The Green Thumb

Creator: Ivana Louis

CSCI 44300 Database Systems
Kelly Van Busum

Enterprise Statement:

The Green Thumb company is an online platform that allows users to create an account to view specific information on growing vegetables in Indiana. Each user has a unique user ID, a user name, a password and the city where they will be gardening recorded. Each location consists of the city, first and last frost date and the length of the growing season. A location can grow many different vegetables and vegetables can be grown in many different locations. The information that can be found on vegetables are the vegetable ID, the type of plant, the variety, whether or not it can be started in doors, when the plant should be started, the amount of sun that is needed for the plant, and the harvest date. Many vegetables belong to a single plant family. The plant family lists the type of plant and the family the plant is in. Plants can have many issues, however the two main issues are diseases and critters. Vegetables can have at most one critter messing with the plant. Critters are listed by critter ID, the treatment involved, the type of critter and what plants can be affected by that specific critter. The same goes for diseases, vegetables can have at most one disease. Diseases are listed by disease ID, the treatment involved, the type of disease and what plants can be affected by that specific disease. The disease cannot exist with out the host plant. The user can access all the information except for the information on other users.

ERD:

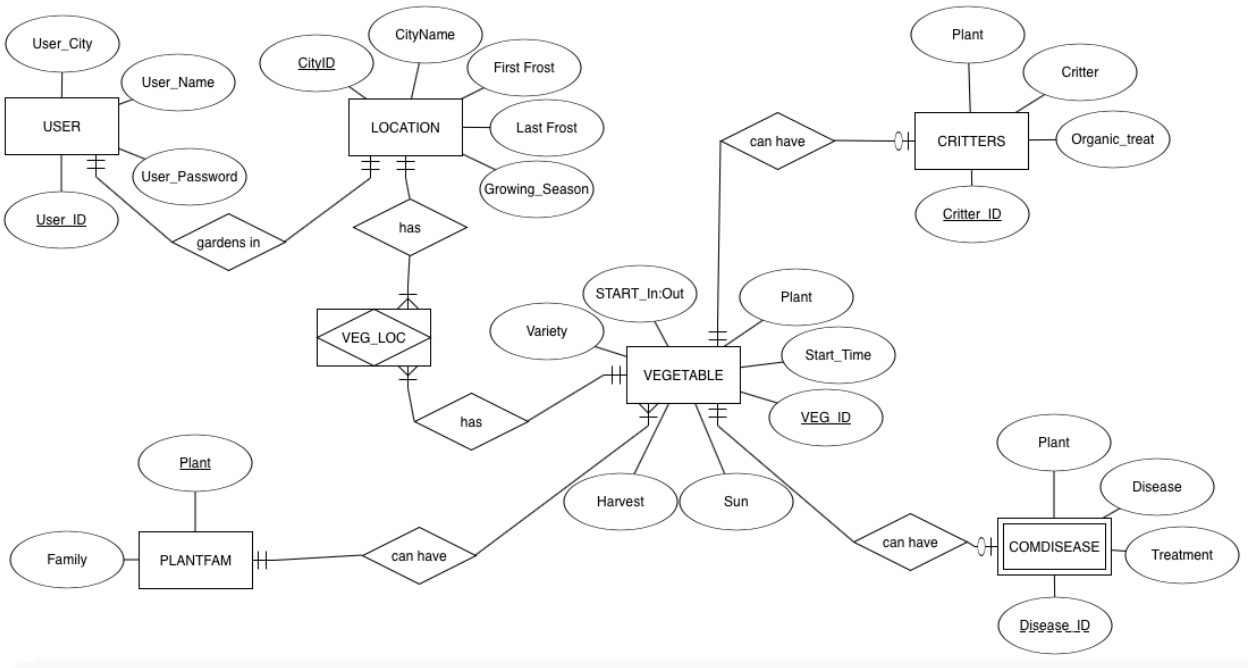

Figure 1: *The above figure is the entity relationship diagram for The Green Thumb database. This diagram is a visual representation of the enterprise statement.*

