Drone Dispatch! Express Delivery

CS 4400: Introduction to Database Systems
Course Project: Spring 2024 Semester

Version History

Version	Date	Notes
0	March 24, 2024	Initial release

Main Use Case

The following is a text description of how the system (i.e., the database tables, foreign keys, and related structures) that you've developed will be used. This information will clarify how data flows through the system, and how the views and stored procedures will allow the system operators to observe and modify the state of the system (i.e., database), respectively.

You are being asked to design and develop a system to monitor deliveries of grocery products to customers. This system will support a "third party" grocery service. Customers will place orders with the service. The service will coordinate with grocery stores to find the products (at variable – but hopefully the lowest – prices) and arrange for a drone to deliver the products to the customer. On delivery, the store will be paid electronically by the customer.

Here is a more detailed list of the use case steps as described above, along with the functions and views that support those actions:

[1] Customers purchase groceries.

```
create procedure add_customer (in ip_uname varchar(40),
      in ip_first_name varchar(100), in ip_last_name varchar(100),
      in ip_address varchar(500), in ip_birthdate date,
      in ip_rating integer, in ip_credit integer)
```

This procedure is used to create a new customer if (and only if) the following conditions are met:

- The new customer's username will be unique in the system.

```
create procedure remove_customer (in ip_uname varchar(40))
```

This procedure is used to remove a customer from the system if (and only if) the following conditions are met:

- The customer does not have any pending orders.
- Note: Only the customer should be removed from the system – if the customer is also an employee, then the employee should remain in the system.

[2] Grocery products are sold to customers.

```
create procedure add_product (in ip_barcode varchar(40),
      in ip_pname varchar(100), in ip_weight integer)
```

This procedure is used to create a new product if (and only if) the following conditions are met:

- The new product's barcode will be unique in the system.

```
create or replace view most_popular_products
```

This view displays the status for all products including the product's popularity in terms of the current orders.

create procedure remove_product (in ip_barcode varchar(40))

This procedure is used to remove an product from the system if (and only if) the following conditions are met:

- The product is not being used in any pending orders.

[3] Drones deliver groceries to customers.

```
create procedure add_drone (in ip_storeID varchar(40),
      in ip_droneTag integer, in ip_capacity integer,
      in ip_remaining_trips integer, in ip_pilot varchar(40))
```

This procedure is used to create a new product if (and only if) the following conditions are met:

- The new drone has been purchased by a valid store and will have a unique tag at that store.
- The pilot is not currently piloting a drone.

create or replace view drone_traffic_control

This view displays the status for all drones including the drone's current order delivery activity.

```
create procedure remove_drone (in ip_storeID varchar(40),
      in ip_droneTag integer)
```

This procedure is used to remove a drone from the system if (and only if) the following conditions are met:

- The drone is not carrying any pending orders.

[4] Pilots fly drones.

```
create procedure add_drone_pilot (in ip_uname varchar(40),
      in ip_first_name varchar(100), in ip_last_name varchar(100),
      in ip_address varchar(500), in ip_birthdate date,
      in ip_taxID varchar(40), in ip_service integer,
      in ip_salary integer, in ip_licenseID varchar(40),
      in ip_experience integer)
```

This procedure is used to create a new pilot if (and only if) the following conditions are met:

- The new pilot's username, tax ID and license will be unique in the system.

create or replace view drone_pilot_roster

This view displays the status for all pilots including the pilot's current and pending flight experience.

```
create procedure swap_drone_control (in ip_incoming_pilot varchar(40),
      in ip_outgoing_pilot varchar(40))
```

This procedure is used to change the pilot of a drone if (and only if) the following conditions are met:

- The incoming pilot is a valid pilot in the system.
- The incoming pilot is not currently controlling a drone.
- The outgoing pilot is currently controlling a drone.

create procedure remove_drone_pilot (in ip_uname varchar(40))

This procedure is used to remove a pilot from the system if (and only if) the following conditions are met:

- The pilot is not controlling a drone.
- Note: Only the drone pilot should be removed from the system – if the drone pilot is also a customer, then the customer should remain in the system.

[5] Stores sell grocery products, hire drone pilots, and maintain drones.

`create or replace view store_sales_overview`

This view displays the status for all stores including the store's current and incoming revenue.

`create or replace view role_distribution`

This view displays the number of users in each of the following roles:

- all users;
- customers, employees, and those who are customers and employees; and,
- drone pilots, store workers, and employees in other roles.

```
create procedure repair_refuel_drone (in ip_drone_store varchar(40),
      in ip_drone_tag integer, in ip_refueled_trips integer)
```

This procedure is used to repair and refuel a drone so that it can make more deliveries (i.e., trips), if (and only if) the following conditions are met:

- The proposed change in the number of trips is non-negative.

[6] Customers use money to purchase groceries.

```
create procedure increase_customer_credits (in ip_uname varchar(40),
      in ip_money integer)
```

This procedure is used to increase a customer's current credit (i.e., money for purchasing groceries) level if (and only if) the following conditions are met:

- The proposed change in a customer's credit is non-negative.

