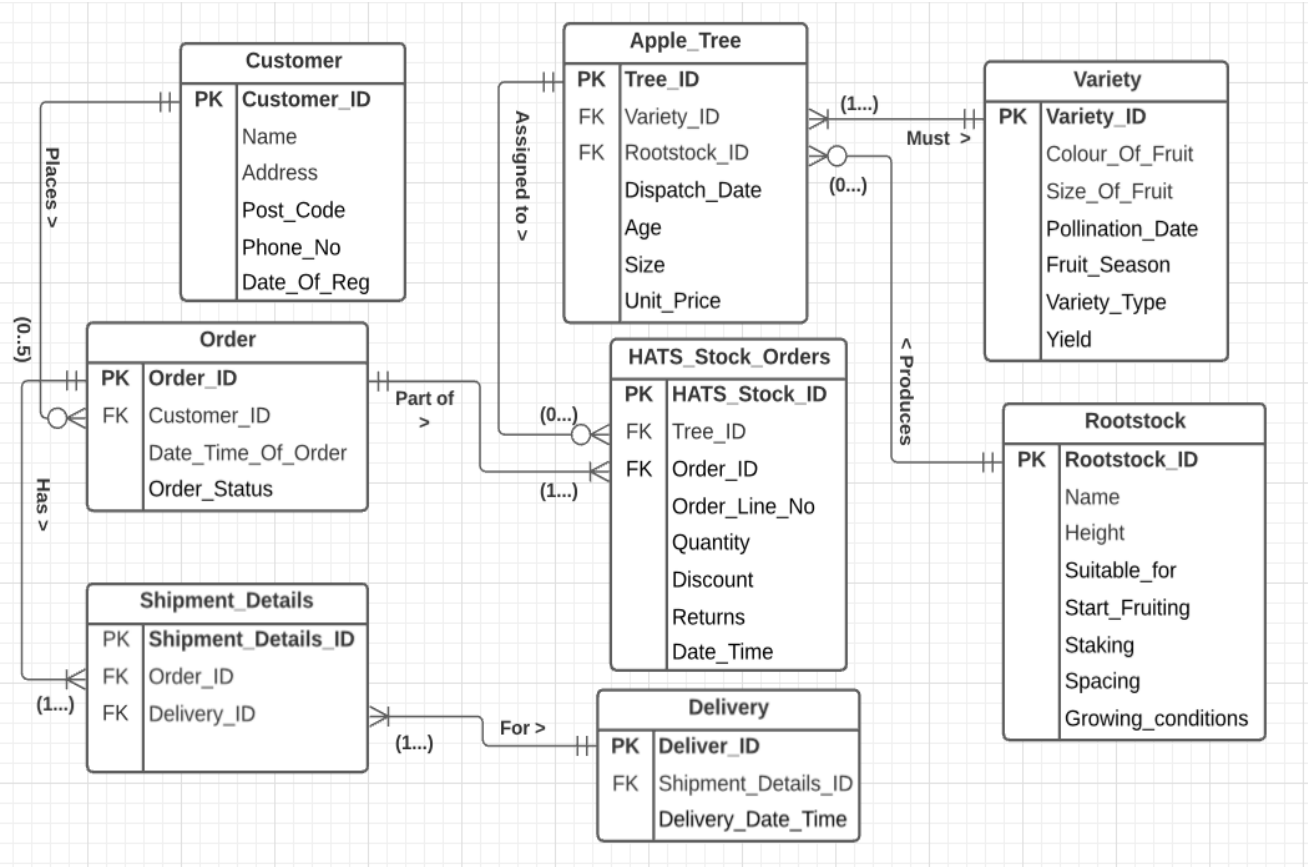
APPLE TREE SUPPLIER

Database

OCTOBER 13, 2020

Table of Contents:

ER Diagram:



Entities:

1. Apple_tree
2. Customer
3. HATS_Stock
4. Delivery
5. Rootstock
6. Variety
7. Order
8. Shipment_Details

Attributes:

Apple_tree:

- Tree_ID
- Variety_ID
- Rootstock_ID
- Dispatch_Date
- Age
- Order_ID
- Size

Variety:

- Varirty_ID
- ColourOfFruit
- SizeOfFruit
- Varirty_Type
- Yield
- Fruit_Season
- Pollination_Group

Rootstock:

- Rootstock_ID
- Name
- Height
- Suitable_For
- Start_Fruiting
- Staking
- Spacing
- Growing_Condtions

Customer:

- Customer_ID
- Name
- Address
- Postcode
- Phone
- DateOfReg

Order:

- Order_ID
- Customer_ID
- Date_Time
- Order_Status

HATS_Stock:

