CPSC 304 Project Cover Page

Project Name: Seed Germ - A gardening management tool

Milestone #: __ 2_____

Date: ___13th_Oct, 2024_________

Group Number: ____14_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Shu (Charlie) Chen	61628137	i8r6v	schen622@student.ubc.ca
David Tianyi Yin	49385537	k1l2s	tianyiy.ubc@gmail.com
Lewis Li	39058169	b9d8f	weitianl@student.ubc.ca

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia.

2. Project Description

Seed Germ is a comprehensive system for managing and tracking plants within a garden, focusing on the plant growth data analytics. The system accommodates various plant types and stages, tracks scheduled and ad-hoc events, and provides traceability from seed purchase through to harvest and distribution. Its functionality includes maintaining records of plant event such as watering, weeding, etc., observations such as bud breaking, fruiting, etc. and generating timelines and analytics for efficient garden management.

3. ER Diagram:

*Please find the updated ER Diagram on the attached last page.

We have made the following changes:

- Order has Cultivar -> Plant is_on Order: Some plants in the database, such as lavender, doesn't have sub-varieties. Regardless, we want to track its growth data.
- Cultivar has stage -> Plant goes_throughstage: like the previous change, we now focus on tracking the grow of plant.
- Added location distinguished_by soil_condition relation: soil condition is essential to the growth of the plant. Now we have 7 meaningful relationships that meets the project requirement.
- Changes in order logic: After discussion, we convinced our client to adopt a more logic way of managing orders: each order will have some plant and suppliers. Not all suppliers have orders. Therefore, orders have total participation. Plants and suppliers have partial participation.
- A few name changes to increase readability:
    - batch tags -> care_notes, indicates the actions to be taken on the next garden visit.
    - Plant_event – observation: observation_ note: Half of the buds have blossomed.

Changes not implemented:

- ISA Relationship Constraints - "missing constraints". Our client refers to Bell peppers, onions, etc., as plants. Cultivars, such as "California wonder", "Orange Sun", are sub variety of Bell peppers. Some cultivars have better yield than others, we are collecting cultivar-specific growth data. However, some plants, such as Lavender, do not have sub-varieties, it's logical that super class, plant, have a partial participation. Since we only have one subgroup under plant, we don't have concerns for overlapping constraints. This is similar to the “visiting student is a student” relation in the class keynotes.

4. The schema derived from your ER diagram (above).

Cultivar ISA Plant

Plant(plant_ID: INTEGER(PK), expected_yield_weight: INTEGER, yield_type: VARCHAR, common_name: VARCHAR, scientific_name: VARCHAR)

Cultivar(**plant_ID** :INTEGER(PK,FK), overview_notes: VARCHAR, tags: VARCHAR)

Plant is_on order

Order(order_ID: INTEGER(PK), order_date: DATE, order_price: INTEGER, order_quantity: INTEGER)

Plant_on_Order(**plant_id:**INTEGER(PK,FK), **order_id:**INTEGER(PK,FK))

*We need assertion to guarantee the total participation constraint on Order in this relationship.

Supplier receives order

Supplier(supplier_ID: INTEGER(PK), supplier_name: VARCHAR, supplier_address: VARCHAR, supplier_tel: VARCHAR(CK))

Supplier_receives_Order (**supplier_ID:** INTEGER(PK,FK), **order_ID:** INTEGER(PK,FK))

*We need assertion to guarantee the total participation constraint on Order in this relationship

Plant goes_through Stage:

Stage (**plant_ID**: INTEGER(PK,FK), stage_name: VARCHAR(PK), start_date:DATE, end_date:DATE)

Batch is_at Stage:

Batch(batch_ID: INTEGER(PK), current_stage: VARCHAR, care_notes: VARCHAR, plant_date: DATE, **stage_name:** VARCHAR(FK))

Location distinguished_by Soil_condition:

Location(field_name: VARCHAR(PK), zone_id: INTEGER(PK), is_outdoor: BOOLEAN, is_irrigated: BOOLEAN)

Soil_condition(soil_type: VARCHAR()PK, pH: INTEGER, organic_matter_concentration: VARCHAR)

distinguished_by(**soil_type**: VARCHAR(PK,FK), **field_name**: VARCHAR(PK,FK), **zone_ID**: INTEGER(PK,FK))

*We need assertion to guarantee the total participation constraint on location in this relationship

