Tree Nursery Database

Jason Hodge

Ian Becker

Data Management MSIS 537L

16 December 2022

Table of Contents

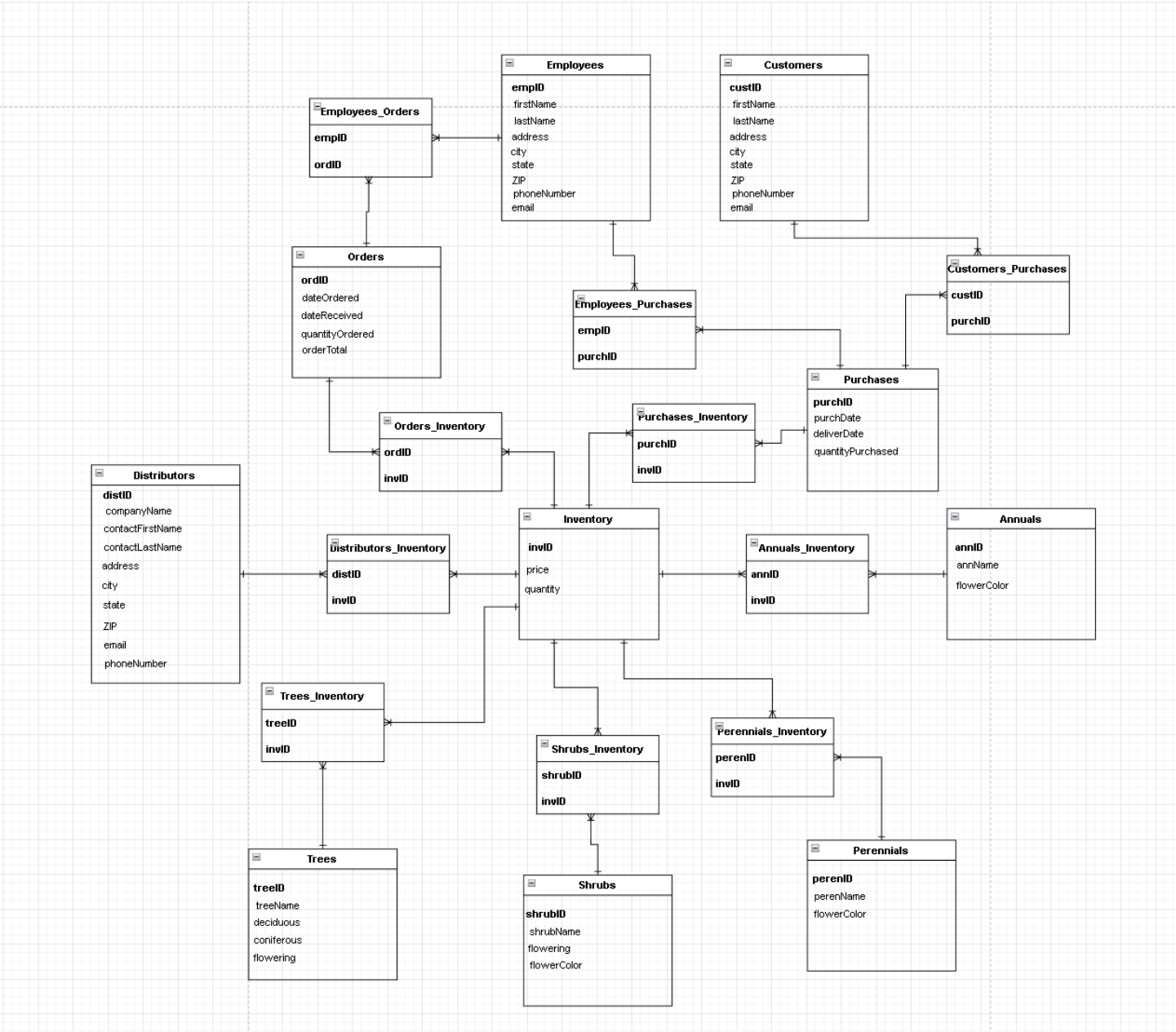
Contents

Executive Summary

This database can be used to keep a tree nursery organized by managing inventory and orders amongst customers and employees, as well as distributors that provide inventory.

