INFO 6210
Data Management and Database Design
Final Project Report

VR kitchen

Team Zero

Yile Peng
Tianyu Zhong
Xiao Luo
Haoqi Huang
Jiaming Duan

Professor Chaiyaporn Mutsalklisana

Contents

Abstract

Cooking is one of the most necessary elements in our daily life. Good dish will bring much more fun to our life and families. But bad cooking outcome may also ruin our mood. It is inevitable for us to cook badly for more than once. And it will be hard for cleaning the kitchen. Wives may spend much time cleaning up the charcoal and oil. At the same time, lots of food is wasted, money as well.

Based on thought that helps cooking amateur enjoy cooking without any hurts and uncomfortable, the VR kitchen comes out for helping them. Besides, we provide certificate in the VR kitchen for any cooking lovers and cooking trainees. Many certificates of food styles are available including Chinese, American, French, Italian, etc.

More than in class, the system provides a game mode that players should challenge the master of cooking. Only if players beat the master will an honor be given. We want to increase the interests of cooking, and to attract more users.

Finally, a social platform is established for making friends and interaction among users. Every user can share their cooking accomplishment with their friends, families, and mates. Comments and likes may be added into their accomplishments.

Key words: database design, VR, algorithms

1. Introduction

1.1 Objectives and functions

Before we start design the system, we decided 5 objectives of the project:

1. Help/train non-professional cooking amateur
2. Saving material cost
3. Providing professional certificate for job hunting
4. Design a game mode for spare time entertainment
5. Give cook lover a platform to get new friends

Based on these objectives, we come up with some questions for the designing of features. And we decide the functions by answering these questions.

Who is the trainer/trainee?
Trainer: professional cooks who can teach student finish tasks step by step
Trainee: people who want to be a professional cook, people who want to learn cook, and people who cook for fun.
Data associated: User ID, user type, name,

What is the cooking dish?
Based on recipe, dishes can be possible achieved by all cooking methods such as cutting, boiling, baking and fundamental cooking procedures. Users can cook everything they want like French (seafood, steak), American(ribs, fries, fried chicken, salad, sandwich…)

How to accomplish the process?
Based on given recipe, and following the instructions step by step in a visual reality environment. Any action can be done in this kitchen. Also we will design algorithm for simulate the cooking process, in order to give users real result of their accomplishment.
Data associated: recipe, tutorial (general and step)

How to teach student without instructors?
By video instruction in each step during cooking. Besides videos, text recipe will work during cooking process. Also we may introduce other users come in your kitchen to supervise.

How can we save materials with this system?

By virtual kitchen, users do not have to buy real ingredients and seasonings that need much money. Users can just operate the virtual ingredients, kitchen facilities and cook virtually. Thus, without purchasing in groceries and fuel cost, users may save much money.

How much money will be saved?
We will introduce real-time grocery and fuel price in our dataset. Once user begins operation, we can calculate the cost of material that he/she has used. After operation, the system will give a price of saving in this cooking.
Data associated: price of ingredient, fuel price, electricity price, and cost for garbage dispose.

What else can be saved in this system?
Besides purchasing saving, users can avoid risk of injury from fire, heat, frozen, cutting, etc. We all know many people get injured in their kitchen. With our system, this risk can be reduced or even avoided.

What is the grading mechanism?
From users' achievement, we may consider all properties of the good flavor but not limited by salty, sweetness, spicy, texture, doneness, juicy, color, nutrition combination, etc. To achieve our aim, we will quantify all properties listed above. For example, we may grade score of them with 100 full marks. Then we will design a algorithm with weight parameters to calculate the final score. Of course users can set their own preference. By adjusting the weight parameters, every taste can be satisfied.
Data associated: Score of all properties related to flavor, color and nutrition, weight parameters, user account.

What courses does the certificate/program set?
Different certificate/program will serve different goal. For example, the certificate for fast-food restaurant cook's course will focus on hamburger making, French fries and so on. And a certificate for Chinese food so on will include on some unique Chinese food cooking way, and focus on dishes like kung pao chicken and so on.
Data associated: certificate, program, courses

What is the standard that student will get the certificate?
A good score passing course in: Concept knowledge, Tutorial finish process in %, assignment score in %, and exam score.
Data associated: Courses will include (homework scores, tutorial finish process, reading quiz, and exam score)

How the certificate will help student for job hunting?
Just like some online courses certificate for specific skills is valid for job hunting (like learning C++ in an online school), The certificate in VR kitchen is going to show that the student who get the certificate has gain enough knowledge and skills to finish given tasks.

How can this system entertain people?
We design a game mode that users may start their chef career from a newcomer.
Data associated: recipe (in the game mode), user account, user accomplishment, game story, ingredient, facilities, seasoning...

What the game mode is in this system?
At the beginning, player is a rookie without any title and honor. By training by inner recipe, fighting with other chef, and attend competitions, player will gather much more grades and will become a cooking master in the end.

What is the path for winning?
Users may attend competition, beat their against, win awards to get more title and honor. At some experience level, user will get a specific honor.

Can users get interaction from other users/players?
Yes. We provide a public platform that users may battle for a title. Scored by a fair algorithm, users may win or lose according to their accomplishment.

How can users network others?
If users set their profile open to public, other users can view their cooking graph, and make comments (like Instagram). We will build a social network platform that is provided for posting their accomplishment and experience. 5.2 What element can user show in their networking?
Materials will be included such as pictures, recipes, videos, text, personalized kitchen...
Data associated: User ID (type,name, dob, interest, cook history), privacy set, user type, name, interest, posting

Other things that may increase intimacy between users?
We can quantify people's interaction in the platform, and give people who have high interaction game bounce or limited recipe.
Data associated: friendship status

1.2 Sequential Diagram

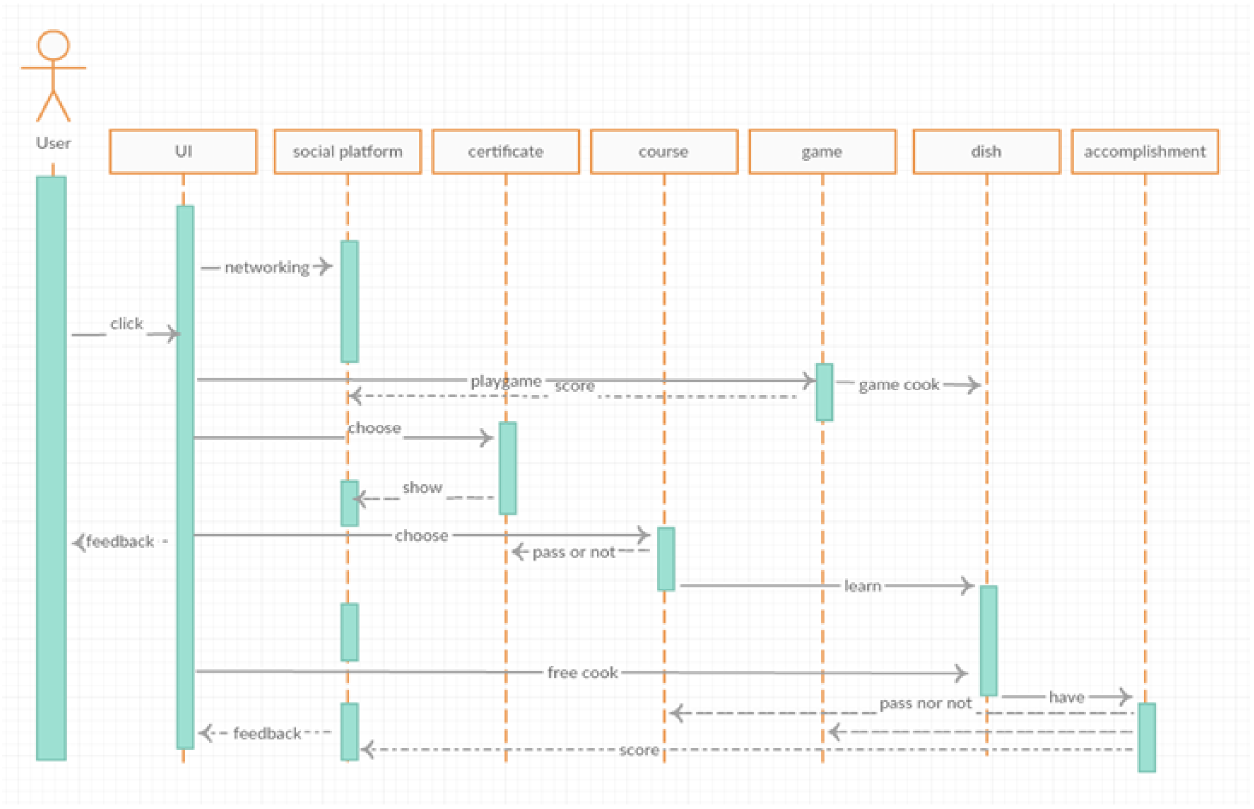


Figure 1-1 Sequential Diagram

From Figure1-1, we can see users are always interactive with UI, which we call the front in, the part of visual reality. And User can view other people's page via social platform. They can also go to certificate selection, and choose an exact course they want to learn, an exact dish they want to cook. Dishes can also be accessed by game mode. If user does not like either of them, they can cook a dish directly from the main page. And each dish will generate a score and feedback from the accomplishment. This score will also be available in the social platform.