Batch goes_to Location

Batch_goes_to_location(**batch_id**: INTEGER(PK,FK), **field_name**: VARCHAR(PK,FK), **zone_id**: INTEGER(PK,FK))

*We need assertion to guarantee the total participation constraint on Batch in this relationship

Order includes Batch

Order_includes_batch (**order_id**: INTEGER, **batch_id**: INTEGER)

User records Plant_event and batch

User (user_id : INTEGER(PK), user_name: VARCHAR, note: VARCHAR)

Plant_event (event_id : INTEGER(PK), event_name: VARCHAR, date: DATE, instruction: VARCHAR, observation: VARCHAR)

records (**user_id**: INTEGER (PK,FK), **event_id**: INTEGER(PK,FK), **batch_id**: INTEGER(PK,FK))

5. Functional Dependencies (FDs)

plant_ID -> overview_notes, tags

plant_ID, stage_name -> start_date, end_date

plant_ID -> expected_yield_weight, yield_type, common_name, scientific_name

scientific_name -> common_name (non-PK/CK FD) supplier_ID-> supplier_name, supplier_address

supplier_tel(CK)->supplier_address, supplier_ID

batch_ID -> current_stage, care_notes, plant_date

current_stage -> stage_name (non-PK/CK FD)

field_name, zone_id -> is_outdoor, is_irrigated

is_outdoor -> is_irrigated (non-PK/CK FD)

soil_type -> pH, organic_matter_concentration

order_id -> order_date, order_price, order_quantity

user_id -> user_name, user_note

6. Normalization

Plant(plant_ID: INTEGER(PK), expected_yield_weight: INTEGER, yield_type: VARCHAR, common_name: VARCHAR, scientific_name: VARCHAR)

The FD scientific_name -> common_name violates BCNF, in that scientific_name is not a key for this relation. We decide to decompose it to BCNF: scientific_name -> common_name

Plant1(scientific_name(PK)， common_name,)

Plant2(plant_ID(PK), expected_yield_weight, yield_type, **scientific_name(FK)**)

Now these relationships are in BCNF, and we do not need to further decompose.

Batch(batch_ID(PK), current_stage, care_notes, plant_date, **stage_name(FK)**)

The FD current_stage -> stage_name violates BCNF, in that current_stage is not a super key for Batch. We decide to decompose it to BCNF.

First round of decomposition: current_stage -> stage_name

Batch1(current_stage(PK), stage_name)

Batch2(batch_ID(PK), **current_stage(FK)**, care_notes, plant_date)

The two relations are both in BCNF, so we do not need to further decompose.

Location(field_name(PK), zone_id(PK), is_outdoor, is_irrigated)

The FD is_outdoor -> is_irrigated violates BCNF, in that is_outdoor is not a super key for Location. We decide to decompose it to BCNF.

First round of decomposition: is_outdoor -> is_irrigated

Location1 (is_outdoor(PK), is_irrigated)
Location2 (field_name, zone_id, **is_outdoor(FK)**)
The two relations are both in BCNF, so we do not need to further decompose.

The following relations are already in BCNF and 3NF:

Cultivar(**plant_ID** :INTEGER(PK,FK), overview_notes: VARCHAR, tags: VARCHAR)

Order(order_lD(PK): INTEGER, order_date: DATE, order_price: INTEGER, order_quantity: INTEGER)

Plant_on_Order(**plant_id**(PK,FK), **order_id**(PK,FK))

Supplier(supplier_ID: INTEGER(PK), supplier_name: VARCHAR, supplier_address: VARCHAR, supplier_tel: VARCHAR(CK))

Supplier_receives_Order (**supplier_ID**(PK,FK), **order_lD**(PK,FK))

Batch_goes_to_location--- (**batch_id**: INTEGER(PK,FK)**,** **field_name**: VARCHAR(PK,FK)**,** **zone_id**: INTEGER(PK,FK))

Order_includes_batch (**order_id**: INTEGER(PK,FK), **batch_id**: INTEGER(PK,FK))

Stage (**plant_ID**: INTEGER(PK,FK), stage_name: VARCHAR(PK), start_date:DATE, end_date:DATE)

Soil_condition(soil_type: VARCHAR(PK), pH: INTEGER, organic_matter_concentration: VARCHAR)

distinguished_by(**soil_type**: VARCHAR(PK,FK), **field_name**: VARCHAR(PK,FK), **zone_ID**: INTEGER(PK,FK))