- HATS_Stock_ID
- Tree_ID
- Order_ID
- Order_(line)_No
- Quantity
- Date_Time

Shipment_Details:

- Shipment_ID
- Order_ID
- Delivery_ID

Delivery:

- Delivery_ID
- Order_ID
- Delivery_Date_Time

Primary keys And Foreign Keys:

Apple_tree:

Primary Key: Tree_ID

Foreign Key: Variety_ID, Rootstock_ID

Variety:

Primary Key: Varirty_ID

Foreign Key: Null

Rootstock:

Primary Key: Rootstock_ID

Foreign Key: Null

Customer:

Primary Key: Cust_ID

Foreign Key: Null

HATS_Stock:

Primary Key: Hats_Stock_ID

Foreign Key: Tree_ID, Order_ID

Order:

Primary Key: Order_ ID

Foreign Key: Customer_ID

Shipment Details:

Primary key: Shipment_ID

Foreign key: Order_ID

Delivery:

Primary Key: Delivery_ID

Foreign Key: Shipment_ID

Relationships, Connectivity, Name and Direction, Participation and Cardinality:

Relationships:

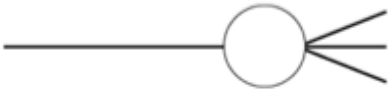
- Describes how one or more entities interact with each other.
- A verb is often used to describe the relationship.

Cardinality:

- The count of instances that are allowed or are necessary between entity relationships.
- Broken into to parts minimum/maximum.
  Minimum: Represents the minimum number of instances that are required in a relationship.
  Maximum: Represents the maximum number of instances that are required in a relationship.

- An easy way to represent cardinality is through the crow's foot notation.
- The mandatory vs optional refers to the minimum and the one vs many refers to the maximum.

-

Symbol	Meaning	Number
	One	N/A
	Many	N/A
	Mandatory-One	Exactly one
	Optional-One	Zero or one
	Mandatory-Many	One or More
	Optional-Many	Zero or more

Direction:

- The direction of a relationship indicates the originating entity of the relationship. The entity from which a relationship originates is the parent entity, the entity where the relationship terminates is the child entity. The one-way is unidirectional and two-way is bidirectional.

Participation:

- In a relationship, participation specifies the presence of an entity when it is related to another entity in the relationship.

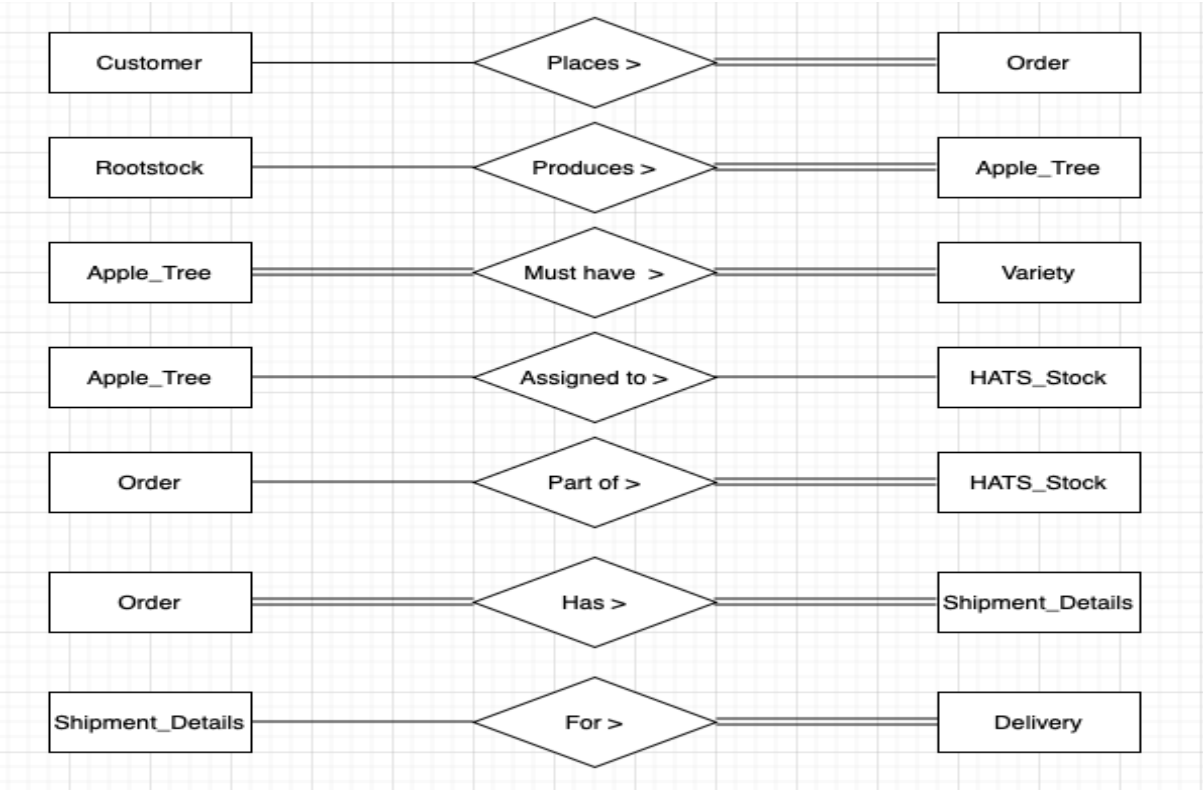
There are two types of participation.