Relational Schema:

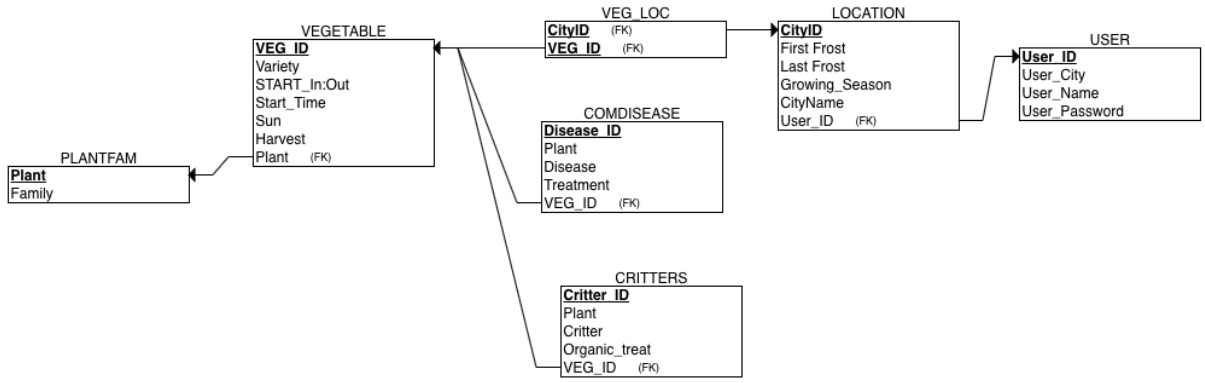

Figure 2: *The figure above is the relational schema based off the entity relationship diagram created for this database.*

Dependency Diagrams:

USER:

User_ID	User_City	User_Name	User_Password

LOCATION:

CityID	City	User_ID	First_Frost	Last_Frost	Growing_Season

VEGETABLE:

VEG_ID	Plant	Variety	START_In:Out	Start_Time	Sun	Harvest

VEG_LOC:

City	**VEG_ID**

COMDISEASE:

Disease_ID	Disease	VEG_ID	Plant	Treatment

CRITTERS:

Critter_ID	Critter	VEG_ID	Plant	Treatment

PLANTFAM:

Plant	Family

Normalization:

The Green Thumb database is in third normal form. This means that all of the attributes of the relation are atomic in nature, there are no multivalued attributes, and all attributes are dependent on the primary key. When looking at the dependency diagrams, it clearly shows that there are no partial or transitive dependencies. These are important qualities of a database in third normal form.

10 SQL Queries:

Query 1:

```
308  -- This view shows the user the city they live in as well as their first frost date,
309  -- last frost date and their growing season.
310  -- A FULL OUTER JOIN WAS USED HERE.
311  create or replace view USER_INFO as
312  select GARDENUSER.USERNAME, GARDENUSER.USERCITY, GARDLOCATION.FIRSTFROST,
313          GARDLOCATION.LASTFROST, GARDLOCATION.GROWINGSEASON
314  from GARDENUSER
315  full outer join GARDLOCATION on GARDENUSER.USERCITY = GARDLOCATION.CITY
316  where GARDENUSER.USERCITY = GARDLOCATION.CITY
317  order by GARDENUSER.USERCITY
318  ;
319
320  savepoint F;
```

	USERNAME	USERCITY	FIRSTFROST	LASTFROST	GROWINGSEASON
1	GreenThumb69	Alexandria	12-OCT-21	28-APR-21	166
2	GreenThumb21	Alexandria	12-OCT-21	28-APR-21	166
3	GreenThumb70	Anderson	12-OCT-21	28-APR-21	166
4	GreenThumb22	Anderson	12-OCT-21	28-APR-21	166
5	Flowergirl27	Anderson	12-OCT-21	28-APR-21	166
6	greenthumb	Anderson	12-OCT-21	28-APR-21	166
7	GreenThumb23	Angola	03-OCT-21	08-MAY-21	147
8	GreenThumb71	Angola	03-OCT-21	08-MAY-21	147
9	GreenThumb24	Auburn	06-OCT-21	04-MAY-21	154
10	GreenThumb72	Auburn	06-OCT-21	04-MAY-21	154

