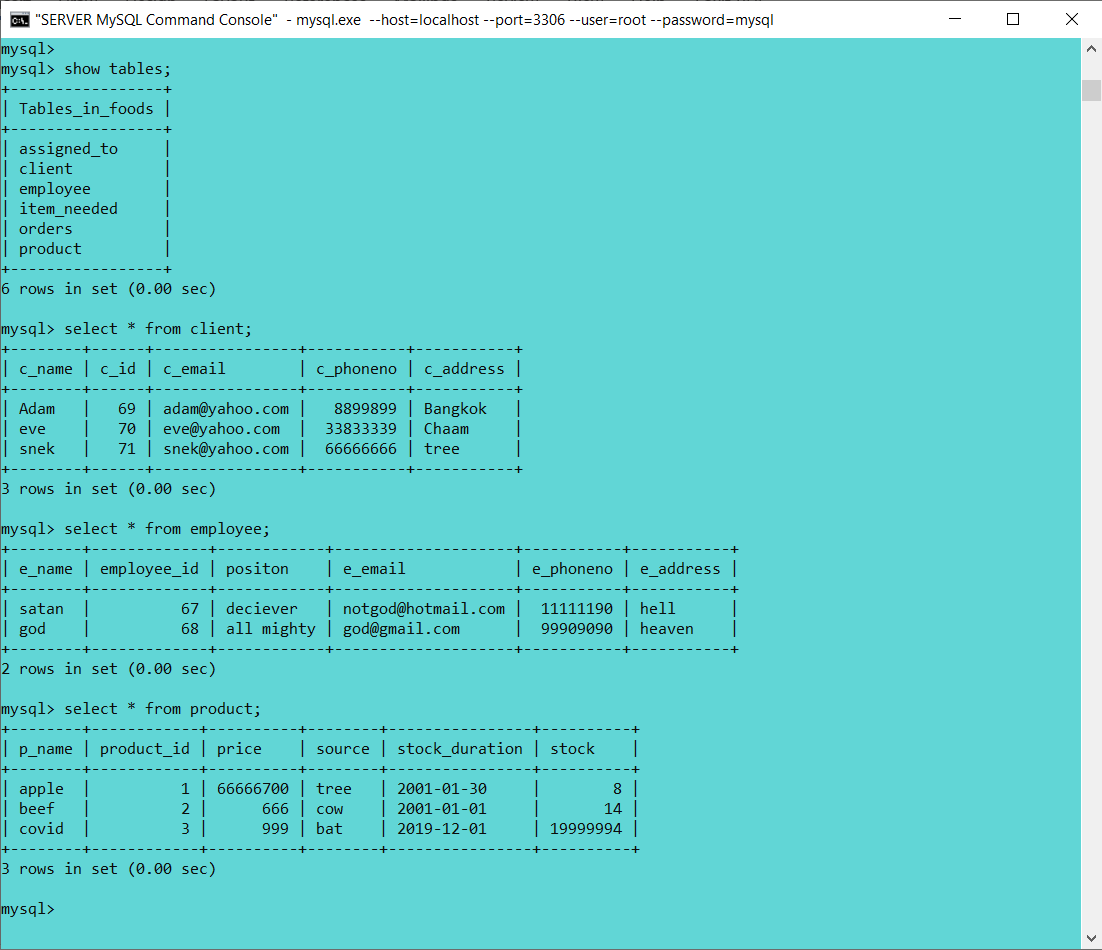
FOREIGN KEY (product\_id) REFERENCES Product(product\_id)

);

# Conditions for 3NF:

1. It has single valued attributes/columns, all the columns in a table should have unique names.
2. It does not have Partial Dependency.
3. There is no transition dependency among the attributes of any table.

# Sample Database:



# Source:

**GitHub ID-** https://github.com/Arnamist/DBApps-3-Installation-instructions-some-security

**Last update -**18/07/2021

**Version-** 2.1 (11 Commits)