This database shows and keeps track of a nurseries inventory and its types of plants categorized into separate tables all under one inventory. This database also includes employees who can order inventory from distributors, as well as purchase inventory from the nursery. Furthermore, we also have customers who come to purchase inventory as trees, shrubs, perennials, and annuals from the nursery. This is done through the use of tables represented in the logical and physical models and create tables statements. With this information and corresponding data, wanted information can be found through useful queries. Examples of these are presented.

Logical Model



Physical Model

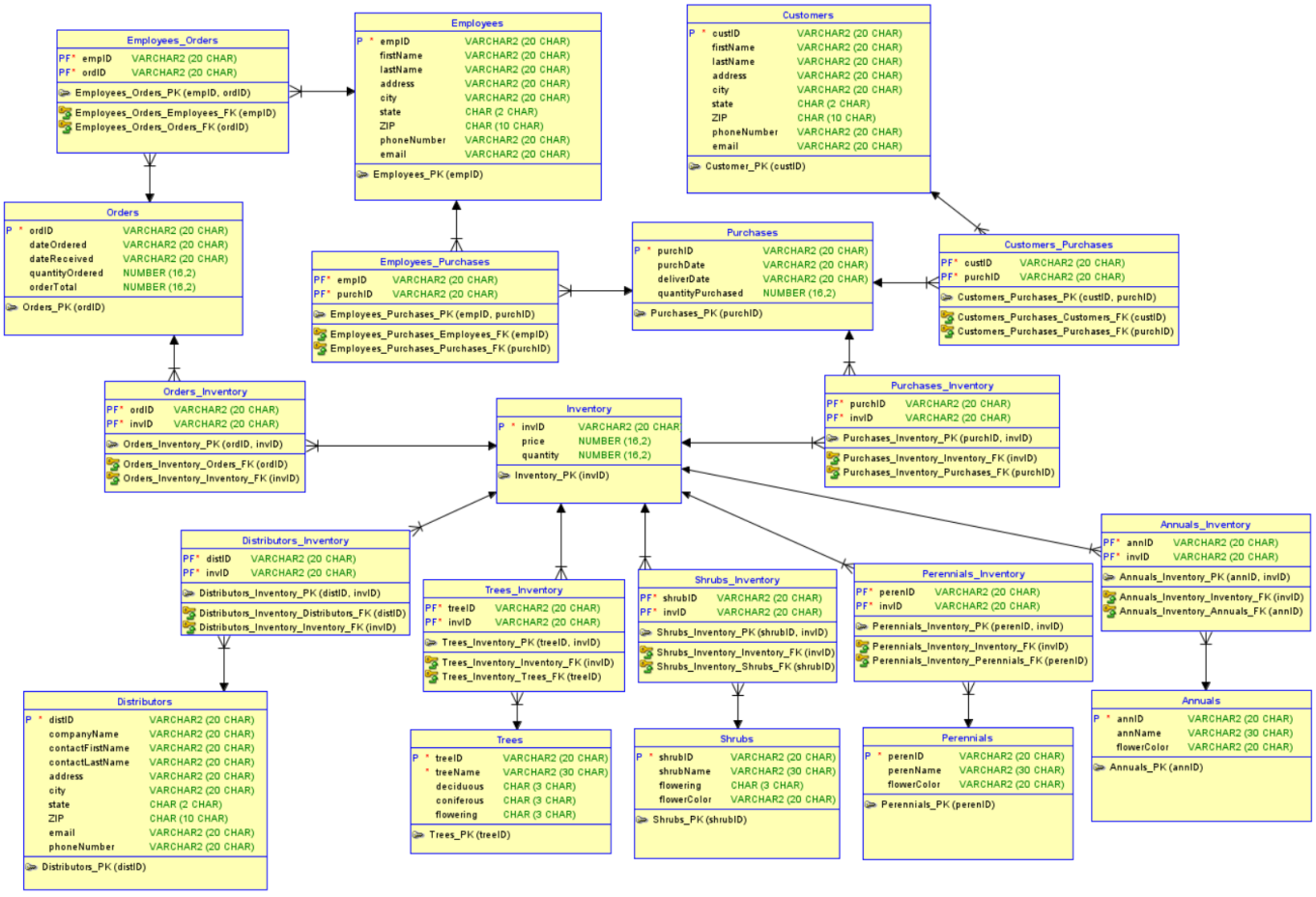


Table Statements

```
CREATE TABLE annuals (
    annid       VARCHAR2(20 CHAR) NOT NULL,
    annname     VARCHAR2(30 CHAR),
    flowercolor VARCHAR2(20 CHAR)
);

ALTER TABLE annuals ADD CONSTRAINT annuals_pk PRIMARY KEY ( annid );
```

*This table creates the annuals table, which is a category of plant and is connected with the overall inventory.

Functional Dependencies: annid -> annname, flowercolor

```
CREATE TABLE customers (
    custid      VARCHAR2(20 CHAR) NOT NULL,
    firstname   VARCHAR2(20 CHAR),
    lastname    VARCHAR2(20 CHAR),
    address     VARCHAR2(20 CHAR),
    city        VARCHAR2(20 CHAR),
    state       CHAR(2 CHAR),
    zip         CHAR(10 CHAR),
    phonenumber VARCHAR2(20 CHAR),
    email       VARCHAR2(20 CHAR)
);
```

```
ALTER TABLE customers ADD CONSTRAINT customer_pk PRIMARY KEY ( custid );
```

*This table is the customers table, which are customers that have made purchases from the nursery's inventory.

Functional Dependencies: custid -> firstname, lastname, address, city, state, zip, phonenumber, email

```
CREATE TABLE distributors (
