Smart Refrigerator Database System

CSI 2132 B
March 10th, 2017
Step 2
Group 4 – B01

Team members (Name, ID):

- Ife Agiri (8205462)
- Atai Akunov (6084916)
- NamChi Nguyen (7236760)
- Lilian Zaky (8237286)

Team member contribution

Member	**Tasks**
Ife Agiri	ER diagram/Relational schema
Atai Akunov	Triggers/Assertions, UI Sketch
NamChi Nguyen	SQL tables/list of queries
Lilian Zaky	ER diagram/Relational schema

Changes/updates made since Step 1:

- Changed the composite attribute of category to an enumerated type
- Removed the list of ingredients as an attribute in the entity Meals to create a relation called Composed_of which has a M:N relation between Meals and Ingredient (ie. Meals are Composed_of Ingredient)
- Removed the entity Report since it will be represented through views and queries (didn't remove from the ER diagram)

1. **ER diagram (see ER-diagram.pdf)**
   Note: Image needs to be rotated.

2. **Relational schema (see Relational-schema.pdf)**
   Note: Image needs to be rotated.

3. **SQL Scripts (also provided in Step_2_Tables.sql file)**

```
/* Composite attributes as enumerated types */
CREATE TYPE category_type AS ENUM ('grain', 'dairy', 'meat', 'vegetable', 'fruit',
'egg', 'juice');

CREATE TYPE usage_type AS ENUM ('rarely', 'sometimes', 'frequently');

/* Database tables */
CREATE TABLE Ingredient(
ingredient_id varchar(10) NOT NULL,
name varchar(15) NOT NULL,
threshold date NOT NULL,
price_per_item numeric CONSTRAINT valid_price CHECK (price_per_item > 0),
count integer,
category category_type,
num_times_used usage_type,
PRIMARY KEY(ingredient_id));

CREATE TABLE Meals(
meal_id integer NOT NULL,
name varchar(30),
description varchar(30),
cuisine varchar(10),
PRIMARY KEY(meal_id));

CREATE TABLE Composed_of(
ingredient_id varchar(10) NOT NULL,
meal_id integer NOT NULL,
quantity integer CONSTRAINT qty_used_in_meal CHECK (quantity > 0),
FOREIGN KEY (ingredient_id) references Ingredient(ingredient_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (meal_id) references Meals(meal_id) ON DELETE RESTRICT ON
UPDATE CASCADE);

CREATE TABLE Orders(
order_num integer NOT NULL,
ingredient_id varchar(10) NOT NULL,
space_requirement integer,
PRIMARY KEY(order_num),
FOREIGN KEY (ingredient_id) references Ingredient(ingredient_id) ON DELETE
RESTRICT ON UPDATE CASCADE);

/* Users */
CREATE TABLE Chef(
chef_id integer NOT NULL,
PRIMARY KEY(chef_id));

CREATE TABLE Regular_user(
order_num integer NOT NULL,
FOREIGN KEY(order_num) references Orders(order_num) ON DELETE RESTRICT
ON UPDATE CASCADE);

CREATE TABLE Administrator (
admin_id integer NOT NULL,
PRIMARY KEY(admin_id));

CREATE TABLE Refrigerator(
ingredient_id varchar(10) NOT NULL,
meal_id integer NOT NULL,
max_capacity integer NOT NULL,
FOREIGN KEY(ingredient_id) references Ingredient(ingredient_id) ON DELETE
RESTRICT ON UPDATE CASCADE,
FOREIGN KEY(meal_id) references Meals(meal_id) ON DELETE RESTRICT ON
UPDATE CASCADE);
```