1.3 ER diagram

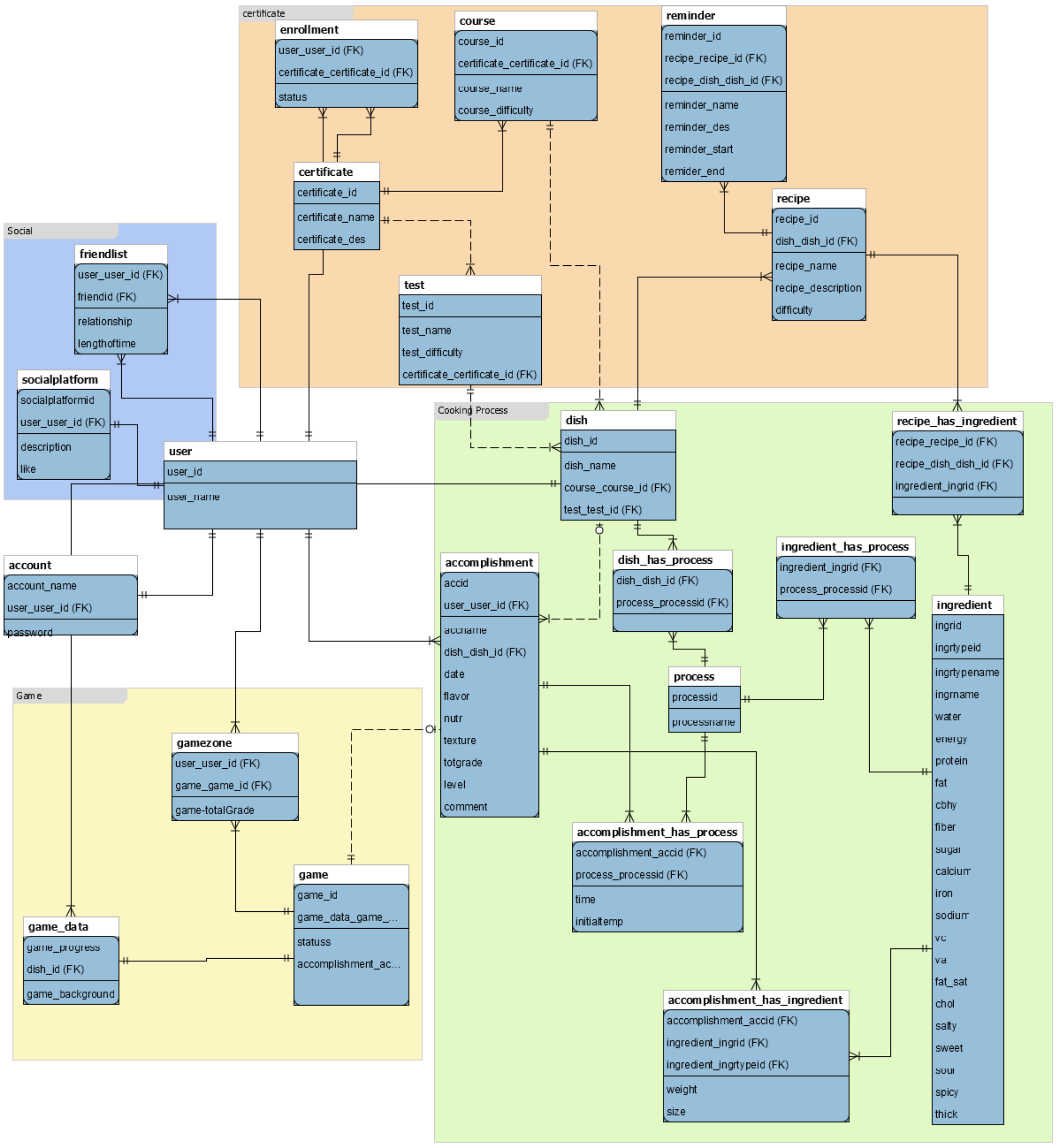


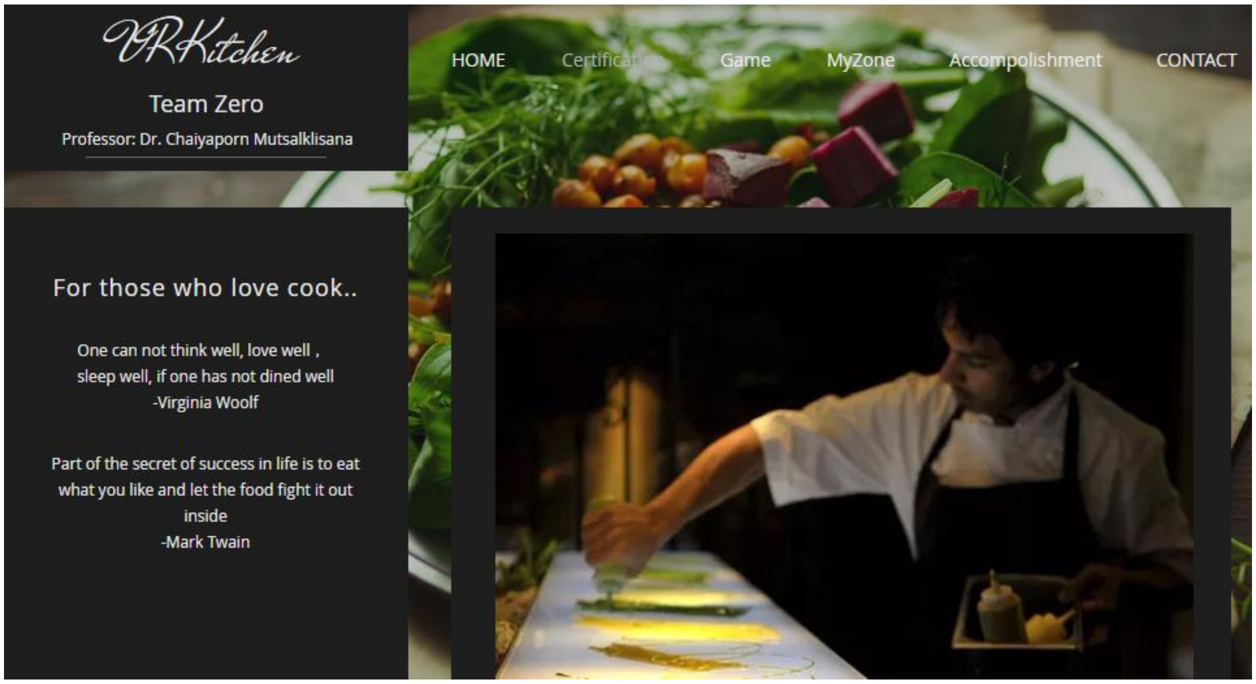
Figure 1-2 ER diagram

Our ERD have 22 entities in total. To make it more clear in functions, we divide them into 5 zone, which are user, social, certificate, game, and cooking process.

1.4 Normalization Form

We make our final ERD in BCNF for normalization. We remove all multi-value attributes, partial dependencies, transitive dependencies, and remaining anomalies. Although doing BCNF will increase the number of tables, it eliminate the anomalies like insert anomaly, update anomaly and delete anomaly in data processing. At the same time, this normalization form is easier for sorting, indexing and searching.

1.5 Front-end design



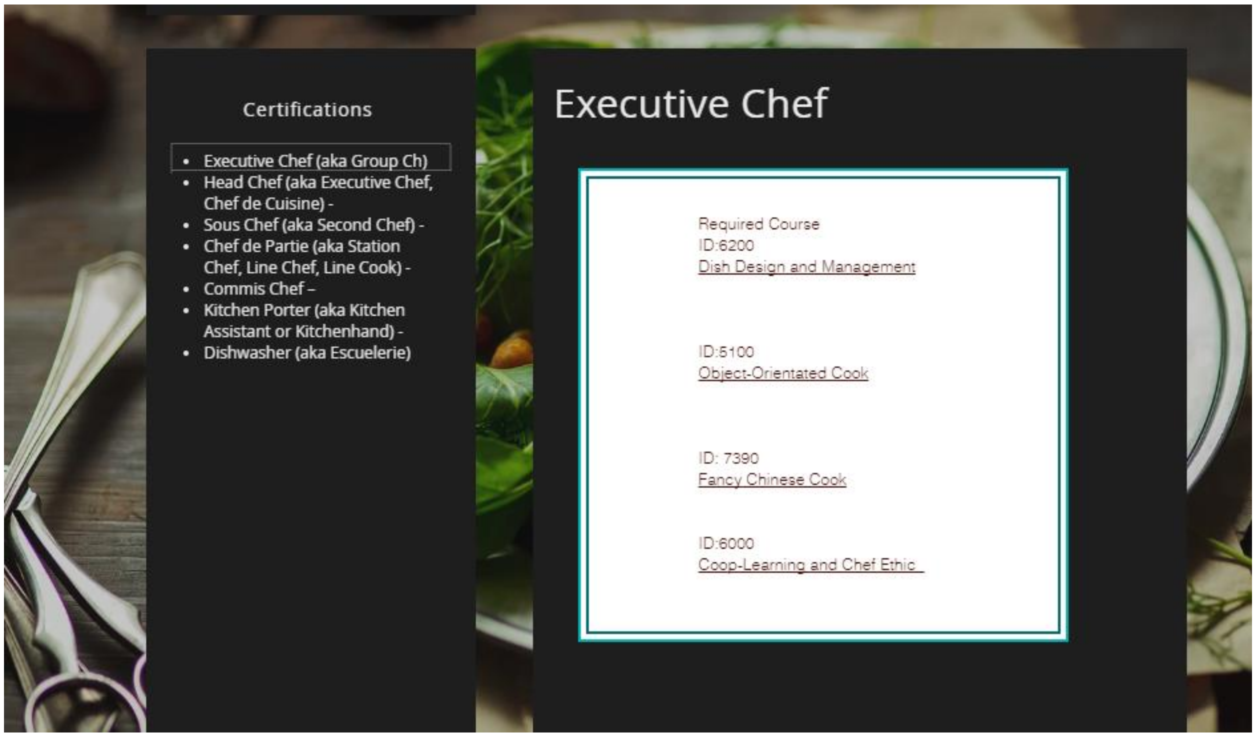
For those who love cook..