    distid          VARCHAR2(20 CHAR) NOT NULL,
    companyname     VARCHAR2(20 CHAR),
    contactfirstname VARCHAR2(20 CHAR),
    contactlastname  VARCHAR2(20 CHAR),
    address         VARCHAR2(20 CHAR),
    city            VARCHAR2(20 CHAR),
    state           CHAR(2 CHAR),
    zip             CHAR(10 CHAR),
    email           VARCHAR2(20 CHAR),
    phonenumber     VARCHAR2(20 CHAR)
);
```

```
ALTER TABLE distributors ADD CONSTRAINT distributors_pk PRIMARY KEY ( distid );
```

*This table is the distributors table, which are the different distributors the employees purchase their inventory from.

Functional Dependencies: distid -> companyname, contactfirstname, contactlastname, address, city, state, zip, email, phonenumber

```
CREATE TABLE employees (
    empid       VARCHAR2(20 CHAR) NOT NULL,
    firstname   VARCHAR2(20 CHAR),
    lastname    VARCHAR2(20 CHAR),
    address     VARCHAR2(20 CHAR),
    city        VARCHAR2(20 CHAR),
    state       CHAR(2 CHAR),
    zip         CHAR(10 CHAR),
    phonenumber VARCHAR2(20 CHAR),
    email       VARCHAR2(20 CHAR)
);

ALTER TABLE employees ADD CONSTRAINT employees_pk PRIMARY KEY ( empid );
```

*This table is the employees table, which provides some relevant necessary information about the employees.

Functional Dependencies: empid -> firstname, lastname, address, city, state, zip, phonenumber, email

```
CREATE TABLE inventory (
    invid    VARCHAR2(20 CHAR) NOT NULL,
    price    NUMBER(16, 2),
    quantity NUMBER(16, 2)
);
```

```
ALTER TABLE inventory ADD CONSTRAINT inventory_pk PRIMARY KEY ( invid );
```

*This table is the inventory, which is a major table in connecting the database. This table provides the link between those that purchase, order, and supply inventory as well as the different types of inventory the nursery has to offer.