/* Populating tables */

```
INSERT INTO Ingredient(ingredient_id, name, threshold, price_per_item, count,
category, num_times_used)
VALUES ('I_BRD', 'bread', '2017-02-28', 2.80, 5, 'grain', 'frequently'),
('I_OJ', 'orange juice', '2017-03-28', 2.00, 3, 'juice', 'rarely'),
('I_BGR', 'burger', '2017-02-28', 1.50, 15, 'meat', 'frequently'),
('I_LTCE', 'lettuce', '2017-02-28', 1.50, 10, 'vegetable', 'frequently'),
('I_TOM', 'tomato', '2017-02-28', 1.30, 20, 'vegetable', 'frequently'),
('I_CHKN', 'chicken', '2017-04-10', 3.20, 15, 'meat', 'frequently'),
('I_BTR', 'butter', '2017-05-28', 1.20, 5, 'dairy', 'frequently'),
('I_CBR', 'cucumber', '2017-02-02', 1.00, 10, 'vegetable', 'sometimes'),
('I_CHS', 'cheese', '2017-01-28', 2.00, 12, 'dairy', 'sometimes'),
('I_SPA', 'spaghetti', '2017-07-28', 1.00, 10, 'grain', 'rarely');

INSERT INTO Meals(meal_id, name, description, cuisine)
VALUES (1,'Tomato Soup' , 'A soup made with tomatoes', 'Canadian'),
(2,'Hamburger' , 'A burger', 'Canadian'),
(3,'Spaghetti with chicken' , 'Made with chicken', 'Italian'),
(4,'Green Salad' , 'A salad', 'American');

INSERT INTO Composed_of(ingredient_id, meal_id, quantity)
VALUES ('I_TOM', 1,1),
('I_BRD', 2,2),
('I_BGR', 2,1),
('I_LTCE', 2,1),
('I_TOM', 2, 1),
('I_CHS', 2, 1),
('I_SPA', 3, 1),
('I_CHKN', 3, 1),
('I_BTR', 3, 1),
('I_CBR', 4, 1),
('I_LTCE', 4,5);

INSERT INTO Chef(chef_id)
VALUES (1);

INSERT INTO Regular_user(order_num)
VALUES (1);
```

```
INSERT INTO Administrator(admin_id)
VALUES (1);
```
/* Drop table queries */
```
DROP TABLE Ingredient CASCADE;
DROP TABLE Meals CASCADE;
DROP TABLE Composed_of CASCADE;
DROP TABLE Orders CASCADE;
DROP TABLE Chef CASCADE;
DROP TABLE Regular_user CASCADE;
DROP TABLE Administrator CASCADE;
DROP TABLE Refrigerator CASCADE;
```

4. Queries, triggers & assertions

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
```
INSERT INTO Meals(meal_id, name, description, cuisine)
VALUES (1,'Tomato Soup' , 'A soup made with tomatoes', 'Canadian');

INSERT INTO Composed_of(ingredient_id, meal_id, quantity)
VALUES ('I_TOM', 1,1);
```

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, num_times_used)
VALUES ('I_BRD', 'bread', '2017-02-28', 2.80, 5, 'grain', 'frequently');
```

Remove an ingredient
```
DELETE FROM Ingredient
WHERE ingredient_id = 'I_BRD';
```

Select ingredients and create an order if ingredients aren’t in stock
```
SELECT ingredient_id
FROM Ingredient
WHERE ingredient_id = ‘I_TOM’;

INSERT INTO Order(order_num, ingredient_id, space_requirement)
```
- If the ingredient isn’t in stock, then insert values using the ingredient id

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.

- Create a view that shows a table of meals based on their cuisine and ingredients

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 name
FROM Ingredient
WHERE name = 'broccoli' OR category = 'vegetable';
```

Request a meal

```
SELECT name
FROM Meals
WHERE name = 'Hamburger';
```

Admin

Prepare an order (fridge/chef)

```
INSERT INTO Orders(order_num, ingredient_id, space_requirement)
VALUES (1, 'I_BGR', 5);
```

View the price of an order (derived attribute)

- Will create a function or a view to calculate the price of the order based on the ingredients

View a report of the most expensive meal

```
CREATE VIEW Cost_Of_Meal AS(
        SELECT * FROM Meals NATURAL JOIN
        ( SELECT C.meal_id,
        SUM(C.quantity * I.price_per_item) AS price_of_ingredients
        FROM Composed_of AS C NATURAL JOIN Ingredient
        AS I GROUP BY meal_id) AS tmp_meal);

SELECT name, price_of_ingredients
FROM Cost_Of_Meal
ORDER BY price_of_ingredients DESC;
```

