Drone Delivery System Application

SOFE3700 – Project Phase II: Project Design

November 7, 2021

Final Project Group 24 Members

Usman Mahmood 100349839

Karanvir Bhogal 100748973

Daniel Grewal 100768376

Mohammed Adnan Hashmi 100753115

Part A: Relational Schema

Microsoft SQL create table commands for the WeDrone application:

```
CREATE DATABASE WeDrone;
GO
USE WeDrone;

CREATE TABLE Locations (
	LocationId INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
	LocationName VARCHAR(255) NULL,
	Latitude DECIMAL(17,14) NULL,
	Longitude DECIMAL(17,14) NULL,
	FullAddress VARCHAR(255) NULL,
	DroneFacility BIT NOT NULL
);

CREATE TABLE FlightLegs (
	FlightLegId INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
	FromLocation INT FOREIGN KEY REFERENCES Locations(LocationId) NOT
NULL,
	ToLocation INT FOREIGN KEY REFERENCES Locations(LocationId) NOT NULL,
	Distance DECIMAL(10,2) NOT NULL
);

CREATE TABLE FlightRoutes (
	RouteId INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
	RouteStart INT FOREIGN KEY REFERENCES Locations(LocationId) NOT NULL,
	RouteEnd INT FOREIGN KEY REFERENCES Locations(LocationId) NOT NULL,
);

CREATE TABLE FlightRouteSteps (
	RouteId INT FOREIGN KEY REFERENCES FlightRoutes(RouteId) NOT NULL,
	CurrentLeg INT FOREIGN KEY REFERENCES FlightLegs(FlightLegId) NOT
NULL,
	NextLeg INT FOREIGN KEY REFERENCES Flightlegs(FlightLegId) NULL,
	IsInitialLeg BIT NOT NULL,
	CONSTRAINT PK_RouteStep PRIMARY KEY (RouteId, CurrentLeg)
);

CREATE TABLE Users (
	UserId INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
	Username VARCHAR(50) NOT NULL UNIQUE,
```