1. Total participation
2. Partial participation.

- **Total Participation:**
  Each entity in the entity set occurs in at least one relationship in that relationship set, and each entity is involved in the relationship at least once. Donated by double lines.

- **Partial Participation:**
  Each entity in the entity set may not occur in at least one relationship in that relationship set. Donated by a single line.

The participations of the entities are shown in the diagram



1. Customer – Order
   Cardinality:
   (one to many)
   There is a one to many relationship between Customer and Order.
   Direction:
   In this case the relationship is optional.

2. Rootstock - Apple_Tree
   Cardinality:
   (one to many)
   There is a one to many relationship between Rootstock and Apple_Tree.
   Direction:
   In this relationship Rootstock is the parent entity and Apple_Tree is the child entity.

3. Apple_Tree - Variety
   Cardinality:
   (one to many)
   There is a one to many relationship between variety and Apple_Tree.
   Direction:
   In this relationship Apple_Tree is the parent entity and Variety is the child entity.

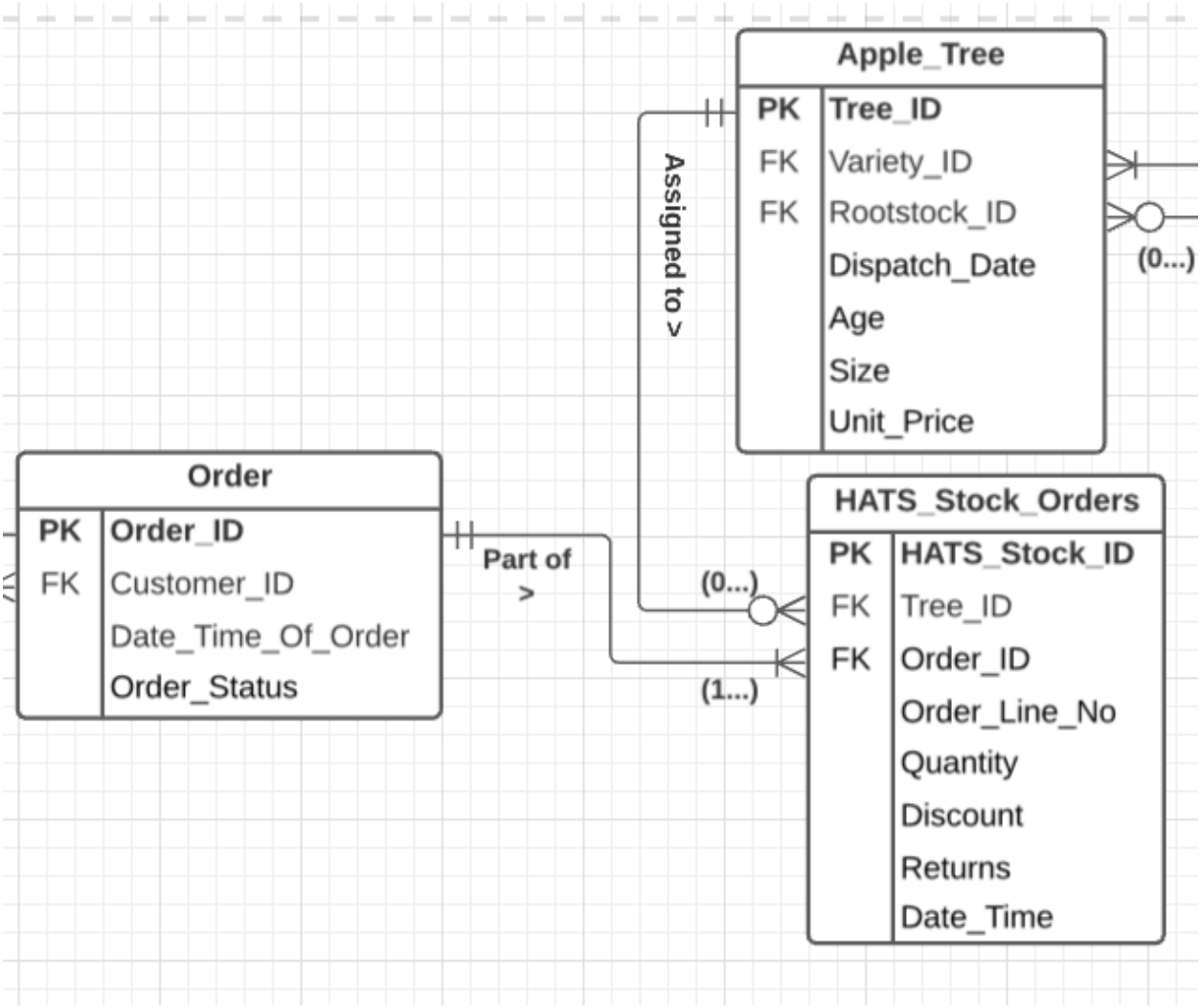
4. HATS_Stock – Apple_Tree
   Cardinality:
   (one to 0 or many)
   There is a many to many relationship between HATS_Stock and Apple_Tree.
   Direction: optional