Query 2:

```
--Finds and Rounds the average harvest date to a whole number
create or replace view PLANT_GROWTH as
select VARIETY, PLANT, MINHARVEST, MAXHARVEST, round((MINHARVEST + MAXHARVEST)/ 2) as AVERAGE_HARVEST
from VEGETABLE;
```

	VARIETY	PLANT	MINHARVEST	MAXHARVEST	AVERAGE_HARVEST
1	Better Boy	Tomato	70	75	73
2	Cherokee Purple	Tomato	80	90	85
3	Early Girl	Tomato	50	50	50
4	Mr. Stripey	Tomato	80	80	80
5	Sweet Million	Tomato	60	62	61
6	Banana Hot	Peppers	75	75	75
7	Banana Sweet	Peppers	75	75	75
8	Habanero	Peppers	95	95	95
9	Jalapeno Hot	Peppers	72	72	72
10	Bell Pepper	Peppers	70	80	75

Query 3:

```
-- Show number of months in the growing season.
select CITY, round(months_between(FIRSTFROST, LASTFROST), 1) as GROWING_MONTHS
from GARDLOCATION;
```

	CITY	GROWING_MONTHS
1	Alexandria	5.5
2	Anderson	5.5
3	Angola	4.8
4	Auburn	5.1
5	Avon	6.1
6	Bedford	5.4
7	Beech Grove	5.6
8	Bloomington	5.8
9	Bluffton	5.5
10	Boonville	6

Query 4:

```
367  -- Changes the science name to the more common english term for users.
368  create or replace view PLANTFAM_ENG as
369  select PLANT, DECODE(FAMILY, 'Solanaceae - Nightshades', 'Nightshades', 'Fabaceae - Legumes', 'Legumes', 'Cucurbitaceae - Cucurbits', 'Cucurbits',
370                       'Brassiaceae - Brassicas', 'Brassicas',  FAMILY) FAMILY
371  from PLANTFAM
372  order by FAMILY;
```

	PLANT	FAMILY
1	Broccoli	Brassicaceae – Brassicas
2	Cabbage	Brassicaceae – Brassicas
3	Cauliflower	Brassicaceae – Brassicas
4	Cucumbers	Cucurbits
5	Melons	Cucurbits
6	Zucchini	Cucurbits
7	Pumpkins	Cucurbits
8	Beans	Legumes
9	Tomatillos	Nightshades
10	Potatoes	Nightshades

Query 5:

```
382  -- Shows the plants that are started indoor and have more than one variety.
383  select Plant, count(*) as Number_Of_Varieties
384  from VEGETABLE
385  where START_IN_OUT = 'Indoor'
386  group by plant
387  Having count(*) > 1
388  ;
```

	PLANT	NUMBER_OF_VARIETIES
1	Eggplant	2
2	Cucumbers	3
3	Tomato	5
4	Zucchini	4
5	Peppers	5
6	Melons	5

Query 6:

```
393  -- Shows plants that are planted at least 6 weeks before frost date.
394  select Plant, variety
395  from VEGETABLE
396  where PLANT in (select PLANT from VEGETABLE where STARTTIME like '6%');
```

	PLANT	VARIETY
1	Tomato	Better Boy
2	Tomato	Cherokee Purple
3	Tomato	Early Girl
4	Tomato	Mr. Stripey
5	Tomato	Sweet Million
6	EggPlant	Black Beauty
7	Eggplant	Ichiban
8	Eggplant	White Eggplant
9	Tomatillos	Tomatillo
10	Broccoli	Lieutenant
11	Cauliflower	White Hybrid