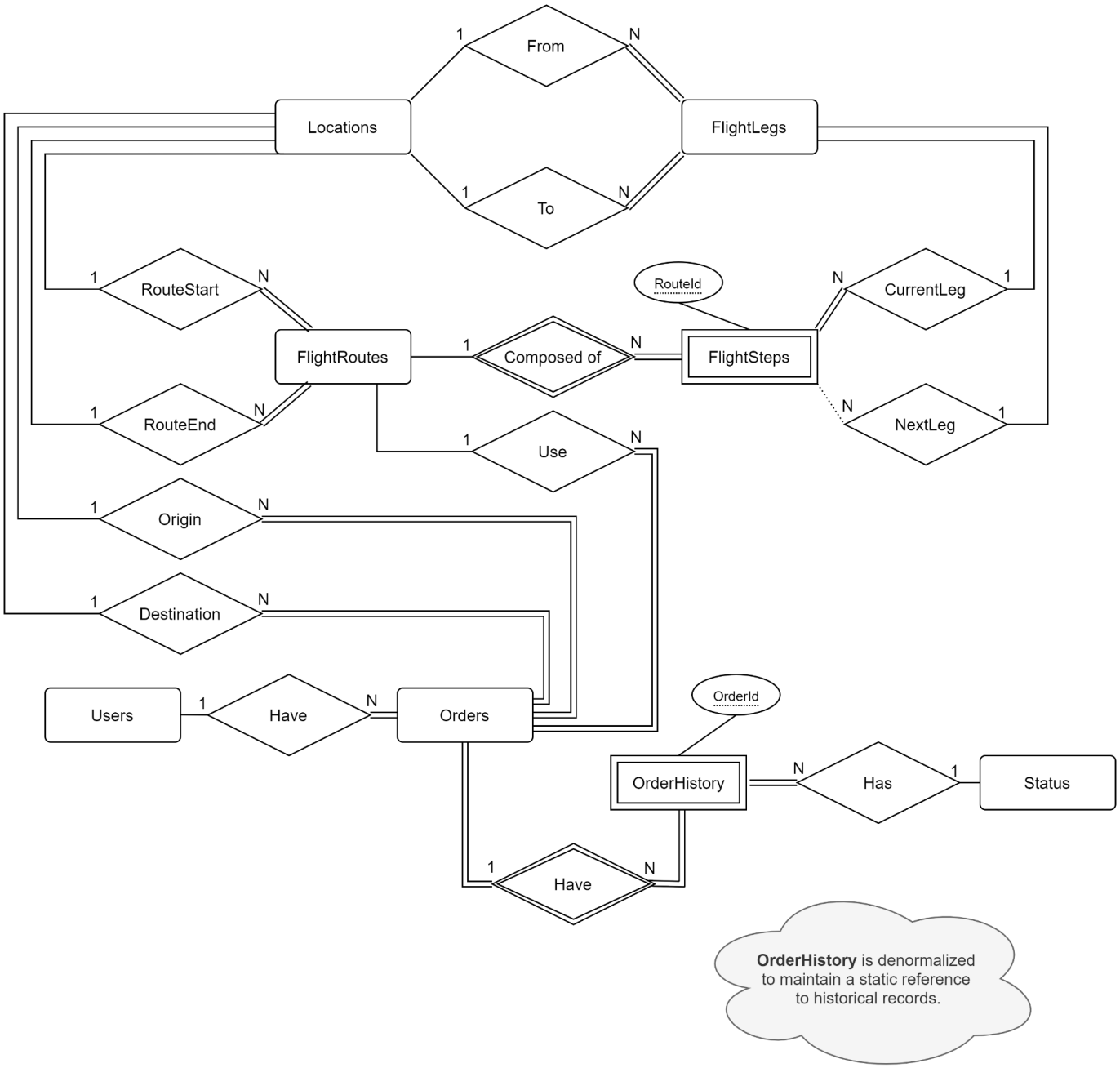
```
	Password VARCHAR(50) NOT NULL,
	FirstName VARCHAR(50) NOT NULL,
	LastName VARCHAR(50) NOT NULL
);

CREATE TABLE Status (
	StatusId INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
	StatusName VARCHAR(50) NOT NULL
);

CREATE TABLE Orders (
	OrderId INT IDENTITY(1,1) PRIMARY KEY NOT NULL,
	UserId INT FOREIGN KEY REFERENCES Users(UserId) NOT NULL,
	Origin INT FOREIGN KEY REFERENCES Locations(LocationId) NOT NULL,
	Destination INT FOREIGN KEY REFERENCES Locations(locationId) NOT
NULL,
	Weight DECIMAL(10,2) NOT NULL,
	Volume DECIMAL(10,2) NOT NULL,
	OrderCreated DATETIME2(7) NOT NULL,
	OrderFilled DATETIME2(7) NULL
);

-- Modelled as a Type 2 Slowly Changing Dimension (SCD)
CREATE TABLE OrderHistory (
	OrderId INT FOREIGN KEY REFERENCES Orders(OrderId) NOT NULL,
	ValidFrom DATETIME2(7) NOT NULL,
	ValidTo DATETIME2(7) NOT NULL,
	TravelingFrom VARCHAR(255) NULL, --denormalized to maintain static
historical record
	TravelingTo VARCHAR(255) NULL, --denormalized to maintain static
historical record
	Distance DECIMAL(10,2) NULL,
	OrderStatus INT FOREIGN KEY REFERENCES Status(StatusId) NOT NULL,
	CONSTRAINT PK_OrderHistory PRIMARY KEY (OrderId, ValidFrom)
);
```

Relational Diagram for the WeDrone Delivery system application:



Note: **OrderHistory** is denormalized to maintain a static reference to historical records.