One can not think well, love well , sleep well, if one has not dined well
-Virginia Woolf

Part of the secret of success in life is to eat what you like and let the food fight it out inside
-Mark Twain

Figure 1-3 (a) main menu of front-end

The diagram a is the blueprint of our project’s front end. Users could choose to play games, view the works they have done, and get certificated by finishing required classes.
For example, after clicking Certification button in Figure1-3 (a), a new page with information of Certification emerges.



Certifications

- Executive Chef (aka Group Ch)
- Head Chef (aka Executive Chef, Chef de Cuisine) -
- Sous Chef (aka Second Chef) -
- Chef de Partie (aka Station Chef, Line Chef, Line Cook) -
- Commis Chef –
- Kitchen Porter (aka Kitchen Assistant or Kitchenhand) -
- Dishwasher (aka Escuelerie)

Executive Chef

Required Course
ID:6200
Dish Design and Management

ID:5100
Object-Orientated Cook

ID: 7390
Fancy Chinese Cook

ID:6000
Coop-Learning and Chef Ethic

Figure 1-3(b) Executive Chef Certification requirement page

1.6 Use case & Account type(privileges)

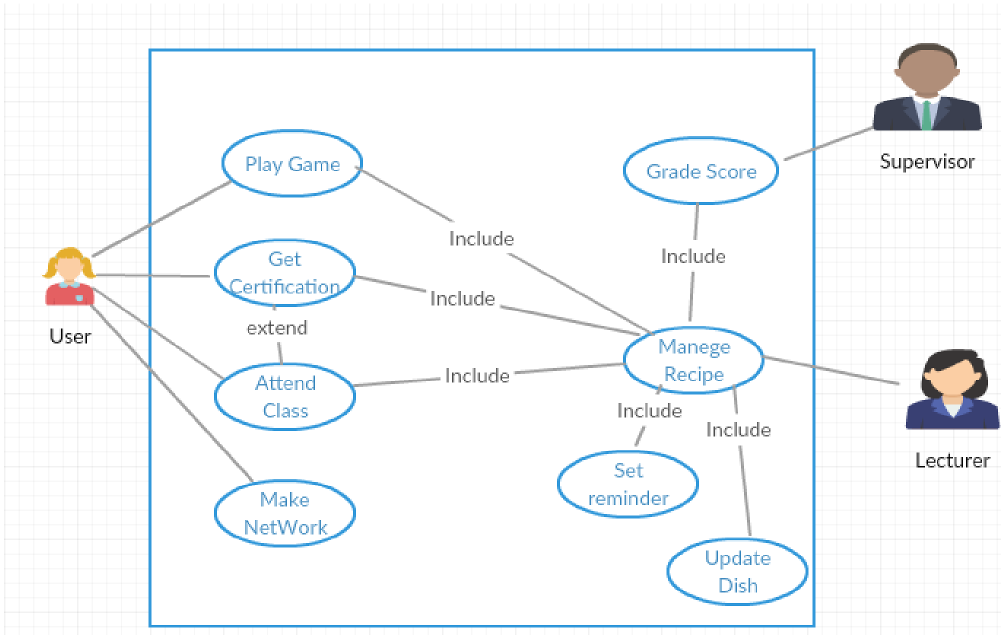


Figure 1-4 Use Case

Basically, users have 3 sections to pick: games, certifications and social network, and there are lectures to help them with certifications test and offer reminder during cooking. Supervisor is in the charge of run grading functions and make sure whole VR Kitchen works well. So we set the privileges for these three account types depending on their purpose.

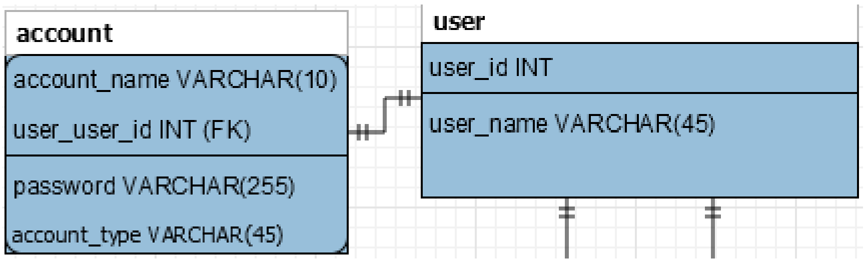
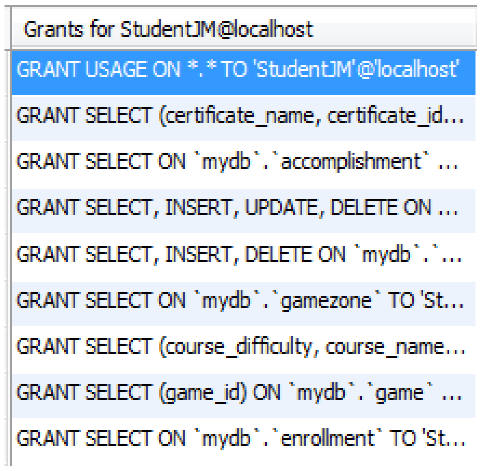


Figure 1-5 account and user table

Appendix I-1 shows the code for privilege .For users, they are allowed to update their password and friend list, but they can only select instead of updating in some parts like courses. Besides, the right of Instructors is limited in course, dish and reminders because their duties is just to manage teaching. However, Supervisors are supposed to monitor all so they have all privileges.



```
Grants for StudentJM@localhost
GRANT USAGE ON *.* TO 'StudentJM'@'localhost'
GRANT SELECT (certificate_name, certificate_id...
GRANT SELECT ON `mydb`.`accomplishment` ...
GRANT SELECT, INSERT, UPDATE, DELETE ON ...
GRANT SELECT, INSERT, DELETE ON `mydb`.`...
GRANT SELECT ON `mydb`.`gamezone` TO 'St...
GRANT SELECT (course_difficulty, course_name...
GRANT SELECT (game_id) ON `mydb`.`game` ...
GRANT SELECT ON `mydb`.`enrollment` TO 'St...
```

Figure 1-6 Privilege for user

2. Certifications

Learning how to cook dishes and get certificates is an important part of our project