Functional Dependencies: invid -> price, quantity

```
CREATE TABLE orders (
    ordid           VARCHAR2(20 CHAR) NOT NULL,
    dateordered     VARCHAR2(20 CHAR),
    datereceived    VARCHAR2(20 CHAR),
    quantityordered NUMBER(16, 2),
    ordertotal      NUMBER(16, 2)
);

ALTER TABLE orders ADD CONSTRAINT orders_pk PRIMARY KEY ( ordid );
```

*This table tracks the orders employees make when putting in an order with the distributors form more inventory.

Functional Dependencies: ordid -> dateordered, datereceived, quantityordered, ordertotal

```
CREATE TABLE perennials (
    perenid     VARCHAR2(20 CHAR) NOT NULL,
    perenname   VARCHAR2(30 CHAR),
```

```
    flowercolor VARCHAR2(20 CHAR)
);

ALTER TABLE perennials ADD CONSTRAINT perennials_pk PRIMARY KEY ( perenid );
```

*This table is one of the different categories of inventory the nursery offers for sale.

Functional Dependencies: perendid -> perenname, flowercolor

```
CREATE TABLE purchases (
    purchid          VARCHAR2(20 CHAR) NOT NULL,
    purchdate        VARCHAR2(20 CHAR),
    deliverdate      VARCHAR2(20 CHAR),
    quantitypurchased NUMBER(16, 2)
);

ALTER TABLE purchases ADD CONSTRAINT purchases_pk PRIMARY KEY ( purchid );
```

*This table tracks the purchases from both customers and employees, as both can make purchases from the nursery.

Functional Dependencies: purchid -> purchdate, deliverdate, quantitypurchased

```
CREATE TABLE shrubs (
```

```
    shrubid     VARCHAR2(20 CHAR) NOT NULL,
    shrubname   VARCHAR2(30 CHAR),
    flowering   CHAR(3 CHAR),
    flowercolor VARCHAR2(20 CHAR)
);

ALTER TABLE shrubs ADD CONSTRAINT shrubs_pk PRIMARY KEY ( shrubid );
```

*The shrubs table is another table that is connected to the inventory as one of the different categories of products the nursery has for sale.

Functional Dependencies: shrubid -> shrubname, flowering, flowercolor

```
CREATE TABLE trees (
    treeid     VARCHAR2(20 CHAR) NOT NULL,
    treename   VARCHAR2(30 CHAR) NOT NULL,
    deciduous  CHAR(3 CHAR),
    coniferous CHAR(3 CHAR),
    flowering  CHAR(3 CHAR)
);

ALTER TABLE trees ADD CONSTRAINT trees_pk PRIMARY KEY ( treeid );
```

*The trees table is also a table connected to the inventory as a different category of plant off the inventory.

Functional Dependencies: treeid -> treename, deciduous, coniferous, flowering

```
CREATE TABLE annuals_inventory (
    annid VARCHAR2(20 CHAR) NOT NULL,
    invid VARCHAR2(20 CHAR) NOT NULL
);

ALTER TABLE annuals_inventory ADD CONSTRAINT annuals_inventory_pk PRIMARY KEY ( annid, invid );
```

*This table serves as a connection table between annuals and inventory.

Functional Dependencies: None

```
CREATE TABLE customers_purchases (
    custid  VARCHAR2(20 CHAR) NOT NULL,
    purchid VARCHAR2(20 CHAR) NOT NULL
);

ALTER TABLE customers_purchases ADD CONSTRAINT customers_purchases_pk PRIMARY KEY ( custid, purchid );
```

*This table serves as a connection table between customers and purchases.

Functional Dependencies: None

```
CREATE TABLE distributors_inventory (
    distid VARCHAR2(20 CHAR) NOT NULL,
    invid  VARCHAR2(20 CHAR) NOT NULL
);

ALTER TABLE distributors_inventory ADD CONSTRAINT distributors_inventory_pk PRIMARY KEY ( distid, invid );
```

*This table serves as a connection table between distributors and inventory.