Part B: Sample Data

Microsoft SQL insert commands for populating sample data into the WeDrone database:

```
INSERT INTO Locations VALUES
('Toronto Pearson International Airport', 43.68232877980147,
-79.62661047567958, '6301 Silver Dart Dr, Mississauga, ON L5P 1B2', 1), --1
('Square One Shopping Centre', 43.5932704575367, -79.6418354765629, '100
City Centre Dr, Mississauga, ON L5B 2C9', 1),
('Vaughan Mills Mall', 43.82549302652521, -79.5381447146669, '1 Bass Pro
Mills Dr, Vaughan, ON L4K 5W4', 1),
('CN Tower', 43.64272145936629, -79.38704607234244, '290 Bremner Blvd,
Toronto, ON M5V 3L9', 1),
('Ontario Science Centre', 43.71718851953277, -79.33851344772478, '770 Don
Mills Rd., North York, ON M3C 1T3', 1),
('Toronto Zoo', 43.82096417612802, -79.1812287514316, '2000 Meadowvale Rd,
Toronto, ON M1B 5K7', 1),
('Ontario Technology University', 43.94565647325994, -78.89679613001036,
'2000 Simcoe St N, Oshawa, ON L1G 0C5', 1),
('Lowes Brampton', NULL, NULL, '370 Kennedy Rd S, Brampton, ON L6W 4V2',
0),
('Lowes Whitby', NULL, NULL, '4005 Garrard Rd, Whitby, ON L1R 0J1', 0);


INSERT INTO FlightLegs VALUES
(1, 2, 9.98), --1
(2, 1, 9.98), --2
(1, 3, 17.43), --3
(3, 1, 17.43), --4
(1, 4, 19.77), --5
(4, 1, 19.77), --6
(1, 5, 23.48), --7
(5, 1, 23.48), --8
(1, 6, 38.95), --9
(6, 1, 38.95), --10
(1, 7, 65.47), --11
(7, 1, 65.47), --12
(2, 3, 27.13), --13
(3, 2, 27.13), --14
(2, 4, 21.24), --15
(4, 2, 21.24), --16
(2, 5, 28.02), --17
(5, 2, 28.02), --18
```

```
(2, 6, 44.85), --19
(6, 2, 44.85), --20
(2, 7, 71.51), --21
(7, 2, 71.51), --22
(3, 4, 23.67), --23
(4, 3, 23.67), --24
(3, 5, 20.05), --25
(5, 3, 20.05), --26
(3, 6, 28.64), --27
(6, 3, 28.64), --28
(3, 7, 53.11), --29
(7, 3, 53.11), --30
(4, 5, 9.15), --31
(5, 4, 9.15), --32
(4, 6, 25.81), --33
(6, 4, 25.81), --34
(4, 7, 51.80), --35
(7, 4, 51.80), --36
(5, 6, 17.11), --37
(6, 5, 17.11), --38
(5, 7, 43.60), --39
(7, 5, 43.60), --40
(6, 7, 26.68), --41
(7, 6, 26.68) --42

INSERT INTO FlightRoutes VALUES
(1, 2), --1
(2, 1), --2
(1, 3), --3
(3, 1), --4
(1, 4), --5
(4, 1), --6
(1, 5), --7
(5, 1), --8
(1, 6), --9
(6, 1), --10
(1, 7), --11
(7, 1), --12
(2, 3), --13
(3, 2), --14
(2, 4), --15
(4, 2), --16
(2, 5), --17
```

```
(5, 2), --18
(2, 6), --19
(6, 2), --20
(2, 7), --21
(7, 2), --22
(3, 4), --23
(4, 3), --24
(3, 5), --25
(5, 3), --26
(3, 6), --27
(6, 3), --28
(3, 7), --29
(7, 3), --30
(4, 5), --31
(5, 4), --32
(4, 6), --33
(6, 4), --34