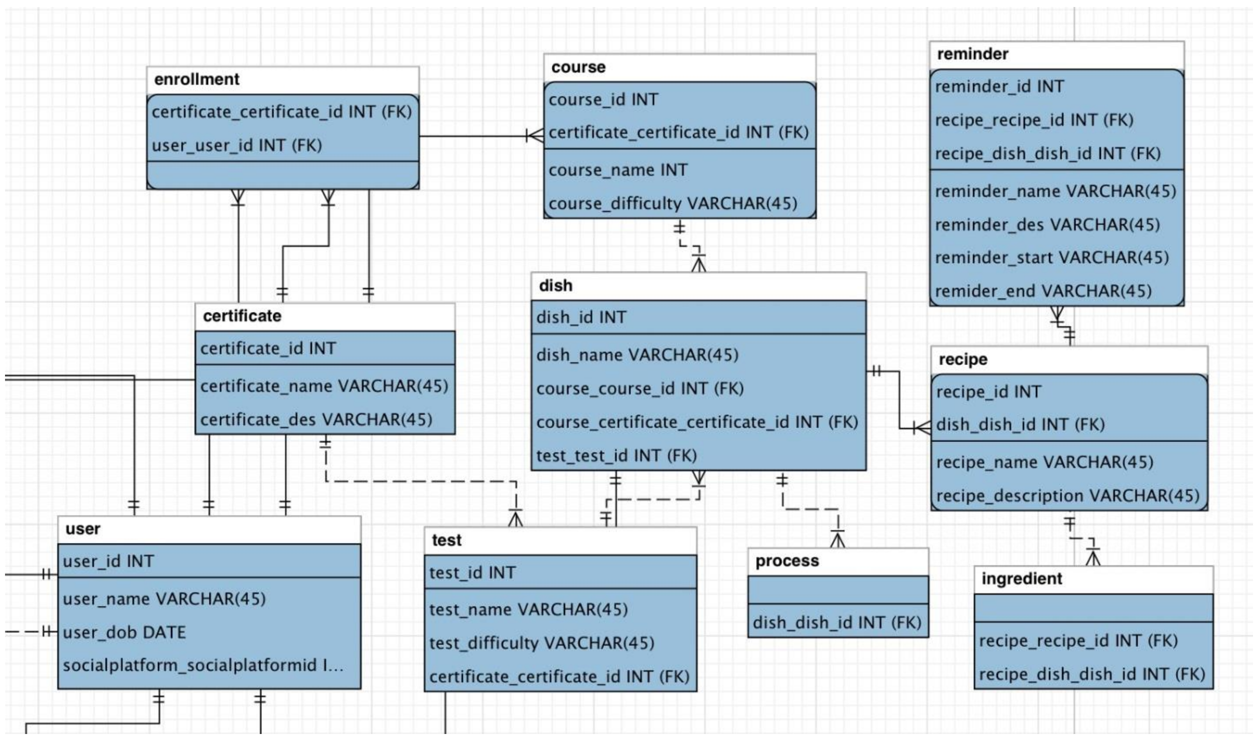
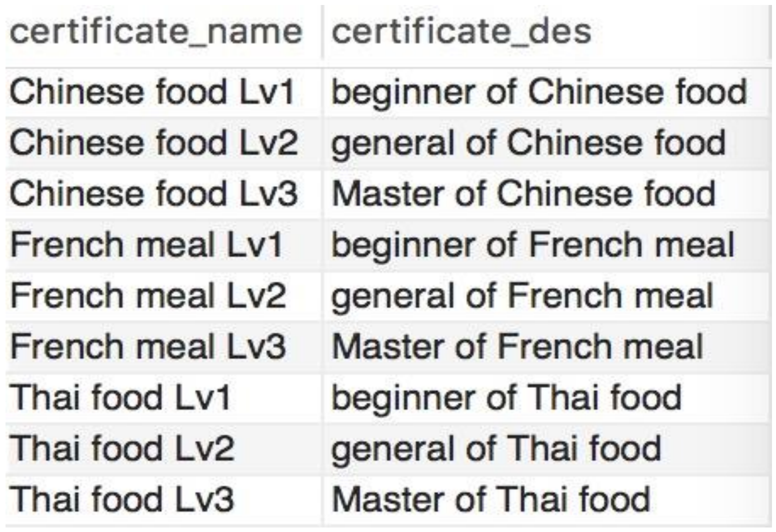

enrollment
certificate_certificate_id INT (FK)
user_user_id INT (FK)
course
course_id INT
certificate_certificate_id INT (FK)
course_name INT
course_difficulty VARCHAR(45)
reminder
reminder_id INT
recipe_recipe_id INT (FK)
recipe_dish_dish_id INT (FK)
reminder_name VARCHAR(45)
reminder_des VARCHAR(45)
reminder_start VARCHAR(45)
remider_end VARCHAR(45)
certificate
certificate_id INT
certificate_name VARCHAR(45)
certificate_des VARCHAR(45)
dish
dish_id INT
dish_name VARCHAR(45)
course_course_id INT (FK)
course_certificate_certificate_id INT (FK)
test_test_id INT (FK)
recipe
recipe_id INT
dish_dish_id INT (FK)
recipe_name VARCHAR(45)
recipe_description VARCHAR(45)
user
user_id INT
user_name VARCHAR(45)
user_dob DATE
socialplatform_socialplatformid I...
test
test_id INT
test_name VARCHAR(45)
test_difficulty VARCHAR(45)
certificate_certificate_id INT (FK)
process
dish_dish_id INT (FK)
ingredient
recipe_recipe_id INT (FK)
recipe_dish_dish_id INT (FK)

Figure 2-1 ERD of certificate

We can see in the figure2-1, users need to enroll certificates. We provide many courses for a certificate. So you can choose dishes to learn in a course, reminders will be displayed during the cooking process. Test table is a similar table as course table. User needs to pass some tests to get their certificate and there will not be reminders during the test cooking process.

2.1 Enrollment

The user’s enrollment status will be updated to pass if the user pass all tests connect to the certificate.



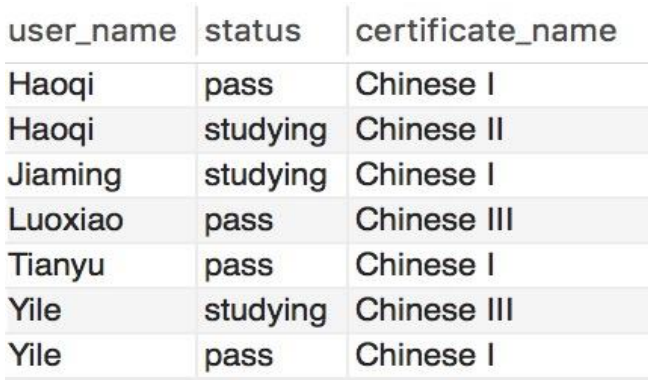
certificate_name	certificate_des
Chinese food Lv1	beginner of Chinese food
Chinese food Lv2	general of Chinese food
Chinese food Lv3	Master of Chinese food
French meal Lv1	beginner of French meal
French meal Lv2	general of French meal
French meal Lv3	Master of French meal
Thai food Lv1	beginner of Thai food
Thai food Lv2	general of Thai food
Thai food Lv3	Master of Thai food

Figure 3-2 list of certificate

First of all we need to show all the certificates and description to users.

User can choose the cooking style they want to learn, Chinese food, French meal, Thai food or something else.

When a user selects a certificate, an enrollment will be generated automatically. Enrollment table is the table between user and certificate, it can show which and how many certificate a user has selected.

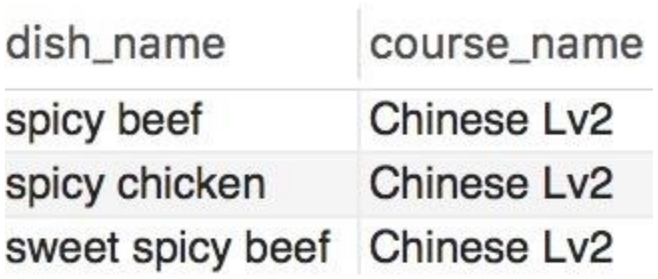


user_name	status	certificate_name
Haoqi	pass	Chinese I
Haoqi	studying	Chinese II
Jiaming	studying	Chinese I
Luoxiao	pass	Chinese III
Tianyu	pass	Chinese I
Yile	studying	Chinese III
Yile	pass	Chinese I

Figure 2-3 status of enrollment

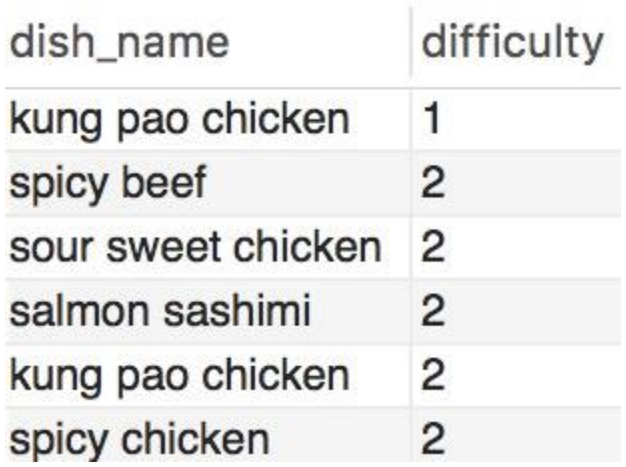
2.2 Dish and Course

If user wants to get a certificate, we provide many courses to help the user to learn how to cook the dishes of this certificate level. User can select the course by searching the dishes of the course. We set different difficulty for different dishes, so user can search the dishes by name or difficulty.



dish_name	course_name
spicy beef	Chinese Lv2
spicy chicken	Chinese Lv2
sweet spicy beef	Chinese Lv2

Figure 2-4 spicy dishes

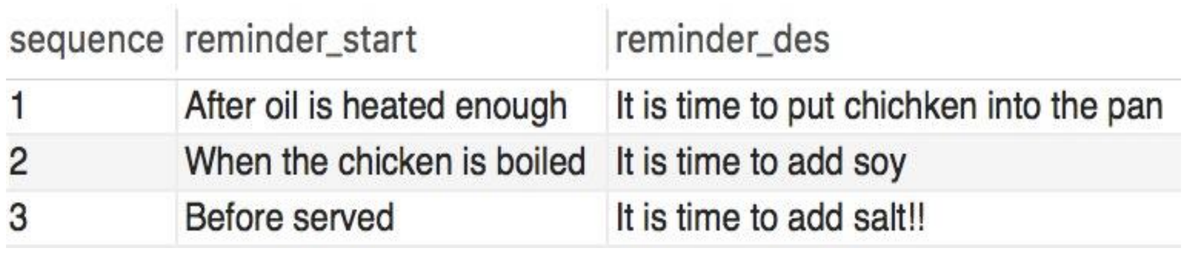


dish_name	difficulty
kung pao chicken	1
spicy beef	2
sour sweet chicken	2
salmon sashimi	2
kung pao chicken	2
spicy chicken	2

Figure 2-5 difficulty in (1,2)

2.3 Reminder

Of course, the most important thing of a course is to remind the users what they need to do during the cooking process. We will show some reminders. The reminders will be displayed in sequence order. And we set the reminders' start time and end time to determine the displayer period.



sequence	reminder_start	reminder_des
1	After oil is heated enough	It is time to put chichken into the pan
2	When the chicken is boiled	It is time to add soy
3	Before served	It is time to add salt!!