7. The SQL DDL statements required to create all the tables from item #6.

```
CREATE TABLE Plant1 (

        common_name VARCHAR,

        scientific_name VARCHAR,

        PRIMARY KEY (scientific_name)
```

```
);
CREATE TABLE Plant2 (
        plant_ID INTEGER PRIMARY KEY,
        expected_yield_weight INTEGER,
        yield_type VARCHAR,
        scientific_name VARCHAR,
        FOREIGN KEY(scientific_name) REFERENCES Plant1 (scientific_name) ON DELETE CASCADE
);
CREATE TABLE Cultivar (
        plant_ID INTEGER,
        overview_notes VARCHAR,
        tags VARCHAR,
        PRIMARY KEY (plant_ID),
        FOREIGN KEY(plant_ID) REFERENCES Plant2 (plant_ID) ON DELETE CASCADE
);
CREATE TABLE Order (
   order_ID int PRIMARY KEY,
   order_date DATE,
   order_price INTEGER,
   order_quantity INTEGER
);
CREATE TABLE  Plant_is_on_Order(
plant_id INTEGER,
order_id INTEGER,
PRIMARY KEY (plant_id, order_lD),
        FOREIGN KEY(plant_id) REFERENCES Plant2 (plant_id) ON DELETE CASCADE,
        FOREIGN KEY(order_id) REFERENCES Order (order_id) ON DELETE CASCADE
) ;
CREATE TABLE Supplier (
    supplier_ID int PRIMARY KEY,
    supplier_name VARCHAR,
    supplier_address VARCHAR,
    supplier_tel VARCHAR UNlQUE
);
CREATE TABLE Supplier_receives_Order (
    supplier_ID INTEGER,
    order_ID INTEGER,
    PRIMARY KEY (supplier_ID, order_ID),
    FOREIGN KEY (supplier_ID) REFERENCES Supplier(supplier_ID) ON DELETE
CASCADE,
    FOREIGN KEY (order_ID) REFERENCES Order(order_ID) ON DELETE CASCADE
);

CREATE TABLE Stage (
    plant_ID INTEGER,
    stage_name VARCHAR,
    start_date DATE,
    end_date DATE,
    PRIMARY KEY (plant_ID, stage_name),
    FOREIGN KEY (plant_ID) REFERENCES Plant2 (plant_ID) ON DELETE CASCADE
);
CREATE TABLE Batch1(
current_stage VARCHAR,
stage_name VARCHAR,
PRIMARY KEY(current_stage),
        FOREIGN KEY(stage_name) REFERENCES Stage (stage_name) ON DELETE CASCADE
);

CREATE TABLE Batch2 (
        batch_id INTEGER,
        current_stage VARCHAR,
        care_notes VARCHAR,
        plant_date DATE,
        PRIMARY KEY (batch_id)，
        FOREIGN KEY(current_stage) REFERENCES Batch1 (current_stage) ON DELETE CASCADE


);

CREATE TABLE Location1 (
is_outdoor BOOLEAN,
        is_irrigated BOOLEAN,
        PRIMARY KEY (is_outdoor)
);

CREATE TABLE Location2 (
field_name VARCHAR,
zone_id INTEGER,
is_outdoor BOOLEAN,
PRIMARY KEY (field_name, Zone_id )，
        FOREIGN KEY(is_outdoor) REFERENCES Location2 (is_outdoor),
);

CREATE TABLE Soil_condition (
   soil_type VARCHAR PRIMARY KEY,
   pH INTEGER,
   organic_matter_concentration VARCHAR
);

CREATE TABLE distinguished_by (
   soil_type VARCHAR,
   field_name VARCHAR,
   zone_ID INTEGER,
   PRIMARY KEY (soil_type, field_name, zone_ID),
   FOREIGN KEY (soil_type) REFERENCES Soil_condition (soil_type) ON DELETE CASCADE,
   FOREIGN KEY (field_name, zone_ID) REFERENCES Location2 (field_name, zone_id) ON DELETE CASCADE
);
```