(4, 7), --35
(7, 4), --36
(5, 6), --37
(6, 5), --38
(5, 7), --39
(7, 5), --40
(6, 7), --41
(7, 6) --42

INSERT INTO FlightRouteSteps VALUES
-- ROUTE 1 (1 -> 2)
(1, 1, NULL, 1),
-- ROUTE 2 (2 -> 1)
(2, 2, NULL, 1),
-- ROUTE 3 (1 -> 3)
(3, 3, NULL, 1),
-- ROUTE 4 (3 -> 1)
(4, 4, NULL, 1),
-- ROUTE 5 (1 -> 4)
(5, 5, NULL, 1),
-- ROUTE 6 (4 -> 1)
(6, 6, NULL, 1),
-- ROUTE 7 (1 -> 5)
(7, 7, NULL, 1),
-- ROUTE 8 (5 -> 1)
(8, 8, NULL, 1),
```

```
-- ROUTE 9 (1 -> 6)
(9, 7, 37, 1),
(9, 37, NULL, 0),
-- ROUTE 10 (6 -> 1)
(10, 38, 8, 1),
(10, 8, NULL, 0),
-- ROUTE 11 (1 -> 7)
(11, 7, 37, 1),
(11, 37, 41, 0),
(11, 41, NULL, 0),
-- ROUTE 12 (7 -> 1)
(12, 42, 38, 1),
(12, 38, 8, 0),
(12, 8, NULL, 0),
--ROUTE 13 (2 -> 3)
(13, 2, 3, 1),
(13, 3, NULL, 0),
--ROUTE 14 (3 -> 2)
(14, 4, 1, 1),
(14, 1, NULL, 0),
--ROUTE 15 (2 -> 4)
(15, 15, NULL, 1),
--ROUTE 16 (4 -> 2)
(16, 16, NULL, 1),
--ROUTE 17 (2 -> 5)
(17, 15, 31, 1),
(17, 31, NULL, 0),
--ROUTE 18 (5 -> 2)
(18, 32, 16, 1),
(18, 16, NULL, 0),
--ROUTE 19 (2 -> 6)
(19, 15, 31, 1),
(19, 31, 37, 0),
(19, 37, NULL, 0),
--ROUTE 20 (6 -> 2)
(20, 38, 32, 1),
(20, 32, 16, 0),
(20, 16, NULL, 0),
--ROUTE 21 (2 -> 7)
(21, 15, 31, 1),
(21, 31, 37, 0),
(21, 37, 41, 0),
(21, 41, NULL, 0),
```

```
--ROUTE 22 (7 -> 2)
(22, 42, 38, 1),
(22, 38, 32, 0),
(22, 32, 16, 0),
(22, 16, NULL, 0),
--ROUTE 23 (3 -> 4)
(23, 7, NULL, 1),
--ROUTE 24 (4 -> 3)
(24, 8, NULL, 1),
--ROUTE 25 (3 -> 5)
(25, 25, NULL, 1),
--ROUTE 26 (5 -> 3)
(26, 26, NULL, 1),
--ROUTE 27 (3 -> 6)
(27, 27, NULL, 1),
--ROUTE 28 (6 -> 3)
(28, 28, NULL, 1),
--ROUTE 29 (3 -> 7)
(29, 27, 41, 1),
(29, 41, NULL, 0),
--ROUTE 30 (7 -> 3)
(30, 42, 28, 1),
(30, 28, NULL, 0),
--ROUTE 31 (4 -> 5)
(31, 31, NULL, 1),
--ROUTE 32 (5 -> 4)
(32, 32, NULL, 1),
--ROUTE 33 (4 -> 6)
(33, 31, 37, 1),
(33, 37, NULL, 0),
--ROUTE 34 (6 -> 4)
(34, 38, 32, 1),
(34, 32, NULL, 0),
--ROUTE 35 (4 -> 7)
(35, 31, 37, 1),
(35, 37, 41, 0),
(35, 41, NULL, 0),
--ROUTE 36 (7 -> 4)
(36, 42, 38, 1),
(36, 38, 32, 0),
(36, 32, NULL, 0),
--ROUTE 37 (5 -> 6)
(37, 37, NULL, 1),
```

```
--ROUTE 38 (6 -> 5)
(38, 38, NULL, 1),
--ROUTE 39 (5 -> 7)
(39, 37, 41, 1),
(39, 41, NULL, 0),
--ROUTE 40 (7 -> 5)
(40, 42, 38, 1),
(40, 38, NULL, 0),
--ROUTE 41 (6 -> 7)
(41, 41, NULL, 1),
--ROUTE 42 (7 -> 6)
(42, 42, NULL, 1)

INSERT INTO Users VALUES
('usman', 'password', 'Usman', 'Mahmood'),
('daniel', 'password', 'Daniel', 'Grewal'),
('adnan', 'password', 'Mohammed Adnan', 'Hashmi'),
