Smart Refrigerator Database System

CSI 2132 B

February 19th, 2017

Step 1

1)

- Group 4
- Team members (Name, ID, Lab #):
  - Ife Agiri (8205462, B01)
  - Atai Akunov (6084916, B01)
  - NamChi Nguyen (7236760, B01)
  - Lilian Zaky (8237286, B01)

2) Project 1: Smart Refrigerator Database System

3) The objective of this project is to successfully create a clear database describing all types of foods stored inside a refrigerator, as well as a list of meals, each with detailed descriptions. In addition, the database will contain three different types of users with unique functionalities.

- The Chef: This user will be able to place different meals with different ingredients. If those ingredients are not available, the chef will place an order that will need to be approved by the administrator. The chef can also see a report of meals required for a cuisine and if the ingredients for it are available or not.
- Regular Users: This user will be able to select a single food item by requesting it or by checking its availability from a list of food categories. In addition, regular users will have the option of requesting a meal to the chef. If the meal ingredients are available, the order will be delivered. If not, the meals will stay in a queue till the ingredients are ready.
- The Administrator: This user will be able to maintain the stock and place orders requested from the chef or the smart refrigerator. The administrator is in charge of approving the orders and receiving reports produced by the smart refrigerator. This report includes the most expensive meal, popular meals, and the top three ingredients used overall.

4) Possible Attributes, Entities and Relations:

Entities

- *INGREDIENT*
- *MEALS*
- *USERS*
  - *CHEF*
  - *REGULAR USERS*
  - *ADMINISTRATOR*
- *ORDER*
- *REPORT*
- *REFRIGERATOR*

Attributes

- *INGREDIENT(ingredient_id, name, threshold, price_per_item, count, category(grain,dairy,meat, vegetables, fruit, egg, juice), num_times_used)*
- *MEALS(meal_id, name, description, cuisine, ingredients)*
- *ORDER(order_num, price(derived), space_requirement(derived from aggregate size of the order), ingredient_id)*
- *CHEF(chef_id, name)*
- *REPORT(report_id, report_type(expenses, popular_meals, top_ingredient), meal_id, order_num)*
- *REFRIGERATOR*(ingredient_id, meal_id)

Relations

- *CHEF* **PLACES** *ORDER*
- *ADMINISTRATOR* **APPROVES** *ORDER*
- *REGULAR USERS* **REQUESTS** *MEALS*
- *ORDER* **SENT TO** *CHEF*

5) Assumptions for initial design

- An order can only be approved by Administrator once the chef places the order
- Category is a composite attribute
- num_times_used is an enumerated attribute ('rarely, 'sometimes', 'frequently')
- Report type is a composite attribute
- A user can only place one order at a time

6) List of possible SQL DML (with sample SQL queries)

Chef

Enter different types of meals and the required ingredients. If any of the defined ingredients is not available he can place an order that needs to be approved by the administrator.

Create a meal

Note: ing_1...ing_n is a list of ingredients

```
INSERT INTO Meals(meal_id, name, description, cuisine, ingredients)
VALUES (1, 'name', 'description', 'cuisine', '{{ing_1, ing_2, ing_3,...,ing_n}}');
```

Remove a meal

```
DELETE FROM Meals
WHERE meal_id = 1;
```

Add an ingredient

```
INSERT INTO Ingredient(ingredient_id, name, threshold, price_per_item, count,
Category(grain,dairy,meat, vegetables, fruit, egg, juice), num_times_used)
VALUES (2, 'bread', '2017-02-28', 2.80, 1, grain, 0);
```

Remove an ingredient
DELETE FROM Ingredient
WHERE ingredient_id = 2;

Select ingredients and create an order if ingredients aren't in stock
INSERT INTO Order(order_num, price(derived), space_requirement(derived from aggregate size of the order), ingredient_id)
SELECT ingredient_id
FROM Ingredient
WHERE ingredient_id = 2;

See a report of meals that belong to a required cuisine and will be able to see whether the ingredients of any meal are available or not.
SELECT M.meal_id, M.ingredients
FROM Report AS R, Meals AS M
WHERE R.meal_id = M.meal_id AND M.cuisine = 'cuisine';

Regular User
Can get single food item out of the fridge either by requesting the food item name or checking from the list of available food based on its category.
(SELECT DISTINCT name
FROM Refrigerator AS R, Ingredient AS I
WHERE I.ingredient_id = R.ingredient_id AND I.name='food name')
UNION
(SELECT DISTINCT name, category
FROM Ingredient
WHERE category='grain');

Request a meal
SELECT name
FROM Meals
WHERE name= 'meal_name';

Admin
Prepare an order (fridge/chef)
INSERT INTO Order(order_num, price(derived), space_requirement(derived from aggregate size of the order), ingredient_id)
VALUES(1, derived_price, derived_size, 2);

View a report
SELECT report_type
FROM Report
WHERE report_type = 'expenses';

7) Triggers & assertions

Trigger: Once a food product is expired, the trigger would provide the product's id to an admin.

```
CREATE TRIGGER past_expiration_date
        BEFORE ingredient_id, threshold ON food
        FOR EACH ROW
        WHEN (gone_bad.Threshold == 0)
        INFORM_administrator(gone_bad.ingredient_id);
```

Assertion: I want to ensure a chef is not allowed to place orders that would exceed a monthly budget or refrigerator volume constraints (Assumption: monthly_budget and refrigerator_volume_size are defined constants)

```
CREATE ASSERTION Chef_Over_Order
        CHECK(order.price<=monthly_budget OR
        order.space_requrement<=refrigerator_volume_size);
```

8) References

Elmasri, R., & Navathe, S. (2017). *Fundamentals of database systems* (7th ed.). 1272 Seiten: S.n.