`create or replace view customer_credit_check`

This view displays the status of all customers including the credits they have available to purchase groceries.

[7] Customers request grocery products by placing orders.

```
create procedure begin_order (in ip_orderID varchar(40), in ip_sold_on date,
      in ip_purchased_by varchar(40), in ip_carrier_store varchar(40),
      in ip_carrier_tag integer, in ip_barcode varchar(40),
      in ip_price integer, in ip_quantity integer)
```

This procedure is used to initiate an order for groceries with one line product – including price and quantity – if (and only if) the following conditions are met:

- The customer, order ID, drone, and product barcode are all valid.
- The price is non-negative, and the quantity is positive.
- The customer has enough credits to purchase the initial products being ordered.
- The drone has enough lifting capacity to carry the initial products.

```
create procedure add_order_line (in ip_orderID varchar(40),
      in ip_barcode varchar(40), in ip_price integer,
      in ip_quantity integer)
```

This procedure is used to add a new line product – including price and quantity – to an existing order if (and only if) the following conditions are met:

- The order ID and the product's barcode are both valid.
- The product being added is not already part of the order.
- The price is non-negative, and the quantity is positive.
- The customer has enough credits to purchase the products being added to the order.

- The drone has enough lifting capacity to carry the products being added.

create or replace view orders_in_progress
This view displays the status of all orders including information about the cost, weight, and product contents.

[8] Drones deliver prepared/finalized orders to customers.

create procedure deliver_order (in ip_orderID varchar(40))
This procedure is used to deliver a prepared/finalized order to a customer if (and only if) the order ID is valid, and the drone has enough trips to deliver this order. This procedure must update the database state in several ways:

- The customer's credit is reduced by the cost of the order.
- The store's revenue is increased by the cost of the order.
- The drone's number of remaining trips is reduced by one.
- The pilot's experience level is increased by one.
- If the order was more than $25, then the customer's rating is increased by one (if permitted).
- All records of the order are otherwise removed from the system.

[9] Alternatively, customers can cancel pending orders.

create procedure cancel_order (in ip_orderID varchar(40))
This procedure is used to cancel the pending delivery of a prepared/finalized order if (and only if) the order ID is valid. This procedure must update the database state in several ways:

- The customer's rating is decreased by one (if permitted).
- All records of the order are otherwise removed from the system.

Main Use Case – Sample Calls

The following is a set of samples calls for the stored procedures. The sample calls are valid for the initial database state – in other words, the sample call for add_drone_pilot() should successfully add a drone pilot. However, there is some overlap between the sample calls, so running one might modify the database state and prevent another sample call from correctly executing, so it is highly recommended that you use the initial database state or keep careful track of the changes that are being made to the database.

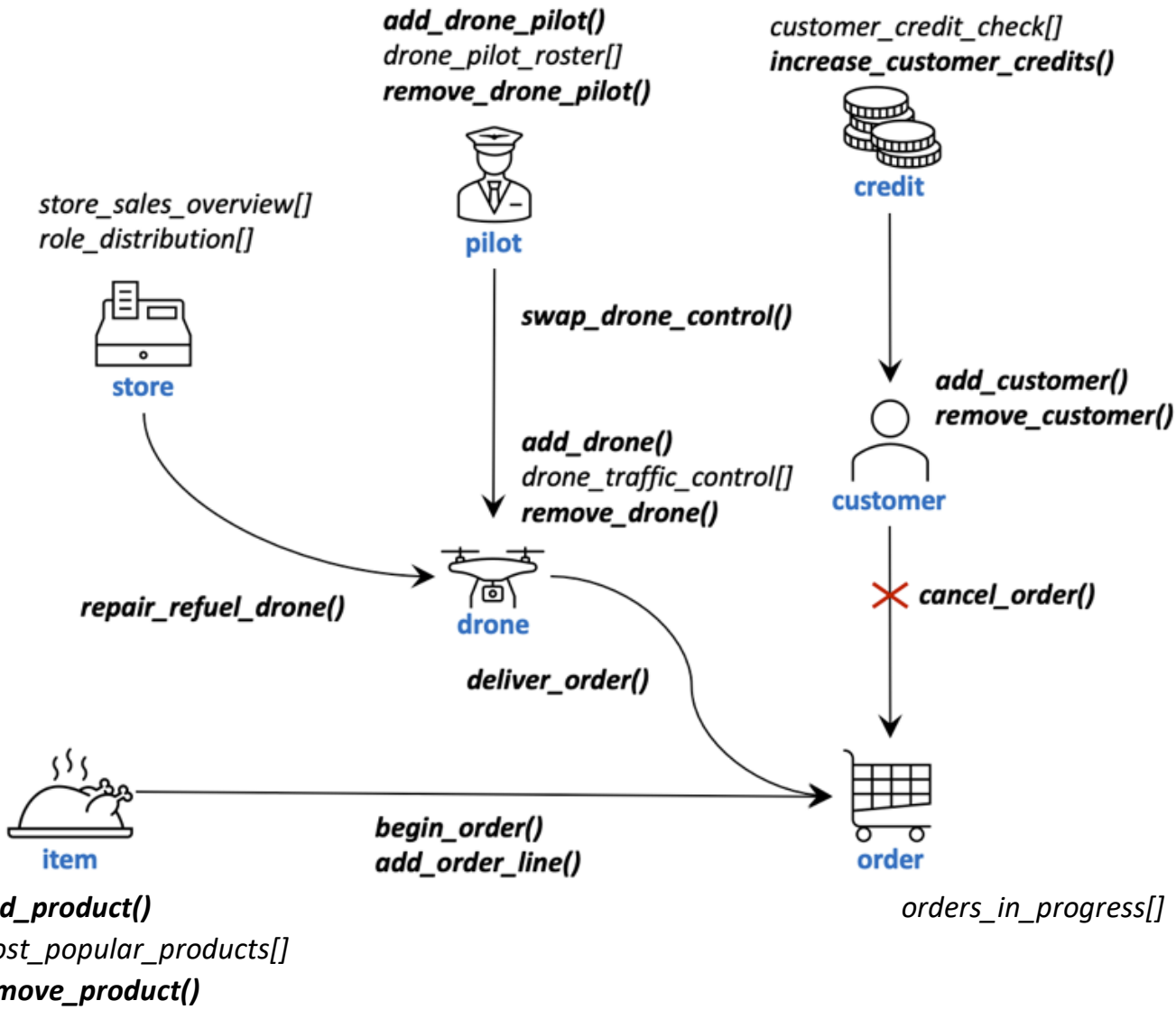
```
CALL add_customer('bsmith', 'Bob', 'Smith', '1155 Morning View Lane', '1987-11-12', 3, 55);

CALL remove_customer('cjordan5');

CALL add_product('kp_2E9A7B', 'key lime pie', 4);

CALL remove_product('ss_2D4E6L');

CALL add_drone('krg', 3, 25, 5, 'bsummers4');

CALL remove_drone('pub', 9);

CALL add_drone_pilot('jlee3', 'James', 'Lee', '8255 Strand Avenue', '1999-03-29', '555-44-6666', 25, 43000, '875564', 4);

CALL repair_refuel_drone('pub', 2, 3);
```