('karanvir', 'password', 'Karanvir', 'Bhogal');

INSERT INTO Status VALUES
('Pending Pick-up'), --1
('Order Retrieved'), --2
('In Transit'), --3
('Pending Drop-off'), --4
('Delivered'), --5
('Cancelled'); --6

INSERT INTO Orders VALUES
(1, 8, 9, 5.0, 12.0, '2021-11-01 8:25', '2021-11-01 9:30'),
(1, 9, 8, 6.0, 14.0, '2021-11-02 07:00', NULL),
(1, 8, 9, 10.0, 24.0, '2021-11-03 06:15', NULL),
(2, 8, 9, 3.0, 9.0, '2021-11-04 05:00', NULL);

INSERT INTO OrderHistory VALUES
-- Order 1: Delivered Order
(1, '2021-11-01 8:25', '2021-11-01 8:35', '6301 Silver Dart Dr,
Mississauga, ON L5P 1B2', '370 Kennedy Rd S, Brampton, ON L6W 4V2', 7.82,
1), -- 1 -> Pickup
(1, '2021-11-01 8:35', '2021-11-01 8:45', '370 Kennedy Rd S, Brampton, ON
L6W 4V2', '6301 Silver Dart Dr, Mississauga, ON L5P 1B2', 7.82, 2), --
Pickup -> 1
(1, '2021-11-01 8:45', '2021-11-01 9:00', '6301 Silver Dart Dr,
Mississauga, ON L5P 1B2', '770 Don Mills Rd., North York, ON M3C 1T3',
```

```
23.48, 3), -- 1 -> 5
(1, '2021-11-01 9:00', '2021-11-01 9:10', '770 Don Mills Rd., North York,
ON M3C 1T3', '2000 Meadowvale Rd, Toronto, ON M1B 5K7', 17.11, 3), -- 5 ->
6
(1, '2021-11-01 9:10', '2021-11-01 9:25', '2000 Meadowvale Rd, Toronto, ON
M1B 5K7', '2000 Simcoe St N, Oshawa, ON L1G 0C5', 26.68, 3), -- 6 -> 7
(1, '2021-11-01 9:25', '2021-11-01 9:30', '2000 Simcoe St N, Oshawa, ON L1G
0C5', '4005 Garrard Rd, Whitby, ON L1R 0J1', 2.75, 4), -- 7 -> Drop-off
(1, '2021-11-01 9:30', '9999-12-31', NULL, NULL, NULL, 5), -- Delivered
-- Order 2: Order In Transit
(2, '2021-11-02 7:00', '2021-11-02 7:10', '2000 Simcoe St N, Oshawa, ON L1G
0C5', '4005 Garrard Rd, Whitby, ON L1R 0J1', 2.75, 1), -- 7 -> Pickup
(2, '2021-11-02 7:10', '2021-11-02 7:20', '4005 Garrard Rd, Whitby, ON L1R
0J1', '2000 Simcoe St N, Oshawa, ON L1G 0C5', 2.75, 2), -- Pickup -> 7
(2, '2021-11-02 7:20', '9999-12-31', '2000 Simcoe St N, Oshawa, ON L1G
0C5', '2000 Meadowvale Rd, Toronto, ON M1B 5K7', 26.68, 3), -- 7 -> 6
-- Order 3: Cancelled Order
(3, '2021-11-03 06:15', '2021-11-03 8:20', '6301 Silver Dart Dr,
Mississauga, ON L5P 1B2', '370 Kennedy Rd S, Brampton, ON L6W 4V2', 7.82,
1), -- 1 -> Pickup
(3, '2021-11-03 8:20', '9999-12-31', NULL, NULL, NULL, 6), -- Cancelled
-- Order 4: Order Pending Pick-up
(4, '2021-11-04 05:00', '9999-12-31', '6301 Silver Dart Dr, Mississauga, ON
L5P 1B2', '370 Kennedy Rd S, Brampton, ON L6W 4V2', 7.82, 1); -- 1 ->
Pickup
```