5. Order - HATS_Stock (One to many)
   Cardinality:
   (One to many)
   There is a One to many relationship between Order and HATS_Stock.
   Direction:
   In this direction Order is the parent entity and HATS_Stock is the child entity.

6. Order - Delivery (one to many)
   Cardinality:
   (one to many)
   There is one to many relationship between Order_Details and Delivery.
   Direction:
   In this relationship Order is the parent entity and Delivery is the child entity.

M-M Relationships and Hidden Entities:

The Order entity and Apple_Tree entity contain many to many relationships so it is broken into two one to many relationships and a hidden entity is created which is called a associative entity i.e (HATS_Stock).



Detailing an assumption:

I have made an assumption that a **Customer** has a one-to-many Relationship with a purchase **Order** because a customer can place many orders, but a given purchase order can be placed by only one customer. Once an order is placed, I will say that it is mandatory for an order to have a customer related to it but optional for a customer to place an order.

Every entity has then been assigned a unique ID (Primary key), a primary key is used to ensure data in the specific table is unique for example customer has been assigned a primary keys to differentiate customers from one another, the same has also been done for Orders, A foreign keys is then used to link two tables together, it is a field or collection of fields in one table that refers to the primary key in another table, foreign keys are used to prevent actions that would destroy links between each other.

Tables are related to other tables with a primary key and foreign key they are often used to define many one to one relationships. The purpose of a foreign key is to ensure referential integrity of the data.

To enforce this we placed a foreign key of the customer in the order table and have it reference the primary key of the customer table, this way we ensure that orders in the order table are related to a specific customer in the customer table and vice versa. This similar logic is also applied to the other tables.

A many to many relationship occurs when multiple records in a table are associated with multiple records in another table, In my ERD many trees are grafted from the same rootstock and each tree can be a different combination of rootstock and variety therefore different trees have a unique tree ID's, and every tree ID can have multiple trees grafted from the same rootstock allowing the stock to stock trees in bulk, this allows customers to able to order a maximum of 5 trees on one order, hence why I had a M:M(many to many) relationship between the **Order** and **Apple_Tree** entity which affected their relationship and cardinality, so to resolve this I have broken down the M:M relationship into two one to many relationships by creating an associative entity (HATS_Stock_Orders) as shown in the image above, I also added attributes such as Order line no and Quantity to HATS_Stock_Orders to determine how many items of that order line the customer has ordered, and a Discount attribute in case of promotions.

Now the relationship between Apple_Tree and HATS_Stock_Orders is a mandatory one to optional many because there must exist a tree for HATS_Stock_Orders but optional for that tree to be ordered, and it is mandatory for HATS_Stock_Order to have at least on order if an order is placed.

Rootstock and Apple_Tree has a one to optional many because there must exist a Rootstock to produce apple trees and a apple tree must be grafted from at least one rootstock, but there can also exist a rootstock that doesn't get used to graft any trees so we have the optional 0 to many notation.

With variety and Apple_Tree we have a one to many mandatory because there must be a variety for one or more tree and a tree must have at least one variety.

There is also a one to many mandatory relationship between Order and HATS_Stock_Orders because HATS_Stock_Order must contain at least one order and an order can be a part of one or many HATS_Stock_Orders.

I also think Rootstock and Variety should be separate entities as they both have their own attributes that could get confused and cause errors, i.e type, if I were to place both Rootstock and Variety as one entity this would affect their attributes list and relationship between them and the Apple_Tree entity as Rootstock determines the type and height of the apple tree, and variety determines the color, size, yield, fruit season, pollination group and variety type.

BY AHMED ABOO