Figure 2-6 Reminders for Kung Pao Chicken

3. Cooking process

The most fundamental part of this project is to design the cooking process simulation. In this part, the point is to simulate the cooking evaluation and give an automatically feedback of every accomplishment done by users. All data will store in the database. Users can check what they have done and the grade of their accomplishments by selecting data in the database.

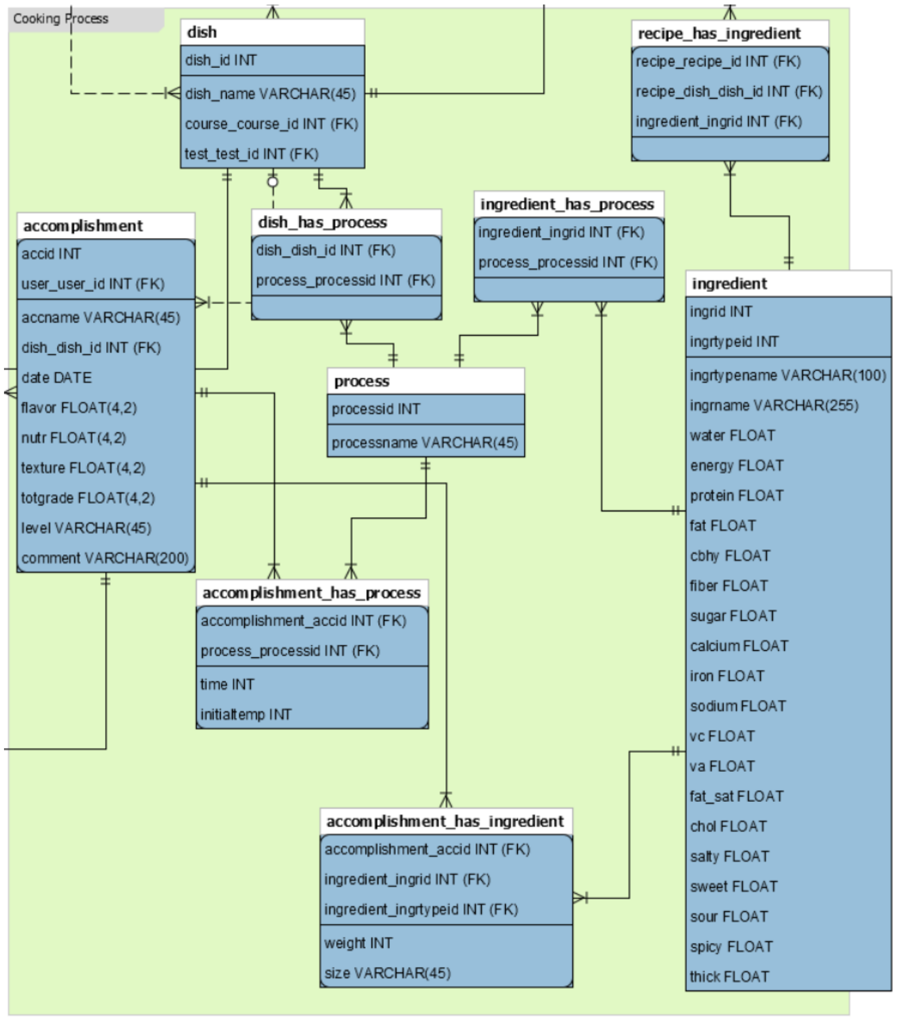


Figure 3-1 ER diagram of cooking process

There are 4 main entities used in the database: ingredient, accomplishment, process, and dish. Table 'Ingredient' contains all the nutrition and flavor information of vegetables, meats, spice and seasonings. Table 'Accomplishment' stores information about name, scores and comments of users' cooking accomplishment. Table 'process' stores all the cooking method in the kitchen. Table 'dish' stores information about the dished that people has known, such as kungpao chicken and clam chowder.

Among these entities, there are all M:N relationships. One ingredient can be a part of 1 to many accomplishments, one accomplishment consists 1 to many ingredients, that is M:N relationship. One ingredient can be cooked by 1 to many cooking process, one cooking process can cook 1 to many ingredients. One accomplishment can be cooked by one to many processes; one process can cook one to many accomplishments. Dish can be made by 1 to many processes; one process can cook one to many dishes.

3.1 Functions

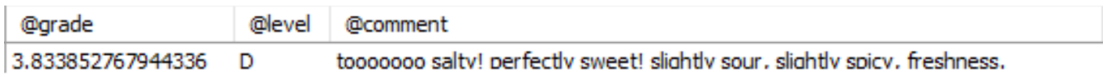
The main function of part of cooking process is to calculate the score of the accomplishment. The evaluation is divided into three subsets: flavor grading, nutrition grading and texture grading. Finally, these three parts will be combined into a final report to users, including score, level and final comment.

3.1.1 Flavor grading

Flavor is one of the most important standards to evaluate food level. Good flavor will bring nice feeling to people, and makes us enjoy the food. Abundant flavor will bring colorful experience of mouth.

An algorithm has been designed for this part. Every ingredient has attributes of flavor, such as salty, sweet, sour, spicy and thick. The value in the attribute means the intensity of the flavor in the ingredient. Using data about this, the score of flavor will be derived with algorithms.

Every accomplishment did by users have score for 5 flavors. The salty distribution of every accomplishment is derived from the content of sodium per 100g. The sweetness distribution is get from sugar content per 100g. The oil distribution is derived from fat content per 100g. The sour and spicy distribution is derived from average score per 100g. After getting these scores, the IF statement will judge the point interval. For example, if the salty distribution is in interval of 300 and 700, the salty score should be 10, the max point, and the comment will be perfectly salty. But if the salty distribution is in the range 0-300, the salty score will be 0 with comment 'no salt'. The other flavor distribution will experience the same process. Then the score of every flavor will be derived. Finally, the percentage of total flavor score is 60% for salty, 20% for sweet, 10% for thick and 5% for spicy and sour.



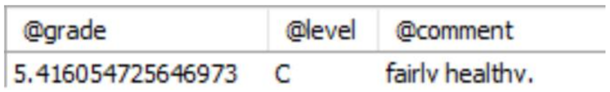
@grade	@level	@comment
3.833852767944336	D	tooooooo saltv! perfectlv sweet! sliahtlv sour. sliahtlv spicv. freshness.

Figure 3-2 Result of flavor grading

3.1.2 Nutrition grading

The second step is to evaluate the nutrition in the accomplishment. All attributes about nutrition in the ingredient table will be considered in the nutrition evaluation algorithm. In the table, content of several nutrition per 100g are listed. With these data, and the evaluation algorithm[2], the total nutrition grade will be derived. The maximum nutrition score should be 10 and the minimum should be 0.

Same as the first part, the distribution (per 100g) of every nutrition is calculated. Then, put the result into the IF statement. The level and comment will be given after executing. The levels are not healthy, not very healthy, fairly healthy, healthy and very healthy. If the total nutrition score is between 8 and 10, the nutrition level will be A, and the comment will be 'very healthy'.



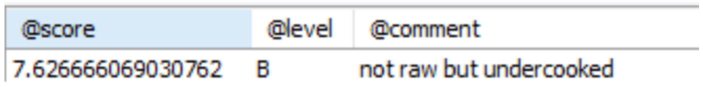
@grade	@level	@comment
5.416054725646973	C	fairlv healthv.

Figure 3-2 Results of nutrition grading

3.1.3 Texture grading

Last but not least, the texture evaluation decides whether the ingredients well-cooked or not. Ingredients with good texture tastes really well but bad texture will destroy the ingredients. It is really important to control the texture of every ingredient in the cooking accomplishments.

Here comes the algorithm of texture evaluation. There are 5 factors that influence the ingredient texture: size of ingredient, cooking method, initial temperature, and type of ingredient and cooking time. A new parameter is introduced which is called speed of heat diffusion. All the factors can be quantified to real number, for example, boiling for 1.5, frying for 3, baking for 2, steaming for 1 and grilling for 3. Another example is for size. IF small, THEN 1, and IF medium, THEN 2, ELSE 3. By multiply these numbers and cooking times, the parameter has already derived. With the number, the algorithm can judge what texture it can be. There are 5 levels coordinate the value of parameter. These levels are raw, not raw but under-cooked, well-cooked, over-cooked and charcoal. For example, if the parameter is in the range of 25-30, this is the best texture for food, and the comment is well-cooked.

