Comp 421 - Database System

Recipes recommendation database - Group 20

Alexandre Olivier, Vincent Petrella, Timothee Guerin

Project Deliverables 2

I - ER Diagram

Please find with the document the file **result/diagram.png** for an updated version of our ER diagram describing the design of our database.

II - Create SQL Schema

This project's database was built using the MySQL database system. It was set up on our own private computer. Here are the create statement that we used :

```
-- Level
CREATE TABLE `level`
(
   `id` INT NOT NULL,
   `name` VARCHAR(45) NOT NULL,
   `point_required` INT DEFAULT 0,
   PRIMARY KEY (`id`)
);

-- User
CREATE TABLE `user`
(
   `id` INT NOT NULL,
   `email` VARCHAR(45) NOT NULL,
   `password` VARCHAR(45) NOT NULL,
   `guidemode` TINYINT DEFAULT 0,
   `level_id` INT NOT NULL,
   PRIMARY KEY (`id`),
   FOREIGN KEY (`level_id`) REFERENCES `level`(`id`)
);

-- Equipement
CREATE TABLE `equipement`
(
   `id` INT NOT NULL,
```

```
    `name` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`id`)
);


-- Recipe type
CREATE TABLE `recipe_type`
(
    `id` INT NOT NULL,
    `name` VARCHAR(45) NOT NULL,
    PRIMARY KEY (`id`)
);

-- Ingredient
CREATE TABLE `ingredient`
(
    `id` INT NOT NULL,
    `name` VARCHAR(45) NOT NULL,
    `isbasic` TINYINT DEFAULT 0,
    PRIMARY KEY (`id`)
);

-- Unit
CREATE TABLE `unit`
(
    `id` INT NOT NULL,
    `name` VARCHAR(45) NOT NULL,
    `symbol` VARCHAR(5) NOT NULL,
    PRIMARY KEY (`id`)
);

-- Unit conversion
CREATE TABLE `unit_conversion`
(
    `id` INT NOT NULL,
    `from_id` INT NOT NULL,
    `to_id` INT NOT NULL,
    `a` FLOAT NOT NULL,
    `b` FLOAT NOT NULL,
    PRIMARY KEY (`id`),
    FOREIGN KEY (`from_id`) REFERENCES `unit`(`id`),
    FOREIGN KEY (`to_id`) REFERENCES `unit`(`id`)
);

-- Recipe ingredient
CREATE TABLE `recipe`
(
```

```
    `id` INT NOT NULL,
    `name` VARCHAR(255) NOT NULL,
    `preparation_time` TIME NOT NULL,
    `cooking_time` TIME NOT NULL,
    `preparation` TEXT NOT NULL,
    `rating` INT NOT NULL,
    `point_required` INT NOT NULL,
    `type_id` INT NOT NULL,
    `user_id` INT NOT NULL,
    PRIMARY KEY (`id`),
    FOREIGN KEY (`type_id`) REFERENCES `recipe_type`(`id`),
    FOREIGN KEY (`user_id`) REFERENCES `user`(`id`)
);

-- Recipe ingredient
CREATE TABLE `recipe_ingredient`
(
    `quantity` FLOAT NOT NULL,
    `recipe_id` INT NOT NULL,
    `unit_id` INT NOT NULL,
    `ingredient_id` INT NOT NULL,
    PRIMARY KEY (`recipe_id`, `ingredient_id`),
    FOREIGN KEY (`ingredient_id`) REFERENCES `ingredient`(`id`),
    FOREIGN KEY (`unit_id`) REFERENCES `unit`(`id`),
    FOREIGN KEY (`recipe_id`) REFERENCES `recipe`(`id`)
);

-- Recipe ingredient
CREATE TABLE `user_recipe_rating`
(
    `rating` INT NOT NULL,
    `user_id` INT NOT NULL,
    `recipe_id` INT NOT NULL,
    FOREIGN KEY (`user_id`) REFERENCES `user`(`id`),
    FOREIGN KEY (`recipe_id`) REFERENCES `recipe`(`id`)
);

-- User equipement
CREATE TABLE `user_equipement`
(
    `user_id` INT NOT NULL,
    `equipement_id` INT NOT NULL,
    PRIMARY KEY (`user_id`, `equipement_id`),
    FOREIGN KEY (`user_id`) REFERENCES `user`(`id`),
    FOREIGN KEY (`equipement_id`) REFERENCES `equipement`(`id`)
);
```

```
-- User recommmened recipe
CREATE TABLE `user_recommened_recipe`
(
   `user_id` INT NOT NULL,
   `recipe_id` INT NOT NULL,
   PRIMARY KEY (`user_id`, `recipe_id`),
   FOREIGN KEY (`user_id`) REFERENCES `user`(`id`),
   FOREIGN KEY (`recipe_id`) REFERENCES `recipe`(`id`)
);

-- User like ingredent
CREATE TABLE `user_like_ingredient`
(
   `user_id` INT NOT NULL,
   `ingredient_id` INT NOT NULL,
   PRIMARY KEY (`user_id`, `ingredient_id`),
   FOREIGN KEY (`user_id`) REFERENCES `user`(`id`),
   FOREIGN KEY (`ingredient_id`) REFERENCES `ingredient`(`id`)
);

-- User equipement
CREATE TABLE `recipe_equipement`
(
   `recipe_id` INT NOT NULL,
   `equipement_id` INT NOT NULL,
   PRIMARY KEY (`recipe_id`, `equipement_id`),
   FOREIGN KEY (`recipe_id`) REFERENCES `recipe`(`id`),
   FOREIGN KEY (`equipement_id`) REFERENCES `equipement`(`id`)
);
```

Please find with this document the file **results/2_describe_tables.txt**, for it gives the following describe queries outputs in a properly formatted way.

III - Insert Statements

The Insert Statements we used were the following:

```
mysql> INSERT INTO `ingredient` (`id`, `name`, `isbasic`) VALUES (1, 'water', 1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO `ingredient` (`id`, `name`, `isbasic`) VALUES (2, 'pasta', 0);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO `ingredient` (`id`, `name`, `isbasic`) VALUES (3, 'bacon', 0);
Query OK, 1 row affected (0.01 sec)


mysql> INSERT INTO `ingredient` (`id`, `name`, `isbasic`) VALUES (4, 'eggs', 0);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO `ingredient` (`id`, `name`, `isbasic`) VALUES (5, 'cream', 0);
Query OK, 1 row affected (0.02 sec)
```

As we can see in the following output of the SELECT * FROM ingredient query, the data was properly inserted in the DB:

```
mysql> SELECT * FROM ingredient;
+----+-------+---------+
| id | name  | isbasic |
+----+-------+---------+
|  1 | water |       1 |
|  2 | pasta |       0 |
|  3 | bacon |       0 |
|  4 | eggs  |       0 |
|  5 | cream |       0 |
+----+-------+---------+
```

IV - Database Population

We inserted a handful of useful and “real world” data into the database in order to test it.

Please find with this document the file **results/4_select_results.txt** for a properly formatted output of the queries SELECT * FROM ‘table’ on all our DB’s tables.

V - Select Queries

Query 1

Select all recipes that have all the 3 given ingredients (In this case

```
SELECT r.`name` FROM recipe r
JOIN recipe_ingredient ri ON ri.recipe_id = r.id
JOIN ingredient i ON i.id = ri.ingredient_id
WHERE i.`name` in ('cream', 'bacon', 'pasta')
GROUP BY r.`id`
HAVING COUNT(r.`id`) >= 3;
```

Query 2

Select recipe where the specified (here user 1) has all the recipe equipement

```
SELECT `name` FROM recipe
WHERE id not in (
        SELECT r.id FROM recipe r
        JOIN recipe_equipement re ON re.recipe_id = r.id
        WHERE (SELECT COUNT(ue.equipement_id)
                FROM user_equipement ue
                WHERE ue.equipement_id = re.equipement_id AND ue.user_id = 1
        ) = 0
);
```

Query 3

Select all recipes that contains an ingredient the user like and order them by the count of ingreditent the user like found in the recipe

```
SELECT r.name FROM recipe r
JOIN recipe_ingredient ri ON ri.recipe_id = r.id
```

```
WHERE ri.ingredient_id in (
        SELECT ui.ingredient_id FROM user_like_ingredient ui
        WHERE ui.user_id = 1
        )
GROUP BY r.id
ORDER BY COUNT(r.id) DESC;
```

Query 4

Select the most 3 liked ingredients

```
SELECT i.`name` FROM user_like_ingredient ui
JOIN ingredient i ON i.id = ui.ingredient_id
GROUP BY i.id
ORDER BY COUNT(i.id) DESC
LIMIT 3;
```

Query 5

Get the top 10 user at the specific level that make the best recipe. This is calculated by getting the sum of the rating of all recipe a user as made and dividing it by the square root of its count. So a user who made 10 recipes with rating 5 will be better ranked that a user who made one recipe with rating 5.

```
SELECT u.email
FROM recipe r
LEFT JOIN user u ON u.id = r.user_id
JOIN `level` l ON l.id = u.level_id
WHERE l.name = 'Beginner'
GROUP BY r.user_id
ORDER BY SUM(r.rating)/SQRT(COUNT(r.id)) DESC
LIMIT 10;
```

Please find along with the document the file **results/5_query_results.txt** for the properly formatted following information for each query:

1) Tables affected by the query
2) the query input in the MySQL command line interface
2) The output of the query from the MySQL command line interface

VI - Data Modification

Update 1

Updates the level of a user (here user 1) **by getting the logarithm of the amount of recipes he made**

```
UPDATE user
SET level_id = FLOOR(LN((
        SELECT COUNT(r.id) FROM recipe r
        WHERE r.user_id = id
        ))) + 1
WHERE id = 1;
```

Update 2

Updates recipe rating by getting the sum of its ratings

```
UPDATE recipe SET rating = (
        SELECT SUM(ur.rating) FROM user_recipe_rating ur
        WHERE ur.recipe_id = id
        )
WHERE id = 1;
```

Update 3

Deletes all the recipes a user made (here user 7) **if the recipe rating is less than -100.**

```
DELETE FROM recipe
WHERE user_id = 7 AND rating < -100;
```

Update 4

Deletes all recipes containing one particular ingredient (here ‘weed’)

```
DELETE FROM recipe_ingredient
WHERE ingredient_id in (
        SELECT id FROM ingredient
        WHERE `name`='weed'
        )
```

Please find along with the document the file **results/6_update_results.txt** for the properly formatted following information for each modification statement:

1) Involved data set before modification.
2) Modification statement input in the MySQL command line interface.
3) Involved data set after modification.

VII - Views

View 1

Creates a view containing all recipe which are of type 'dessert' (mmmhh)

```
CREATE VIEW recipe_dessert AS
SELECT * FROM recipe
WHERE type_id = 2;
```

View 2

Creates a view containing all easy recipes (with point_required less than 2) and that requires at most 3 ingredients

```
CREATE VIEW easy_recipe AS
SELECT * FROM recipe
WHERE point_required < 2 AND (
        SELECT COUNT(ingredient_id) FROM recipe_ingredient
        WHERE recipe_id = id
        ) <=  3;
```

We use the following queries to retrieve data from our views :

```
-- Select the dessert with a rating of at least 10

mysql> SELECT `name` FROM recipe_dessert WHERE rating >= 10;
+---------------------+
| name                |
+---------------------+
| Exotic Salad    |
+---------------------+
1 row in set (0.00 sec)


-- Select the easy recipes made by user 1

mysql> SELECT `name` FROM easy_recipe WHERE user_id = 1;
+-----------------+
| name          |
+-----------------+
| Apple Pie    |
+-----------------+
1 row in set (0.00 sec)
```

VIII - Checks

As mentioned earlier, we used MySQL as database system, MySQL doesn't support check constraints, so we couldn't produce an output. After discussion with Prof Kemme, since MySQL has XML support (for future deliverables), we were allowed to keep on with MySQL and just provide the scripts for the constraints:

```
-- Rating must be 1 or -1

ALTER TABLE user_recipe_rating
ADD CHECK (rating = 1 OR rating = -1);


-- An ingredient must have quantity greater than 0.

ALTER TABLE recipe_ingredient
ADD CHECK (quantity > 0);
```