Functional Dependencies: None

```
CREATE TABLE employees_orders (
    empid VARCHAR2(20 CHAR) NOT NULL,
    ordid VARCHAR2(20 CHAR) NOT NULL
);

ALTER TABLE employees_orders ADD CONSTRAINT employees_orders_pk PRIMARY KEY ( empid, ordid );
```

*This table serves as a connection table between employees and orders.

Functional Dependencies: None

```
CREATE TABLE employees_purchases (
    empid   VARCHAR2(20 CHAR) NOT NULL,
    purchid VARCHAR2(20 CHAR) NOT NULL
);

ALTER TABLE employees_purchases ADD CONSTRAINT employees_purchases_pk PRIMARY KEY ( empid, purchid );
```

*This table serves as a connection table between employees and purchases.

Functional Dependencies: None

```
CREATE TABLE orders_inventory (
    ordid VARCHAR2(20 CHAR) NOT NULL,
    invid VARCHAR2(20 CHAR) NOT NULL
);

ALTER TABLE orders_inventory ADD CONSTRAINT orders_inventory_pk PRIMARY KEY ( ordid, invid );
```

*This table serves as a connection table between orders and inventory.

Functional Dependencies: None

```
CREATE TABLE perennials_inventory (
    perenid VARCHAR2(20 CHAR) NOT NULL,
    invid   VARCHAR2(20 CHAR) NOT NULL
```

);

ALTER TABLE perennials_inventory ADD CONSTRAINT perennials_inventory_pk PRIMARY KEY (perenid, invid);

*This table serves as a connection table between perennials and inventory.

Functional Dependencies: None

CREATE TABLE purchases_inventory (

    purchid VARCHAR2(20 CHAR) NOT NULL,

    invid   VARCHAR2(20 CHAR) NOT NULL

);

ALTER TABLE purchases_inventory ADD CONSTRAINT purchases_inventory_pk PRIMARY KEY (purchid, invid);

*This table serves as a connection table between purchases and inventory.

Functional Dependencies: None

CREATE TABLE shrubs_inventory (

    shrubid VARCHAR2(20 CHAR) NOT NULL,

    invid   VARCHAR2(20 CHAR) NOT NULL

);

ALTER TABLE shrubs_inventory ADD CONSTRAINT shrubs_inventory_pk PRIMARY KEY (shrubid, invid);

*This table serves as a connection table between shrubs and inventory.

Functional Dependencies: None

```
CREATE TABLE trees_inventory (
    treeid VARCHAR2(20 CHAR) NOT NULL,
    invid  VARCHAR2(20 CHAR) NOT NULL
);
```

ALTER TABLE trees_inventory ADD CONSTRAINT trees_inventory_pk PRIMARY KEY (treeid, invid);

*This table serves as a connection table between trees and inventory.

Functional Dependencies: None

SQL Queries

```
select C.firstName, C.lastName

from Customers C, Customers_Purchases CP, Purchases P

where C.custID = CP.custID

and CP.purchID = P.purchID

and P.quantityPurchased >= 14

Order By P.quantityPurchased DESC;
```

	FIRSTNAME	LASTNAME
1	Ron	Thompson

Displays the first and last name of customers who purchased a quantity greater than or equal to 14.

```
select I.price

from inventory I, trees_inventory TI, trees as T

where I.invID = TI.treeID

and TI.treeID = t.treeID

and t.flowering = 'yes';
```

Price

149.99 and 169.99

Displays the prices of flowering trees.

Known Problems

- Some of the sizes of the source types could be made bigger to enhance usability for longer date entries.
- Some functionality between tables as some queries did not work.

Future Enhancements

- Possibly combining employee and customer into a people table initially.
- Add a table for other products such as mulch, dirt, gravel, etc.
- Employees can get a reduced price for products, would likely require a redesign.