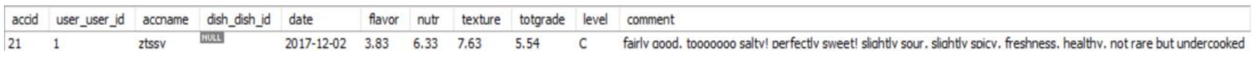


@score	@level	@comment
7.626666069030762	B	not raw but undercooked

Figure 3-3 Results of text grading

3.1.4 Total grading

This is the final part of the grading system. All the scores derived by algorithms above are integrated into one final score, one final level and one final comment. The stored procedure designed above will be called in this stored procedure. Then the results will help determine the final grade of the accomplishment. The percentage of grading is 50% for flavor, 15% for nutrition and 35% for texture score. And final level and comment will be given to users. The A level needs total score of 8-10, with comment ‘excellent!’.
After calculation, all scores and comments will inserted into table ‘accomplishment’ according to accomplishment ID that user input.



accid	user_user_id	accname	dish_dish_id	date	flavor	nutr	texture	totgrade	level	comment
21	1	ztssv	NULL	2017-12-02	3.83	6.33	7.63	5.54	C	fairly good, tooooooo salty! perfectly sweet! slightly sour, slightly spicy, freshness, healthy, not rare but undercooked

Figure 3-4 Final result of total grading

3.2 The flow of cooking

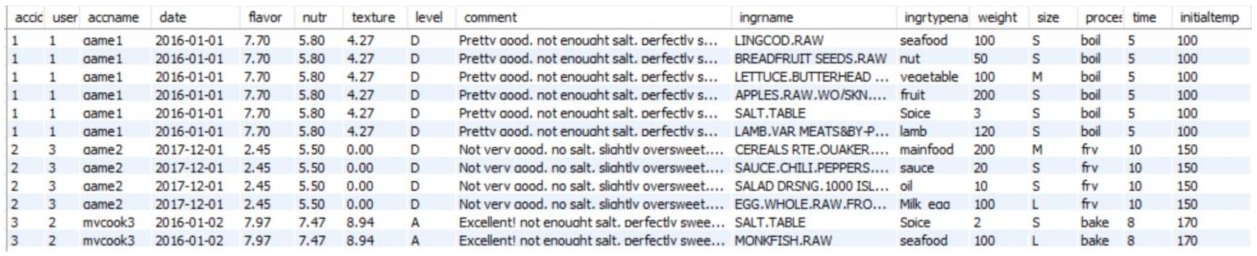
The flow of cooking process is listed below.

1. Create an accomplishment and add information including cooking process, initial temperature and accomplishment name. A unique accomplishment ID will be given (stored procedure ‘addacc’).
2. Insert ingredients into the accomplishment, according to accomplishment ID. Information of ingredients that will be cooked should also be inserted, including size and weight of every ingredient (stored procedure ‘addingr’).
3. Calculation the scores according to information of accomplishment. Grades and comments will be given (stored procedure ‘gradeflavor’,’gradenutr’ and ‘gradetexture’)
4. Total evaluation including final score, level and comment will be given (stored procedure ‘totgrade’).

The input of the grading system is name of accomplishment, cooking method, cooking time, initial temperature, and ingredients. The output will be a unique accomplishment ID, all scores, comments, and level. These data can be also used in other part of this system, including teaching and gaming process. For example, if one cooking game begins, an accomplishment ID will generated simultaneously. Then players may add other elements in the cooking process such as ingredients and cooking method.

3.3 Views and triggers

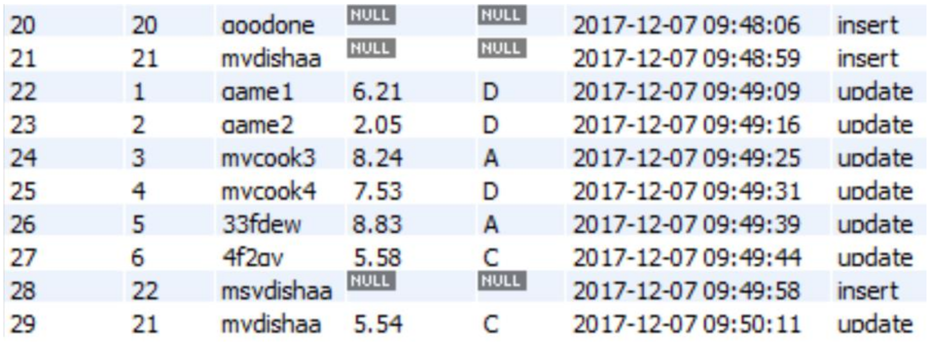
A view for accomplishment information is designed. It consists of all accomplishments that have been finished by all users. This view includes name, date, all scores, comments and level for the completed accomplishments. If someone want to see all accomplishments that one user has completed, use the WHERE clause.



accic	user	accname	date	flavor	nutr	texture	level	comment	ingrname	ingrtypena	weight	size	proces	time	initialtemp
1	1	oame1	2016-01-01	7.70	5.80	4.27	D	Prettv aood. not enouaht salt. perfectlv s...	LINGCOD.RAW	seafood	100	S	boil	5	100
1	1	oame1	2016-01-01	7.70	5.80	4.27	D	Prettv aood. not enouaht salt. perfectlv s...	BREADFRUIT SEEDS.RAW	nut	50	S	boil	5	100
1	1	oame1	2016-01-01	7.70	5.80	4.27	D	Prettv aood. not enouaht salt. perfectlv s...	LETTUCE.BUTTERHEAD ...	veaetable	100	M	boil	5	100
1	1	oame1	2016-01-01	7.70	5.80	4.27	D	Prettv aood. not enouaht salt. perfectlv s...	APPLES.RAW.WO/SKN....	fruit	200	S	boil	5	100
1	1	oame1	2016-01-01	7.70	5.80	4.27	D	Prettv aood. not enouaht salt. perfectlv s...	SALT.TABLE	Spice	3	S	boil	5	100
1	1	oame1	2016-01-01	7.70	5.80	4.27	D	Prettv aood. not enouaht salt. perfectlv s...	LAMB.VAR MEATS&BY-P...	lamb	120	S	boil	5	100
2	3	oame2	2017-12-01	2.45	5.50	0.00	D	Not verv aood. no salt. sliahtlv oversweet....	CEREALS RTE.OUAKER....	mainfood	200	M	frv	10	150
2	3	oame2	2017-12-01	2.45	5.50	0.00	D	Not verv aood. no salt. sliahtlv oversweet....	SAUCE.CHILI.PEPPERS....	sauce	20	S	frv	10	150
2	3	oame2	2017-12-01	2.45	5.50	0.00	D	Not verv aood. no salt. sliahtlv oversweet....	SALAD DRSNG.1000 ISL...	oil	10	S	frv	10	150
2	3	oame2	2017-12-01	2.45	5.50	0.00	D	Not verv aood. no salt. sliahtlv oversweet....	EGG.WHOLE.RAW.FRO...	Milk eaa	100	L	frv	10	150
3	2	mvcook3	2016-01-02	7.97	7.47	8.94	A	Excellent! not enouaht salt. perfectlv swee...	SALT.TABLE	Spice	2	S	bake	8	170
3	2	mvcook3	2016-01-02	7.97	7.47	8.94	A	Excellent! not enouaht salt. perfectlv swee...	MONKFISH.RAW	seafood	100	L	bake	8	170

Figure 3-5 View of accomplishment information

For documenting the history of accomplishment inserting and updating, two triggers are designed. After a unique accomplishment ID is established, trigger will add the documentation that an accomplishment has been created into log. When scores and comments are added, the trigger will add the documentation that the accomplishment has been finished into log.



20	20	aoodone	NULL	NULL	2017-12-07 09:48:06	insert
21	21	mvdishaa	NULL	NULL	2017-12-07 09:48:59	insert
22	1	oame1	6.21	D	2017-12-07 09:49:09	update
23	2	oame2	2.05	D	2017-12-07 09:49:16	update
24	3	mvcook3	8.24	A	2017-12-07 09:49:25	update
25	4	mvcook4	7.53	D	2017-12-07 09:49:31	update
26	5	33fdew	8.83	A	2017-12-07 09:49:39	update
27	6	4f2av	5.58	C	2017-12-07 09:49:44	update
28	22	msvdishaa	NULL	NULL	2017-12-07 09:49:58	insert
29	21	mvdishaa	5.54	C	2017-12-07 09:50:11	update

Figure 3-6 Result of trigger execution

All the code will be listed in the appendix.

4. Game mode

4. 1 Functions

This part is built on the base of cooking process and certificate. It still use the frame which build by Xiao Luo and Tianyu, but , if will offers more to a user, it will contains certain background story along with the cooking process, and it will collect the performance(grade) in every game and come up with a total grade to show the accomplishment of a user.