Part C: Views

View 1: Computes a join of at least three tables

```
CREATE VIEW vw_ShowAllOrders AS

-- View 1: Computes a join of at least three tables
-- This view gets all orders in the WeDrone system, who ordered them, their
current status, and their origin/destination
-- and presents it in a user readable manner.

SELECT o.OrderId AS 'Order Id', CONCAT(u.FirstName,' ', u.LastName) AS
'Ordered By',
            lorigin.FullAddress AS 'Package Pick-up',
ldestination.FullAddress AS 'Package Destination', s.StatusName AS 'Current
Status',
            o.OrderCreated AS 'Ordered On'
FROM Orders o
INNER JOIN Users u ON o.UserId = u.UserId
INNER JOIN OrderHistory oh ON oh.OrderId = o.OrderId
INNER JOIN Status s ON s.StatusId = oh.OrderStatus
INNER JOIN Locations lorigin ON lorigin.LocationId = o.Origin
INNER JOIN Locations ldestination ON ldestination.LocationId =
o.Destination
WHERE ValidTo = '9999-12-31';
```

View 2: Uses nested queries with the ANY or ALL operator and uses a GROUP BY clause

```
CREATE VIEW vw_OrdersWithDistanceNotCancelled AS

-- View 2: Uses nested queries with the ANY or ALL operator and uses a
GROUP BY clause
-- Returns all orders that were not cancelled and also provides the
aggregate sum of the distance travelled in each order.

SELECT o.*, od.Distance
From Orders o
INNER JOIN (
      SELECT OrderId, SUM(distance) AS 'Distance'
      FROM OrderHistory oh
      WHERE oh.OrderStatus != 2
      GROUP BY OrderId) AS od ON od.OrderId = o.OrderId
WHERE o.OrderId != ANY(SELECT DISTINCT OrderId FROM OrderHistory WHERE
OrderStatus = 6);
```

View 3: A correlated nested query

```
CREATE VIEW vw_CustomersWithFilledOrders AS
-- View 3: A correlated nested query
-- Returns user records for all users that have created orders that have
been delivered.

SELECT * FROM Users u
WHERE EXISTS (
      SELECT UserId FROM Orders o
      WHERE o.UserId = u.UserId AND o.OrderFilled IS NOT NULL
);
```

View 4: Uses a FULL JOIN

```
CREATE VIEW vw_AllUsersAndTheirOrders AS
-- View 4: Uses a FULL JOIN
-- Returns all users and all of their associated orders, if they have made
any.

SELECT u.*, o.OrderId, o.Origin, o.Destination, o.Weight, o.Volume,
o.OrderCreated, o.OrderFilled
FROM Users u
FULL JOIN Orders o ON o.UserId = u.UserId;
```

View 5: Uses nested queries with any of the set operations UNION, EXCEPT, or INTERSECT

```
CREATE VIEW vw_FlightPlanCTE AS
-- View 5: Uses nested queries with any of the set operations UNION,
EXCEPT, or INTERSECT
-- Uses a recursive CTE to build each step of the route the drone will need
to travel in order from Location 1 to Location 7.

-- Choose the start and end of the route

WITH FlightPlanCTE (RouteId, RouteStart, RouteEnd, CurrentLeg, NextLeg)
AS
(
    -- Initial leg
    SELECT fr.RouteId, RouteStart, RouteEnd, CurrentLeg, NextLeg
      FROM FlightRouteSteps s
      INNER JOIN FlightRoutes fr ON fr.RouteId = s.RouteId
      WHERE fr.RouteStart = 1 AND fr.RouteEnd = 7 AND s.IsInitialLeg = 1
```

```
    UNION ALL

    -- Recursive call to next leg
    SELECT fr.RouteId, fr.RouteStart, fr.RouteEnd, s.CurrentLeg, s.NextLeg
      FROM FlightRouteSteps s
      INNER JOIN FlightRoutes fr ON fr.RouteId = s.RouteId
      INNER JOIN FlightPlanCTE ON s.CurrentLeg = FlightPlanCTE.NextLeg
      WHERE fr.RouteStart = 1 AND fr.RouteEnd = 7
)
-- Query result
SELECT *
FROM FlightPlanCTE fp
INNER JOIN FlightLegs fl ON fp.CurrentLeg = fl.FlightLegId;
```

View 6: Returns all orders that are over 10 kilograms in weight.

```
CREATE VIEW vw_OrdersWithWeightOver10 AS

-- View 6: Returns all orders that are over 10 kilograms in weight.

SELECT o.*
From Orders o
WHERE o.Weight < 10;
```

View 7: Returns all orders that are over 10 cubic feet in volume.

```
CREATE VIEW vw_OrdersWithVolumeOver10 AS

-- View 7: Returns all orders that are over 10 cubic feet in volume.

SELECT o.*
From Orders o
WHERE o.Volume > 10;
```

View 8: Returns all location nodes that are “facility network nodes”.

```
CREATE VIEW vw_ShowFacilityNodes AS

-- View 8: Returns all location nodes that are marked as "facility nodes".
-- These are the nodes that are fixed and represent where drones are stored
-- The opposite is when DroneFacility = 0, these are pickup/drop off
locations
-- Later in the application when more nodes are added, we want to easily be
able
```

```
-- to find these locations from the stored user locations.

SELECT l.*
From Locations l
WHERE l.DroneFacility = 1;
```

View 9: Returns all flight legs between nodes that are less than 10 kilometers.

```
CREATE VIEW vw_FlightLegsLessThan10 AS

-- View 9: Returns all flight legs that are less than 10 km
-- These are the shortest distance routes between nodes

SELECT fl.*
From FlightLegs fl
WHERE fl.Distance < 10;
```

View 10: Returns all orders that were successfully delivered.