8. INSERT statements to populate each table with at least 5 tuples.

```
INSERT INTO Plant1 (common_name, scientific_name) VALUES
('Rose', 'Rosa rubiginosa'),
('Lily', 'Lilium candidum'),
('Sunflower', 'Helianthus annuus'),
('Tulip', 'Tulipa gesneriana'),
('Daffodil', 'Narcissus poeticus');

INSERT INTO Plant2 (plant_ID, expected_yield_weight, yield_type, scientific_name) VALUES
(1, 200, 'Flowers', 'Rosa rubiginosa'),
(2, 150, 'Flowers', 'Lilium candidum'),
(3, 300, 'Seeds', 'Helianthus annuus'),
(4, 180, 'Flowers', 'Tulipa gesneriana'),
(5, 220, 'Bulbs', 'Narcissus poeticus');

INSERT INTO Cultivar (plant_ID, overview_notes, tags) VALUES
(1, 'Thorny shrub, scented flowers', 'garden, scented'),
(2, 'Classic white lily, very fragrant', 'ornamental, fragrant'),
(3, 'Tall, yellow flowers, edible seeds', 'field, edible'),
(4, 'Spring-blooming, colorful flowers', 'garden, spring'),
(5, 'Bright yellow flowers, blooms in early spring', 'spring, bulbs');

INSERT INTO "Order" (order_ID, order_date, order_price, order_quantity) VALUES
(101, '2023-10-01', 500, 10),
(102, '2023-10-02', 700, 15),
(103, '2023-10-03', 450, 8),
(104, '2023-10-04', 600, 12),
(105, '2023-10-05', 800, 20);

INSERT INTO Plant_is_on_Order (plant_id, order_id) VALUES
(1, 101),
(2, 102),
(3, 103),
(4, 104),
(5, 105);

INSERT INTO Supplier (supplier_ID, supplier_name, supplier_address, supplier_tel) VALUES
(201, 'Green Suppliers', '123 Green St.', '123-456-7890'),
(202, 'FlowerMart', '456 Floral Rd.', '987-654-3210'),
(203, 'BotanicCorp', '789 Garden Ave.', '456-789-1234'),
(204, 'AgriGoods', '101 Field Blvd.', '321-654-9870'),
(205, 'PlantWorld', '303 Nature Ln.', '789-123-4567');

INSERT INTO Supplier_receives_Order (supplier_ID, order_ID) VALUES
(201, 101),
(202, 102),
(203, 103),
(204, 104),
(205, 105);

INSERT INTO Stage (plant_ID, stage_name, start_date, end_date) VALUES
(1, 'Germination', '2023-01-01', '2023-01-10'),
(2, 'Flowering', '2023-02-01', '2023-02-15'),
(3, 'Seed Production', '2023-03-01', '2023-03-20'),
(4, 'Harvesting', '2023-04-01', '2023-04-10'),
(5, 'Bulb Growth', '2023-05-01', '2023-05-20');

INSERT INTO Batch1 (current_stage, stage_name) VALUES
('Germination', 'Germination'),
('Flowering', 'Flowering'),
('Seed Production', 'Seed Production'),
('Harvesting', 'Harvesting'),
('Bulb Growth', 'Bulb Growth');

INSERT INTO Batch2 (batch_id, current_stage, care_notes, plant_date) VALUES
(301, 'Germination', 'Keep moist and warm', '2023-01-02'),
(302, 'Flowering', 'Maintain sunlight', '2023-02-05'),
(303, 'Seed Production', 'Harvest seeds', '2023-03-10'),
(304, 'Harvesting', 'Cut flowers', '2023-04-05'),
(305, 'Bulb Growth', 'Maintain soil moisture', '2023-05-10');

INSERT INTO Location1 (is_outdoor, is_irrigated) VALUES
(TRUE, FALSE),
(FALSE, TRUE);
```