Query 7:

```
470  drop sequence passwordID;
471  create sequence passwordID increment by 137 start with 1000;
472
473  insert into GARDENUSER values(3, 'johnDeer', passwordID.NextVal, NULL, NULL);
```

66	3	johnDeer	1000	(null)	(null)

Query 8:

```
-- A user has been looking up information for a different location with the longest growing
-- season more than 10 times. Update their location based on the new location they are researching.
update GARDENUSER set UserCity = (select City from GARDLOCATION where GrowingSeason = 205)
where UserName = 'love2garden';
```

13	16	love2garden	19671995	8	Evansville

Query 9:

```
-- View Three
-- This View shows vegetable varieties based on grow time.
create or replace view PLANT_GROWTH as
select VARIETY, PLANT, MINHARVEST, MAXHARVEST
from VEGETABLE
;
```

	VARIETY	PLANT	MINHARVEST	MAXHARVEST
1	Better Boy	Tomato	70	75
2	Cherokee Purple	Tomato	80	90
3	Early Girl	Tomato	50	50
4	Mr. Stripey	Tomato	80	80
5	Sweet Million	Tomato	60	62
6	Banana Hot	Peppers	75	75
7	Banana Sweet	Peppers	75	75
8	Habanero	Peppers	95	95
9	Jalapeno Hot	Peppers	72	72
10	Bell Pepper	Peppers	70	80

Query 10:

```
-- View Four
-- This view can show both diseases and pests/animals that destroy plants
-- usuall pests and diseases go hand in hand so its nice to have them together.
-- THE JOIN REQUIREMENT WAS USED HERE
create or replace view PLANT_CONTROL as
select CRITTERS.PLANT, CRITTERS.CRITTER, CRITTERS.ORGANICTREAT as CRITTER_TREATMENT,
        COMDISEASE.DISEASE, COMDISEASE.TREATMENT as PLANT_TREATMENT
from CRITTERS
join COMDISEASE on CRITTERS.PLANT = COMDISEASE.PLANT
where CRITTERS.PLANT = COMDISEASE.PLANT
order by CRITTERS.PLANT
;
```

	PLANT	CRITTER	CRITTER_TREATMENT	DISEASE	PLANT_TREATMENT
1	Beans	Rabbits	Red Pepper Flakes	Fungus on Foliage	Fungicides
2	Broccoli	Cabbage Worms	Bacillus thuringiensis based organic insecticide	Sclerotinia Stem Rot	Plant cannot be saved - focus on clearing soil and rotate crops
3	Cabbage	Cabbage Loopers	Bacillus thuringiensis based organic insecticide	Club Root	Remove plants and raise soil pH to 7.2
4	Cauliflower	Cabbage Root Maggots	Bacillus thuringiensis based organic insecticide	Alterbaria leaf spot	Fungicides
5	Cucumbers	Cucumber Beetles	Insecticidal Soap Solution, Cornstartch, Neem Oil	Cucumber Mosaic	(null)
6	Eggplant	Flea Beetles	Neem Oil, Cornstarch	Alternaria Rot	Fungacides including captan or copper
7	Melons	Greenhouse Whitefly	Spray of with water, Insecticidal Soap Solution	Powdery Mildew	Fungicides including sulfur, lime-sulfur, neem oil, potassium bi
8	Peppers	Aphids	Spray off with water and insecticidal soap solution	Leaf Spot	Baking Soda, Dish soap, Water solution
9	Potatoes	Wireworm	Entomopathogenic Nematodes	Blight	Remove all affected leaves and burn them or place them in the ga
10	Tomato	Tomato Hornworm	Bacillus thuringiensis based organic insecticide	Verticillium Wilt	Plant cannot be saved - focus on clearing soil for next year
11	Zucchini	Japanese Beetles	Insecticidal Soap Solution, Cornstartch, Neem Oil	Bacterial Wilt	Plant cannot be saved - focus on clearing diseased pant before d

Extend Course Material

To extend this project, I decided to add in some PL/SQL queries. I used Oracle SQL developer to create this database. This made some of the SQL queries slightly different. However, I was more comfortable using this platform for PL/SQL. PL/SQL is Procedural Language extension to the Structured Query Language. It allows for the declaration of constants and variables, procedures, functions, triggers and more. I have included some examples of some of the PL/SQL code blocks used in this project.