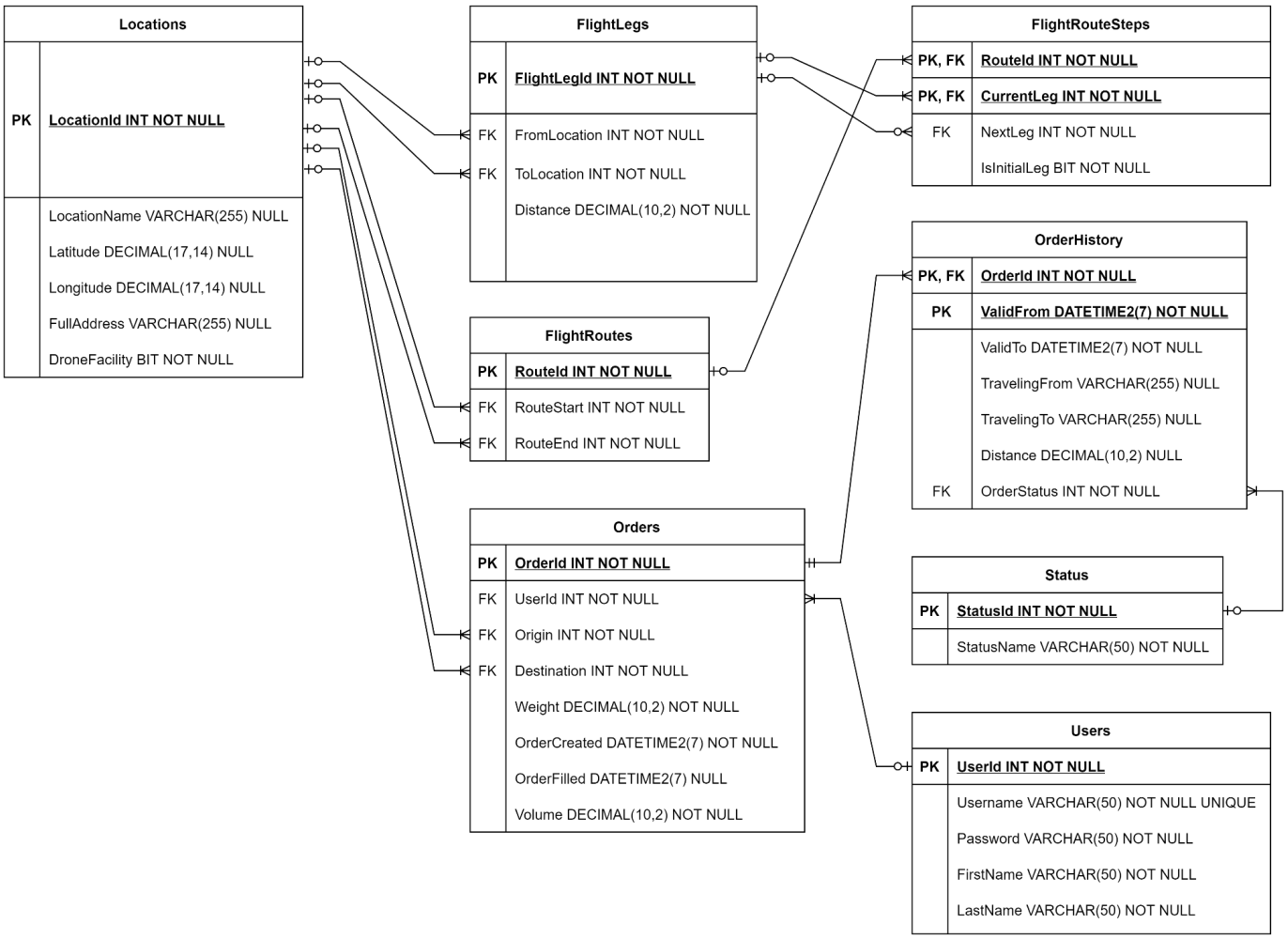
```
CREATE VIEW vw_OrdersDelivered AS

-- View 10: Return all orders that were successfully delivered

SELECT s.*
From Status s
WHERE s.StatusName = 'Delivered';
```

Part D: E-R Schema Diagram

Our design of the primary ER Diagram:



Auto-generated ER Diagram from Microsoft SQL Management Studio: