Database Lab work assignment 1 实验报告

杨智
PB13011079

实验环境

操作系统:	mac osx 10.10
dbms:	mysql 5.7

实验说明

这个实验挺简单的，纯粹是为了让我们熟悉SQL。所以我在实验报告里面只是把代码贴出来了，没有另作说明。我觉得代码本身都很好看懂的。

exp 1

根据要求创建tables

```
CREATE TABLE IF NOT EXISTS Dish (
        food VARCHAR(2) NOT NULL,
        cuisine ENUM('Chinese', 'Japanese', 'Indian') NOT NULL
        );

CREATE TABLE IF NOT EXISTS Ingredients (
        food VARCHAR(2) NOT NULL,
        ingredient ENUM('beef', 'butter', 'cumin', 'pork', 'rice',
                'fish', 'cabbage', 'tomato') NOT NULL,
        quantity TINYINT UNSIGNED NOT NULL
        );

CREATE TABLE IF NOT EXISTS Calories (
        ingredient ENUM('beef', 'butter', 'cumin', 'pork', 'rice',
                'fish', 'cabbage', 'tomato') NOT NULL,
        quantity TINYINT UNSIGNED NOT NULL,
        calories SMALLINT UNSIGNED NOT NULL
        );

CREATE TABLE IF NOT EXISTS ShoppingList (
        cuisine ENUM('Chinese', 'Japanese', 'Indian') NOT NULL,
        ingredient ENUM('beef', 'butter', 'cumin', 'pork', 'rice',
                'fish', 'cabbage', 'tomato') NOT NULL,
        total_calories FLOAT UNSIGNED NOT NULL
        );
```

在tables中插入初始数据

```
INSERT INTO Dish VALUES
        ('CA', 'Chinese'),
        ('CB', 'Chinese'),
        ('JA', 'Japanese'),
        ('IA', 'Indian'),
        ('IB', 'Indian');

INSERT INTO Ingredients VALUES
        ('CA', 'beef', 30),
        ('CA', 'tomato', 50),
        ('CA', 'cumin', 5),
        ('CB', 'cumin', 10),
        ('CB', 'pork', 30),
        ('CB', 'butter', 5),
        ('CB', 'rice', 15),
        ('JA', 'rice', 50),
        ('JA', 'fish', 30),
        ('JA', 'butter', 16),
        ('IA', 'beef', 45),
        ('IA', 'butter', 20),
        ('IA', 'rice', 30),
        ('IB', 'pork', 30),
        ('IB', 'cumin', 10),
        ('IB', 'cabbage', 20);

INSERT INTO Calories VALUES
        ('beef', 2, 500),
        ('butter', 1, 750),
        ('cumin', 2, 35),
        ('pork', 1, 330),
        ('rice', 10, 700),
        ('fish', 1, 180),
        ('cabbage', 5, 30),
        ('tomato', 1, 30);
```

(a)Find the foods that include more than 16 units of butter. 查询含有大于16单位量的牛油组成的成分的食物

```
SELECT DISTINCT Dish.food
FROM Dish INNER JOIN Ingredients
ON Dish.food = Ingredients.food
WHERE Ingredients.ingredient = 'pork' AND quantity > 16;
```

(b) Find the cuisines that use cumin as an ingredient (in at least one dish). 查询将茴香作为盘菜成分的烹调风格

```
SELECT DISTINCT cuisine
FROM Dish INNER JOIN Ingredients
ON Dish.food = Ingredients.food
WHERE ingredient = 'cumin';
```

(c) Change the ingredients for foods so that they include no more than 16 units of butter. (If they already include more than 16 units of butter, then change it to 16 units of butter.) 修改食物的组成成分，使牛油不超过16单位

```
UPDATE Ingredients
SET quantity = 16
WHERE quantity > 16 AND ingredient = 'butter';
```

(d) Remove beef from all Indian (cuisine) dishes. 删除所有印度风格的牛肉成分

```
DELETE Ingredients.*
```

```
FROM Ingredients INNER JOIN Dish
ON Ingredients.food = Dish.food
WHERE cuisine = 'Indian' AND ingredient = 'beef';
```

(e) Create a summary table, ShoppingList that contains the total quantity of each ingredient used by all the dishes for each cuisine. 创建一张ShoppingList表，对每种风格汇总其食物中每种组成成分的采购量

```
INSERT INTO ShoppingList
SELECT cuisine, ingredient, SUM(quantity) AS total_quantity
FROM Ingredients INNER JOIN Dish
ON Ingredients.food = Dish.food
GROUP BY cuisine, ingredient;
```

(f) For each dish, set the number of calories equal to the total number of calories for the entire dish. Note that the quantity in relation Calories need not match the quantity in Ingredients, so you need to compute calories per unit 对每盘菜，计算其卡路里总和，注意，卡路里表的数量与组成表的数量不一样，因此你需要计算单位量的卡路里量。

```
SELECT food, sum(calories / Calories.quantity * Ingredients.quantity) AS Calories_Sum
FROM Ingredients INNER JOIN Calories
ON Calories.ingredient = Ingredients.ingredient
GROUP BY food;
```

最后，把所有的SQL语句都写在一个EXP1.SQL文件里面：

```
USE LEARNSQL

/*
	CREATING TABLE INGREDIENTS & DISH & CALORIES
*/

DROP TABLE Ingredients;

DROP TABLE Dish;

DROP TABLE Calories;

DROP TABLE ShoppingList;
```

```
CREATE TABLE IF NOT EXISTS Dish (
	food VARCHAR(2) NOT NULL,
	cuisine ENUM('Chinese', 'Japanese', 'Indian') NOT NULL
	);

CREATE TABLE IF NOT EXISTS Ingredients (
	food VARCHAR(2) NOT NULL,
	ingredient ENUM('beef', 'butter', 'cumin', 'pork', 'rice',
		'fish', 'cabbage', 'tomato') NOT NULL,
	quantity TINYINT UNSIGNED NOT NULL
	);

CREATE TABLE IF NOT EXISTS Calories (
	ingredient ENUM('beef', 'butter', 'cumin', 'pork', 'rice',
		'fish', 'cabbage', 'tomato') NOT NULL,
	quantity TINYINT UNSIGNED NOT NULL,
	calories SMALLINT UNSIGNED NOT NULL
	);

INSERT INTO Dish VALUES
	('CA', 'Chinese'),
	('CB', 'Chinese'),
	('JA', 'Japanese'),
	('IA', 'Indian'),
	('IB', 'Indian');

INSERT INTO Ingredients VALUES
	('CA', 'beef', 30),
	('CA', 'tomato', 50),
	('CA', 'cumin', 5),
	('CB', 'cumin', 10),
	('CB', 'pork', 30),
	('CB', 'butter', 5),
	('CB', 'rice', 15),
	('JA', 'rice', 50),
	('JA', 'fish', 30),
	('JA', 'butter', 16),
	('IA', 'beef', 45),
	('IA', 'butter', 20),
	('IA', 'rice', 30),
	('IB', 'pork', 30),
	('IB', 'cumin', 10),
	('IB', 'cabbage', 20);

INSERT INTO Calories VALUES
	('beef', 2, 500),
	('butter', 1, 750),
	('cumin', 2, 35),
	('pork', 1, 330),
	('rice', 10, 700),
	('fish', 1, 180),
	('cabbage', 5, 30),
	('tomato', 1, 30);

SELECT * FROM Dish;
SELECT * FROM Ingredients;

SELECT * FROM Calories;

/* Find the foods that include more than 16 units of butter. */

SELECT DISTINCT Dish.food
FROM Dish INNER JOIN Ingredients
ON Dish.food = Ingredients.food
WHERE Ingredients.ingredient = 'pork' AND quantity > 16;

/* Find the cuisines that use cumin as an ingredient (in at least one dish). */

SELECT DISTINCT cuisine
FROM Dish INNER JOIN Ingredients
ON Dish.food = Ingredients.food
WHERE ingredient = 'cumin';

/* Change the ingredients for foods so that they include no more than 16 units of butter.
        (If they already include more than 16 units of butter, then change it to 16 units of butter.) */

UPDATE Ingredients
SET quantity = 16
WHERE quantity > 16 AND ingredient = 'butter';

SELECT * FROM Ingredients;

/* Remove beef from all Indian (cuisine) dishes.  */

DELETE Ingredients.*
FROM Ingredients INNER JOIN Dish
ON Ingredients.food = Dish.food
WHERE cuisine = 'Indian' AND ingredient = 'beef';

SELECT * FROM Ingredients;

/* Create a summary table, ShoppingList that contains the total quantity of each ingredient
used by all the dishes for each cuisine.  */

CREATE TABLE IF NOT EXISTS ShoppingList (
        cuisine ENUM('Chinese', 'Japanese', 'Indian') NOT NULL,
        ingredient ENUM('beef', 'butter', 'cumin', 'pork', 'rice',
                'fish', 'cabbage', 'tomato') NOT NULL,
        total_calories FLOAT UNSIGNED NOT NULL
        );

INSERT INTO ShoppingList
SELECT cuisine, ingredient, SUM(quantity) AS total_quantity
FROM Ingredients INNER JOIN Dish
ON Ingredients.food = Dish.food
GROUP BY cuisine, ingredient;

SELECT * FROM SHOPPINGLIST;

/* For each dish, set the number of calories equal to the total number of calories
```

```
    for the entire dish. Note that the quantity in relation Calories need not match the quantity in
Ingredients,
    so you need to compute calories per unit. */

SELECT food, sum(calories / Calories.quantity * Ingredients.quantity) AS Calories_Sum
FROM Ingredients INNER JOIN Calories
ON Calories.ingredient = Ingredients.ingredient
GROUP BY food;
```

输出的结果如下:

```
mysql> SOURCE ./EXP1.SQL
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

Query OK, 16 rows affected (0.01 sec)
Records: 16  Duplicates: 0  Warnings: 0

Query OK, 8 rows affected (0.00 sec)
Records: 8  Duplicates: 0  Warnings: 0

+------+----------+
| food | cuisine  |
+------+----------+
| CA   | Chinese  |
| CB   | Chinese  |
| JA   | Japanese |
| IA   | Indian   |
| IB   | Indian   |
+------+----------+
5 rows in set (0.00 sec)

+------+------------+----------+
| food | ingredient | quantity |
+------+------------+----------+
| CA   | beef       |       30 |
| CA   | tomato     |       50 |
| CA   | cumin      |        5 |
| CB   | cumin      |       10 |
| CB   | pork       |       30 |
| CB   | butter     |        5 |
| CB   | rice       |       15 |
| JA   | rice       |       50 |
| JA   | fish       |       30 |
| JA   | butter     |       16 |
| IA   | beef       |       45 |
| IA   | butter     |       20 |
| IA   | rice       |       30 |
| IB   | pork       |       30 |
| IB   | cumin      |       10 |
| IB   | cabbage    |       20 |
+------+------------+----------+
16 rows in set (0.00 sec)

+------------+----------+----------+
| ingredient | quantity | calories |
+------------+----------+----------+
| beef       |        2 |      500 |
| butter     |        1 |      750 |
| cumin      |        2 |       35 |
| pork       |        1 |      330 |
| rice       |       10 |      700 |
| fish       |        1 |      180 |
| cabbage    |        5 |       30 |
| tomato     |        1 |       30 |
+------------+----------+----------+
8 rows in set (0.00 sec)

+------+
| food |
+------+
| CB   |
| IB   |
+------+
2 rows in set (0.00 sec)

+---------+
| cuisine |
+---------+
| Chinese |
| Indian  |
+---------+
2 rows in set (0.00 sec)

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

+------+------------+----------+
| food | ingredient | quantity |
+------+------------+----------+
| CA   | beef       |       30 |
| CA   | tomato     |       50 |
| CA   | cumin      |        5 |
| CB   | cumin      |       10 |
| CB   | pork       |       30 |
| CB   | butter     |        5 |
| CB   | rice       |       15 |
| JA   | rice       |       50 |
| JA   | fish       |       30 |
| JA   | butter     |       16 |
| IA   | beef       |       45 |
| IA   | butter     |       16 |
| IA   | rice       |       30 |
| IB   | pork       |       30 |
| IB   | cumin      |       10 |
| IB   | cabbage    |       20 |
+------+------------+----------+
16 rows in set (0.00 sec)

Query OK, 1 row affected (0.00 sec)

+------+------------+----------+
| food | ingredient | quantity |
+------+------------+----------+
| CA   | beef       |       30 |
| CA   | tomato     |       50 |
| CA   | cumin      |        5 |
| CB   | cumin      |       10 |
| CB   | pork       |       30 |
| CB   | butter     |        5 |
| CB   | rice       |       15 |
| JA   | rice       |       50 |
| JA   | fish       |       30 |
| JA   | butter     |       16 |
| IA   | butter     |       16 |
| IA   | rice       |       30 |
| IB   | pork       |       30 |
| IB   | cumin      |       10 |
| IB   | cabbage    |       20 |
+------+------------+----------+
15 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 14 rows affected (0.00 sec)
Records: 14  Duplicates: 0  Warnings: 0

+----------+------------+----------------+
| cuisine  | ingredient | total_calories |
+----------+------------+----------------+
| Chinese  | beef       |             30 |
| Chinese  | butter     |              5 |
| Chinese  | cumin      |             15 |
| Chinese  | pork       |             30 |
| Chinese  | rice       |             15 |
| Chinese  | tomato     |             50 |
| Japanese | butter     |             16 |
| Japanese | rice       |             50 |
| Japanese | fish       |            30 |
| Indian   | butter     |            16 |
| Indian   | cumin      |            10 |
| Indian   | pork       |            30 |
| Indian   | rice       |            30 |
| Indian   | cabbage    |            20 |
+----------+------------+---------------+
14 rows in set (0.00 sec)

+------+--------------+
| food | Calories_Sum |
+------+--------------+
| CA   |    9087.5000 |
| CB   |   14875.0000 |
| IA   |   14100.0000 |
| IB   |   10195.0000 |
| JA   |   20900.0000 |
+------+--------------+
5 rows in set (0.00 sec)

mysql>
```

— — — — — — — — — — — — — — — —

— — — — — — — — — — — — — — — —

exp 2

根据实验要求创建tables

```
CREATE TABLE IF NOT EXISTS Cars(
        licence VARCHAR(10) NOT NULL,
        owner VARCHAR(10) NOT NULL,
        model VARCHAR(10) NOT NULL
        );

CREATE TABLE IF NOT EXISTS Motorcycle(
        licence VARCHAR(10) NOT NULL,
        owner VARCHAR(10) NOT NULL,
        model VARCHAR(10) NOT NULL
        );

CREATE TABLE IF NOT EXISTS Ownership(
        owner VARCHAR(10),
        numberOfCars TINYINT UNSIGNED NOT NULL,
        numberOfMotorcycles TINYINT UNSIGNED NOT NULL
        );
```

在tables中插入初始数据

```
INSERT INTO Cars VALUES
        ('C011', 'P1', 'M1'),
        ('C012', 'P3', 'M1'),
        ('C013', 'P4', 'M3'),
        ('C014', 'P4', 'M2'),
        ('C015', 'P5', 'M3');

INSERT INTO Motorcycle VALUES
        ('M011', 'P2', 'M7'),
        ('M012', 'P3', 'M8'),
        ('M013', 'P4', 'M7'),
        ('M014', 'P4', ‘M8’);
```

(a) List people who own a car but do not own a motorcycle.

```
SELECT DISTINCT Cars.owner
FROM Cars LEFT JOIN Motorcycle
ON Cars.owner = Motorcycle.owner
WHERE Motorcycle.owner IS NULL;
```

(b) List people who own a car and also own a motorcycle.

```
SELECT DISTINCT Cars.owner
FROM Cars INNER JOIN Motorcycle
ON Cars.owner = Motorcycle.owner;
```

(c) List the number of cars owned by each person.

```
SELECT owner, COUNT(licence) AS CAR_NUM
FROM Cars
GROUP BY owner;
```

(d)Write ONE SQL request that populates the table :
Ownership(owner, numberOfCars, numberOfMotorcycles)
with all the owners in Cars and in Motorcycles and includes the counts of their cars and motorcycles. Make sure this request works for people who own both car(s) and motorcycle(s), car(s) but not motorcycle(s), and motorcycle(s) and not car(s).

```
INSERT INTO Ownership
SELECT OWNER, SUM(C1) AS numberOfCars, SUM(C2) AS numberOfMotorcycles
FROM (
SELECT OWNER, COUNT(LICENCE) AS C1, '' AS C2 FROM CARS GROUP BY OWNER
UNION
SELECT OWNER, '', COUNT(LICENCE) FROM Motorcycle GROUP BY OWNER
) AS TT
GROUP BY OWNER;
```

把所有的SQL语句写在一个EXP2.SQL文件里面

```
USE LEARNSQL

/*
        CREATING TABLES
*/

DROP TABLE Cars;

DROP TABLE Motorcycle;

DROP TABLE Ownership;

CREATE TABLE IF NOT EXISTS Cars(
        licence VARCHAR(10) NOT NULL,
        owner VARCHAR(10) NOT NULL,
        model VARCHAR(10) NOT NULL
        );

CREATE TABLE IF NOT EXISTS Motorcycle(
        licence VARCHAR(10) NOT NULL,
        owner VARCHAR(10) NOT NULL,
        model VARCHAR(10) NOT NULL
        );

INSERT INTO Cars VALUES
        ('C011', 'P1', 'M1'),
        ('C012', 'P3', 'M1'),
        ('C013', 'P4', 'M3'),
        ('C014', 'P4', 'M2'),
        ('C015', 'P5', 'M3');

INSERT INTO Motorcycle VALUES
        ('M011', 'P2', 'M7'),
        ('M012', 'P3', 'M8'),
        ('M013', 'P4', 'M7'),
        ('M014', 'P4', 'M8');

SELECT * FROM Cars;

SELECT * FROM Motorcycle;

/* List people who own a car but do not own a motorcycle. */

SELECT DISTINCT Cars.owner
FROM Cars LEFT JOIN Motorcycle
ON Cars.owner = Motorcycle.owner
WHERE Motorcycle.owner IS NULL;

/* List people who own a car and also own a motorcycle. */

SELECT DISTINCT Cars.owner
FROM Cars INNER JOIN Motorcycle
ON Cars.owner = Motorcycle.owner;

/* List the number of cars owned by each person. */

SELECT owner, COUNT(licence) AS CAR_NUM
FROM Cars
GROUP BY owner;

/* Write ONE SQL request that populates the table :
        Ownership(owner, numberOfCars, numberOfMotorcycles) */

CREATE TABLE IF NOT EXISTS Ownership(
        owner VARCHAR(10),
        numberOfCars TINYINT UNSIGNED NOT NULL,
        numberOfMotorcycles TINYINT UNSIGNED NOT NULL
        );

INSERT INTO Ownership
SELECT OWNER, SUM(C1) AS numberOfCars, SUM(C2) AS numberOfMotorcycles
FROM (
SELECT OWNER, COUNT(LICENCE) AS C1, '' AS C2 FROM CARS GROUP BY OWNER
UNION
SELECT OWNER, '', COUNT(LICENCE) FROM Motorcycle GROUP BY OWNER
) AS TT
GROUP BY OWNER;

SELECT * FROM Ownership;
```

结果如下：

```
mysql> SOURCE ./EXP2.SQL
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.04 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

+---------+-------+-------+
| licence | owner | model |
+---------+-------+-------+
| C011    | P1    | M1    |
| C012    | P3    | M1    |
| C013    | P4    | M3    |
| C014    | P4    | M2    |
| C015    | P5    | M3    |
+---------+-------+-------+
5 rows in set (0.00 sec)

+---------+-------+-------+
| licence | owner | model |
+---------+-------+-------+
| M011    | P2    | M7    |
| M012    | P3    | M8    |
| M013    | P4    | M7    |
| M014    | P4    | M8    |
+---------+-------+-------+
4 rows in set (0.00 sec)

+-------+
| owner |
+-------+
| P1    |
| P5    |
+-------+
2 rows in set (0.00 sec)

+-------+
| owner |
+-------+
| P3    |
| P4    |
+-------+
2 rows in set (0.00 sec)

+-------+---------+
| owner | CAR_NUM |
+-------+---------+
| P1    |       1 |
| P3    |       1 |
| P4    |       2 |
| P5    |       1 |
+-------+---------+
4 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

+-------+--------------+---------------------+
| owner | numberOfCars | numberOfMotorcycles |
+-------+--------------+---------------------+
| P1    |            1 |                   0 |
| P2    |            0 |                   1 |
| P3    |            1 |                   1 |
| P4    |            2 |                   2 |
| P5    |            1 |                   0 |
+-------+--------------+---------------------+
5 rows in set (0.00 sec)

mysql>
```