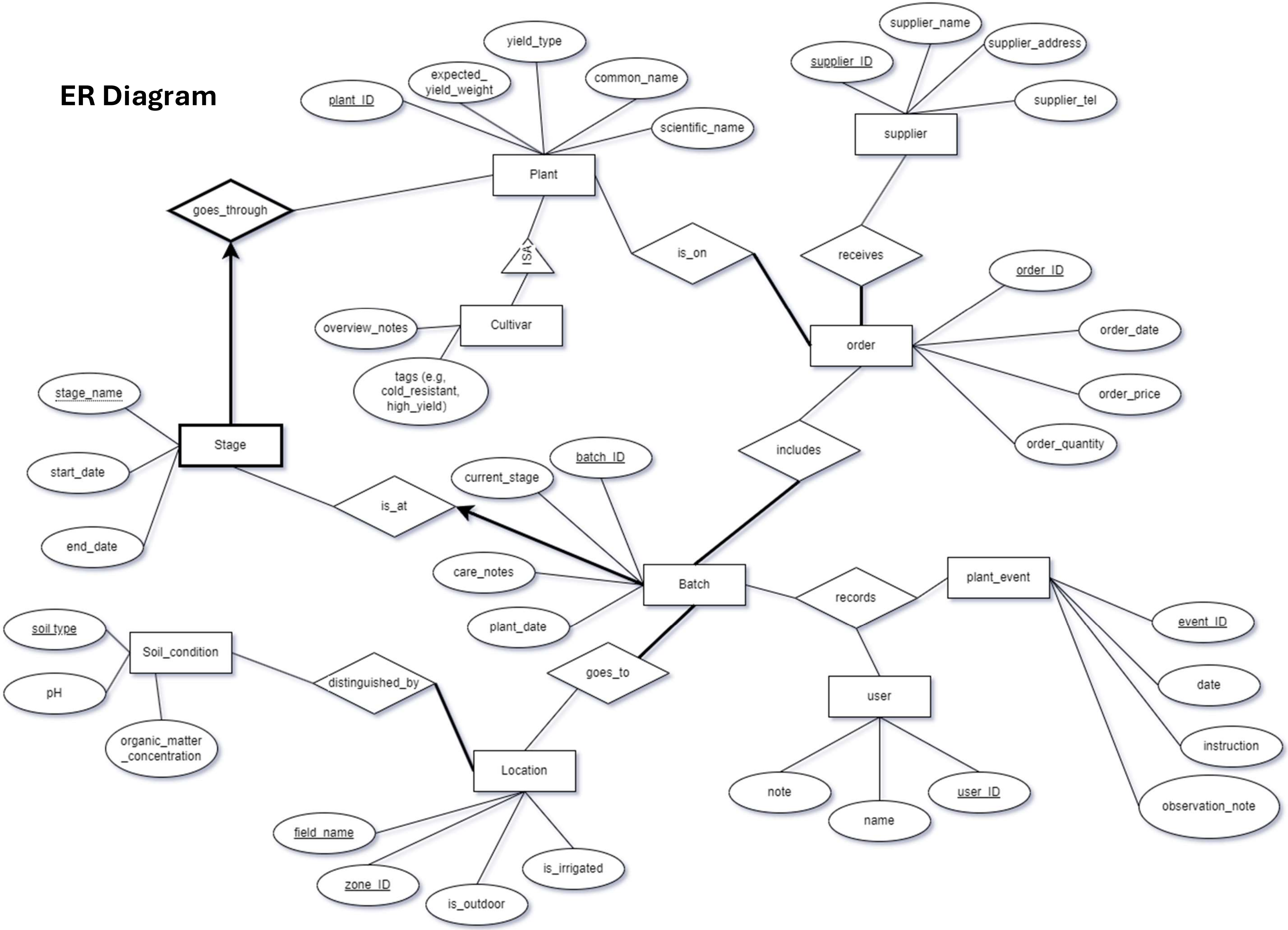
(It is intentional to only have 2 tuples, because outdoor fields are not irrigated, vice versa.)

```
INSERT INTO Location2 (field_name, zone_id, is_outdoor) VALUES
('Field A', 1, TRUE),
('Field B', 2, TRUE),
('Greenhouse 1', 3, FALSE),
('Field C', 4, TRUE),
('Field D', 5, TRUE);

INSERT INTO Soil_condition (soil_type, pH, organic_matter_concentration) VALUES
('Loam', 6, 'High'),
('Sandy', 5, 'Low'),
('Clay', 7, 'Medium'),
('Peaty', 6, 'High'),
('Silty', 6, 'Medium');

INSERT INTO distinguished_by (soil_type, field_name, zone_ID) VALUES
('Loam', 'Field A', 1),
('Sandy', 'Field B', 2),
('Clay', 'Greenhouse 1', 3),
('Peaty', 'Field C', 4),
('Silty', 'Field D', 5);
```

ER Diagram


Plant
plant_ID
expected_
yield_weight
yield_type
common_name
scientific_name
goes_through
ISA
Cultivar
overview_notes
tags (e.g, cold_resistant, high_yield)
supplier
supplier_ID
supplier_name
supplier_address
supplier_tel
receives
is_on
order
order_ID
order_date
order_price
order_quantity
includes
Stage
stage_name
start_date
end_date
is_at
Batch
current_stage
batch_ID
care_notes
plant_date
records
plant_event
event_ID
date
instruction
observation_note
user
note
name
user_ID
goes_to
Soil_condition
soil type
pH
organic_matter
_concentration
distinguished_by
Location
field_name
zone_ID
is_outdoor
is_irrigated