Example 1:

```
-- This sequence is used in the code block to create an incrementing userID.
drop sequence createID;
create sequence createID increment by 1 start with 21;

-- Purpose: the purpose of this code block is to generate users so that the
-- database as a whole can work. Without users there's no reason for the database
-- to exist. The cursor retieves data from the location table and pops it into
-- the user table. Then the code block is used to process data and give values
-- to the created variables. It uses exception handling to ensure the proper use
-- of the cursor involved in this code block. This all is done in a procedure,
-- the procedure is called and the number of users you want to add is inserted
-- into the garden users table.
```

```
CREATE or REPLACE
Procedure addUsers(n IN number)
as
UserID          Number;
UserName        VARCHAR(100);
UserPassword    VARCHAR(50);

            cursor Location_cursor is
                select CityID, City from GARDLOCATION;
            Location_val Location_cursor%ROWTYPE;


BEGIN
    open Location_cursor;
    dbms_random.seed (val => 0);
    FOR i IN 1..n LOOP
        --open Location_cursor;
        fetch Location_cursor into Location_val;
        UserID := createID.NextVal;
        UserName := 'GreenThumb' || UserID;
        UserPassword := dbms_random.string('x',TRUNC(dbms_random.value(6,12)));
        INSERT INTO GARDENUSER VALUES(UserID ,UserName, UserPassword, Location_val.CityID, Location_val.City);
    END LOOP;
    close Location_cursor;
EXCEPTION
    WHEN INVALID_CURSOR THEN
        DBMS_OUTPUT.PUT_LINE('Cannot close an unopened cursor');
    WHEN CURSOR_ALREADY_OPEN THEN
        DBMS_OUTPUT.PUT_LINE('Must close cursor before reopening.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Non-Cursor related Error');
END;
/
```

20	20	username	123456789	5	Avon
21	21	GreenThumb21	T87D2G	1	Alexandria
22	22	GreenThumb22	XTG9HRTGC6X	2	Anderson
23	23	GreenThumb23	RP4M41	3	Angola
24	24	GreenThumb24	Y9P0VZ04	4	Auburn

Example 2:

-- These two tables were created to keep track of system activities. These tables do not serve an
-- actual purpose to the user, or the idea behind the database for this project, that is why they are
-- not included in the original ERD. However, if this was to be an actual system for a real
-- company these triggers would be extremely useful.

```
548  drop table VEGETABLE_LOG;
549  create table VEGETABLE_LOG as select VEG_ID, VARIETY, PLANT from VEGETABLE;
550  alter table VEGETABLE_LOG add Edit varchar2(15);
551  alter table VEGETABLE_LOG add MOD_USER varchar2(15);
552  alter table VEGETABLE_LOG add MOD_TIMESTAMP Date;
553  select * from VEGETABLE_LOG;
554  delete from VEGETABLE_LOG;
555
556  drop table USER_LOG;
557  create table USER_LOG as select * from GARDENUSER;
558  alter table USER_LOG add Edit varchar2(15);
559  alter table USER_LOG add MOD_USER varchar2(15);
560  alter table USER_LOG add MOD_TIMESTAMP Date;
561  select * from USER_LOG;
562  delete from USER_LOG;
```

```
553  -- Purpose: The purpose of these triggers are to keep track of any insertions
554  -- or deletions in the vegetable table, and any deletions in the user table.
555  -- the trigger puts the information into the appropriate table that was created.
556  -- Keeping track of these edits can prove to be useful for the database
557  -- for example when reinstating deleted accounts.
```

```
582  Create or replace trigger USER_DEL_ROW
583  after delete on GARDENUSER
584  for each row
585  begin
586  Insert into USER_LOG(UserID, UserName, UserPassword, UserCityID, UserCity, EDIT , MOD_USER, MOD_TIMESTAMP)
587  values (:old.UserID, :old.UserName, :old.UserPassword, :old.UserCityID, :old.UserCity, 'Delete',user ,Sysdate);
588  end; -- Note: When the new and old are referenced inside the begin/end block, must use colons (:)
589  /
```

```
574  Create or replace trigger VEG_INS_ROW
575  after insert on VEGETABLE
576  for each row
577  begin
578  Insert into VEGETABLE_LOG(VEG_ID, VARIETY, PLANT, EDIT, MOD_USER, MOD_TIMESTAMP)
579  values (:new.VEG_ID, :new.VARIETY, :new.PLANT,'Insert' ,user ,Sysdate);
580  end; -- Note: When the new and old are referenced inside the begin/end block, must use colons (:)
581  /
582
583  Create or replace trigger VEG_DEL_ROW
584  after delete on VEGETABLE
585  for each row
586  begin
587  Insert into VEGETABLE_LOG(VEG_ID, VARIETY, PLANT, EDIT ,MOD_USER, MOD_TIMESTAMP)
588  values (:OLD.VEG_ID, :OLD.VARIETY, :OLD.PLANT,'Delete', user ,Sysdate);
589  end; -- Note: When the new and old are referenced inside the begin/end block, must use colons (:)
590  /
```

```
601  -- EXAMPLE
602  SELECT * FROM VEGETABLE;
603  INSERT INTO VEGETABLE VALUES(32, 'Patio Star', 'Zucchini', 'Indoor', '4 to 6 Weeks Before Last Frost Date', 'Full Sun', 50, 50);
604  DELETE FROM VEGETABLE WHERE VEG_ID = 33;
605  INSERT INTO VEGETABLE VALUES(34, 'Caserta', 'Zucchini', 'Indoor', '4 to 6 Weeks Before Last Frost Date', 'Full Sun', 55, 55);
606
607  -- EXAMPLE
608  DELETE FROM GARDENUSER WHERE UserCityID = 44;
609  DELETE FROM GARDENUSER WHERE UserCityID = 45;
610  DELETE FROM GARDENUSER WHERE UserCityID = 46;
611
612  select * from VEGETABLE_LOG;
613  select * from USER_LOG;
614
```

	VEG_ID	VARIETY	PLANT	EDIT	MOD_USER	MOD_TIMESTAMP
1	32	Patio Star	Zucchini	Insert	ILOUIS	04-MAY-21
2	33	Bush Baby	Zucchini	Insert	ILOUIS	04-MAY-21
3	34	Caserta	Zucchini	Insert	ILOUIS	04-MAY-21
4	33	Bush Baby	Zucchini	Delete	ILOUIS	04-MAY-21

	USERID	USERNAME	USERPASSWORD	USERCITYID	USERCITY	EDIT	MOD_USER	MOD_TIMESTAMP
1	67	GreenThumb67	DDIIM1UPS	47	Jasper	Delete	ILOUIS	04-MAY-21
2	64	GreenThumb64	RSK7DB9WX	44	Hobart	Delete	ILOUIS	04-MAY-21
3	65	GreenThumb65	K7NSIQ3F5S	45	Huntington	Delete	ILOUIS	04-MAY-21
4	3	leila129	harryStyles	46	Indianapolis	Delete	ILOUIS	04-MAY-21
5	7	ivanabananas	nana1995	46	Indianapolis	Delete	ILOUIS	04-MAY-21
6	8	voldemort	1234	46	Indianapolis	Delete	ILOUIS	04-MAY-21
7	66	GreenThumb66	7Y552UCMT5	46	Indianapolis	Delete	ILOUIS	04-MAY-21