View a report of frequently used ingredients

- Create a view to count the rows where the ingredient_id appears in Composed_of

View a report of the top 3 most used ingredients in a meal

- Create a view to display the ingredients that are frequently used in a meal

Assertion/Triggers

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

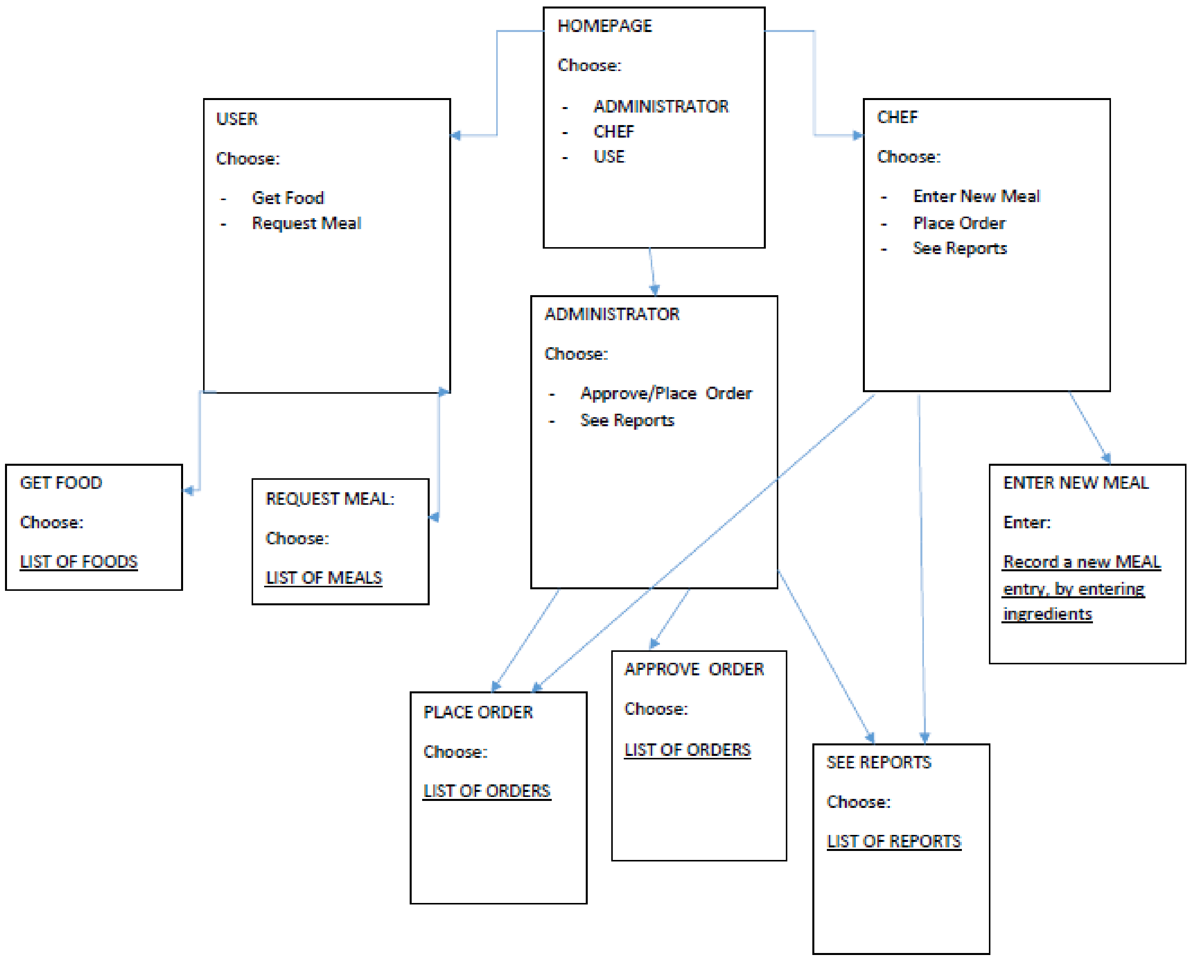
```
CREATE TRIGGER past_expiration_date
        BEFORE ingredient_id, threshold ON food
        FOR EACH ROW
        WHEN (gone_bad.Threshold == 0)
        INFORM_administrator(gone_bad.ingredient_id);
```

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_requirement<=refrigerator_volume_size);
```

Assertion/trigger: To check if the quantity used in a meal isn't higher than the ingredient stock then an order is placed

5. Initial website design


HOMEPAGE
Choose:
- ADMINISTRATOR
- CHEF
- USE
USER
Choose:
- Get Food
- Request Meal
CHEF
Choose:
- Enter New Meal
- Place Order
- See Reports
ADMINISTRATOR
Choose:
- Approve/Place Order
- See Reports
GET FOOD
Choose:
LIST OF FOODS
REQUEST MEAL:
Choose:
LIST OF MEALS
ENTER NEW MEAL
Enter:
Record a new MEAL entry, by entering ingredients
PLACE ORDER
Choose:
LIST OF ORDERS
APPROVE ORDER
Choose:
LIST OF ORDERS
SEE REPORTS
Choose:
LIST OF REPORTS