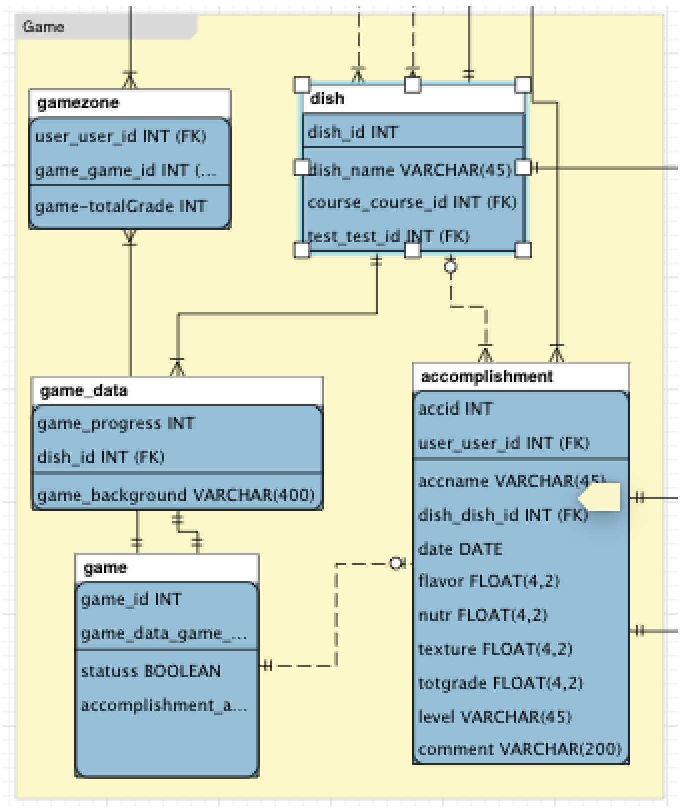
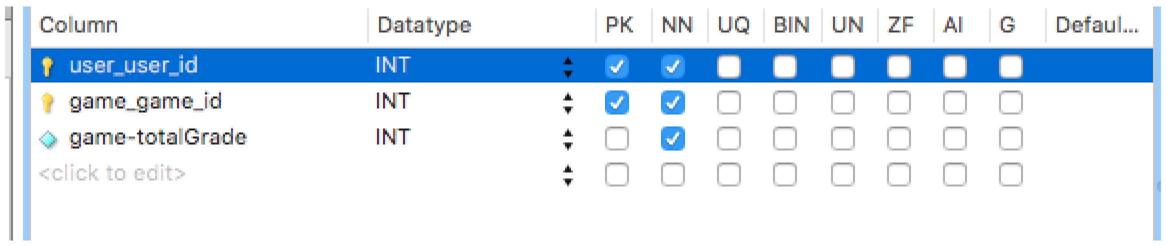


Figure 4-1 EER diagram for gamemode

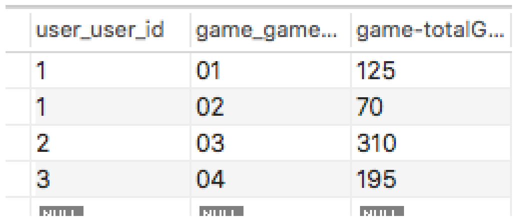
As you can see in the Figure4-1, there are actually three entities in this part, but this part also has connections to the certificate dish table and cooking processing accomplishment mode.

The first entity is game zone table, which contains the game_id that belong to a certain user, also it shows the total grade of each game_id.



Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Defaul...
user_user_id	INT	✓	✓							
game_game_id	INT	✓	✓							
game-totalGrade	INT		✓							
<click to edit>										

Figure 4-2 attributes in gamezone

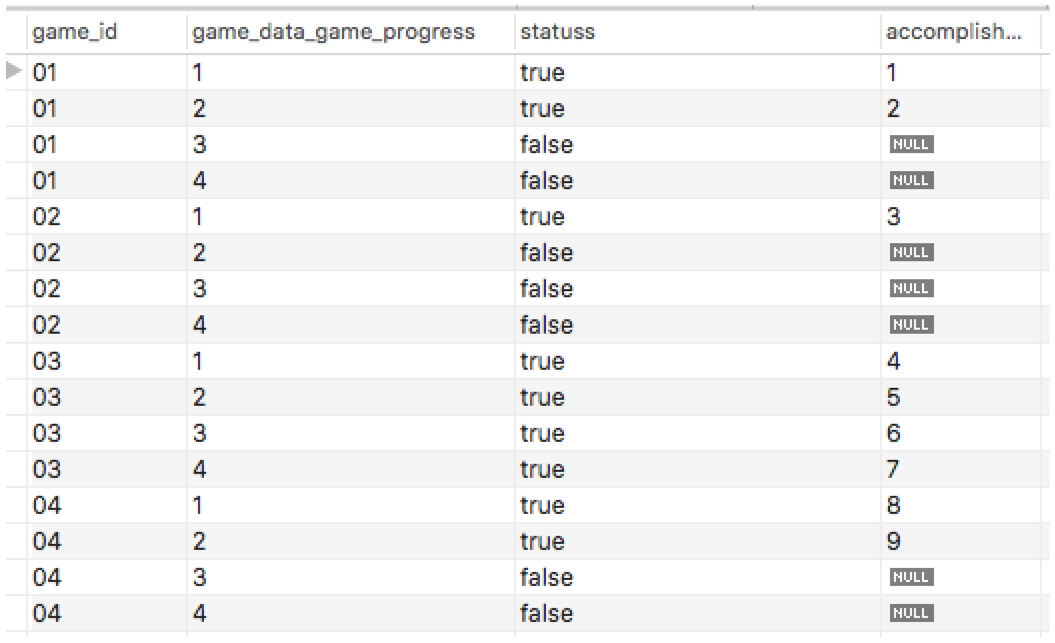


user_user_id	game_game...	game-totalG...
1	01	125
1	02	70
2	03	310
3	04	195
NULL	NULL	NULL

Figure 4-3 values of gamezone

Figure 4-2 and 4-3 are attributes and values of gamezone table.

The second entity is game: it contains the data of all the game_id. In our design, game mode has 4 parts, and the total grade of a game_id will be the sum of every grade of each part.

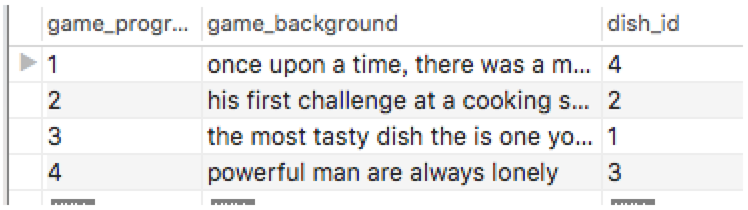


game_id	game_data_game_progress	statuss	accomplish...
01	1	true	1
01	2	true	2
01	3	false	NULL
01	4	false	NULL
02	1	true	3
02	2	false	NULL
02	3	false	NULL
02	4	false	NULL
03	1	true	4
03	2	true	5
03	3	true	6
03	4	true	7
04	1	true	8
04	2	true	9
04	3	false	NULL
04	4	false	NULL

Figure 4-4 dataset of game table

Figure 4-4 shows the table's attribute and the value of them, for each game_id, it has 4 game_progress, and for each progress, it has a status to show whether the user finish it or not, and if it finished, there also will be an accomplishment_id link to it.

The third entity is called game_data. It contains the dish_id of each progress and the background of it.



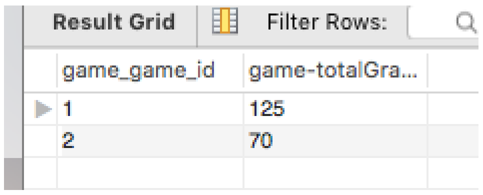
game_progr...	game_background	dish_id
1	once upon a time, there was a m...	4
2	his first challenge at a cooking s...	2
3	the most tasty dish the is one yo...	1
4	powerful man are always lonely	3

Figure 4-5 dataset of game_data table

Figure 4-5 shows attributes and the value of them for each game_progress. It stores a unique story background of it and a dish_id. Through the dish_id, we can find the instructions of it via the part we mentioned before.

4.2 Store Procedures

When the user login the game mode, he or she should see all the game files they have and their grades, so we write a procedure to find them by user_id.



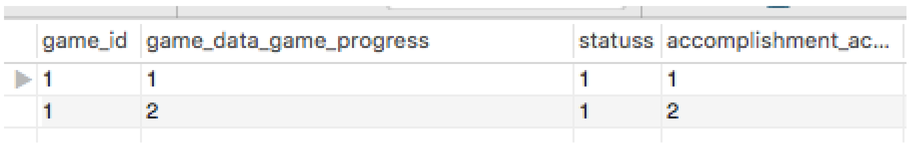
Result Grid Filter Rows:

game_game_id	game-totalGra...
1	125
2	70

Figure 4-6 The result of store procedure1 for user1

Figure 4-6 shows the all the game-id of user who has user_id equals 1, when the user see the records the screen, he can choose a file to continue, or delete it. He can also start a new game and get a new game id.