Main Use Case – Sample Calls (continued)

```
CALL increase_customer_credits('jstone5', 15);

CALL begin_order('pub_317', '2024-05-24', 'cjordan5', 'pub', 9,
'hs_5E7L23M', 4, 3);

CALL add_order_line('krg_217', 'hs_5E7L23M', 8, 1);

CALL deliver_order('pub_305');

CALL cancel_order('pub_303');
```

Main Use Case – Visual Overview

The following diagram overview shows ***stored procedures()*** in bold italics, and (global) *views[]* in italics only.



Main Use Case – Expected Results

The following expected results for the global views are generated from the initial database state:

`role_distribution:`

category	total
users	13
customers	6
employees	10
customer_employer_overlap	3
drone_pilots	6
store_workers	3
other_employee_roles	1

`customer_credit_check:`

customer_name	rating	current_credit	credit_already_allocated
awilson5	2	100	16
bsummers4	3	110	0
cjordan5	3	50	0
jstone5	4	40	30
lrodriguez5	4	60	0
sprince6	5	30	30

`drone_pilot_roster:`

pilot	licenseID	drone_serves_store	drone_tag	successful_deliveries	pending_deliveries
agarcia7	610623	NULL	NULL	38	0
awilson5	314159	pub	1	41	1
bsummers4	411911	NULL	NULL	35	0
fprefontaine6	657483	pub	9	2	0
lrodriguez5	287182	pub	2	67	2
tmccall5	181633	krg	1	10	1

`drone_traffic_control:`

drone_serves_store	drone_tag	pilot	total_weight_allowed	current_weight	deliveries_allowed	deliveries_in_progress
krg	1	tmccall5	15	12	4	1
pub	1	awilson5	10	10	3	1
pub	2	lrodriguez5	20	20	2	2
pub	9	fprefontaine6	45	0	1	0

`store_sales_overview:`

store_id	sname	manager	revenue	incoming_revenue	incoming_orders
krg	Kroger	echarles19	300	30	1
pub	Publix	hstark16	200	46	3

most_popular_products:

barcode	product_name	weight	lowest_price	highest_price	lowest_quantity	highest_quantity	total_quantity
ap_9T25E36L	antipasto platter	4	4	10	1	1	2
clc_4T9U25X	chocolate lava cake	5	3	3	2	2	2
hs_5E7L23M	hoagie sandwich	3	3	3	2	2	2
pr_3C6A9R	pot roast	6	15	20	1	2	3
ss_2D4E6L	shrimp salad	3	NULL	NULL	0	0	0

orders_in_progress:

orderID	cost	num_products	payload	contents
krg_217	30	1	12	pot roast
pub_303	24	2	10	antipasto platter,pot roast
pub_305	6	1	10	chocolate lava cake
pub_306	16	2	10	antipasto platter,hoagie sandwich