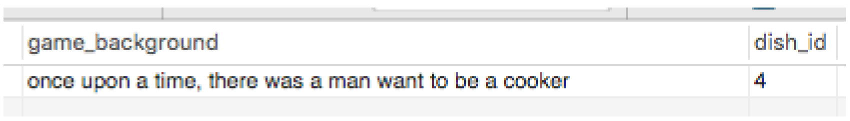
After the user choose a certain game file, he or she should be able to see the detail of that file. which means they should see what they have achieved. Figure 4-7 shows the detail of the game_id 1, we can see the status of progress 1,2 is true. And the status is a Boolean ,1 means the user already finish that progress and get a complete work for it. The accomplishment_id represent the outcome the game, we can use it find the score in the accomplishment table.



game_id	game_data_game_progress	statuss	accomplishment_ac...
1	1	1	1
1	2	1	2

Figure 4-7 The result of store procedure2 for game 1

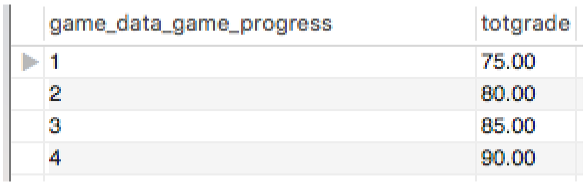
When the user chooses a progress, the system should return the background and the instructions of it. For example, if the user chooses the 4th progress, the system should find out what background is linked to it and the dish user should cook during the game. Figure 4-8 shows the background and dish_id for progress 4.



game_background	dish_id
once upon a time, there was a man want to be a cooker	4

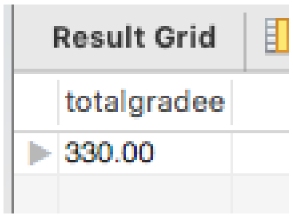
Figure 4-8 The result of store procedure3 for progress 3

The total grade in the gamezone table is the sum of the grades for each part, When we have a game_id we should be able to calculate the totalgrade of it. Figure 4-9 shows the grade of game 3 We have to add them together to get the total grade as it shown in the Figure4-10.



game_data_game_progress	totgrade
1	75.00
2	80.00
3	85.00
4	90.00

Figure 4-9 The result of store procedure4 for game3



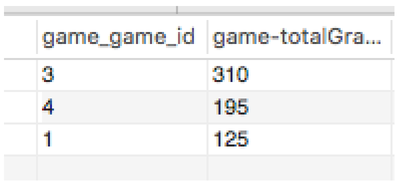
Result Grid

totalgradee
330.00

Figure 4-10 The result of store procedure5 for user3

4.3 View

The user should be able to see the top 3 total grades above all users.



game_game_id	game-totalGra...
3	310
4	195
1	125

Figure 4-11The view of Top 3 player

The admin can see all the users who finish all the progress

4.4 Trigger

when a user finish a new game, it will update the data in the game table, once the game table is change, the total grade of a game id will be change too, so to make sure the data is correct , I write a trigger to update the totalgrade in the gamezone table.It hard to show the result, I will include my code in the back.

5. Social Networking

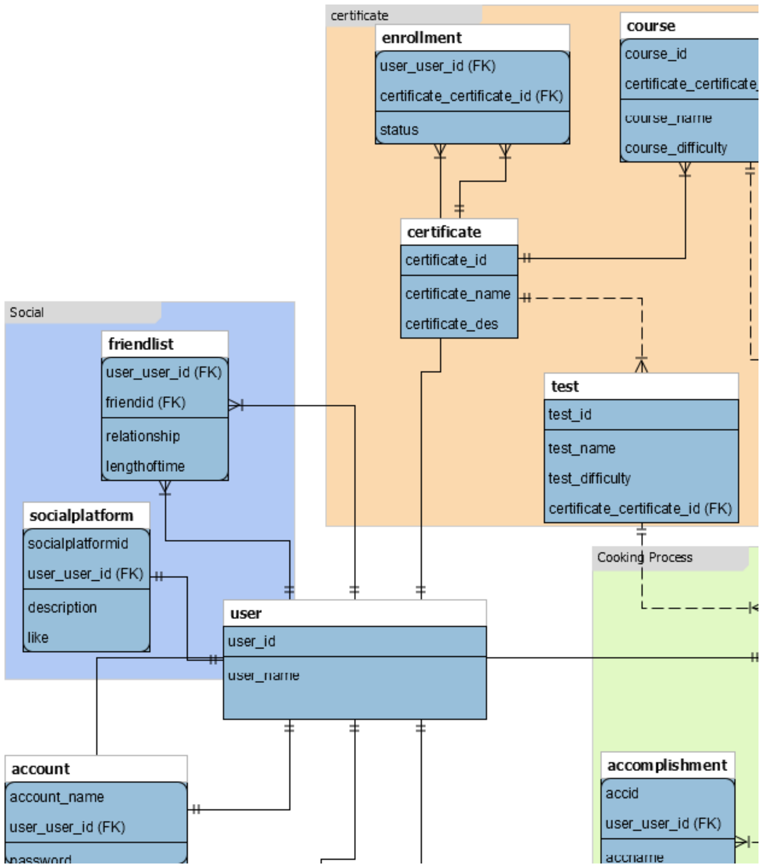


Figure 5-3 ERD for social networking

There are three main entities works for social networking, but queries can also extract information from other entities connected with user.

In detail, social platform is a zone of each user: it has the description about how its user wants to describe the zone. And the like means the number of people like his or her zone.one user only has one socialplatform. And because one person can have one to many friends, the relationship between user and friendlist is M:N. What we must mention is that, people's friend can also found from the big user table.

5.1 Friends

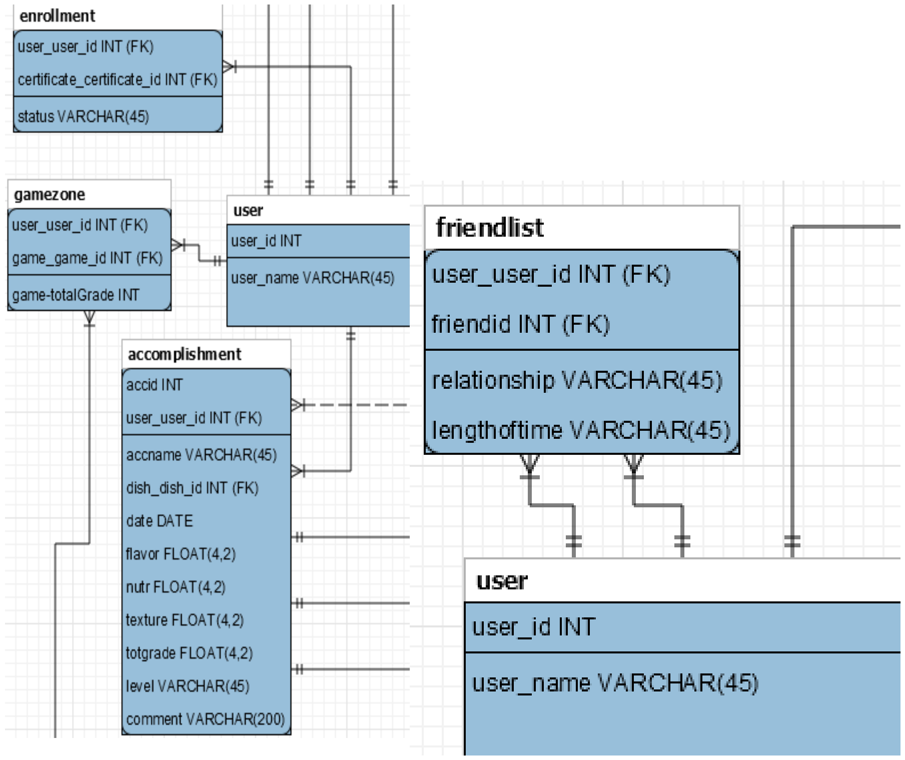
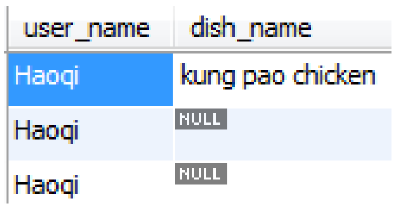


Figure 5-2 Friend list and zones

Making friends is creating M:N relationships among users. Appendix I-5 shows how to view all friends for specific person.

5.2 Personal Zone

The relationships between user-games, user-certifications and user- dishes are M:N as well. All personal zone are derived tables that are used for storing using history. Appendix I-5 shows how to use accomplishment table to search all dishes a user have cooked or remaining to be done.

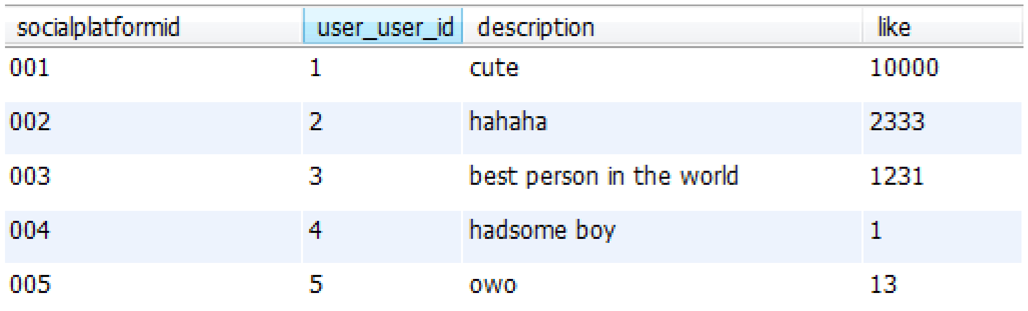


user_name	dish_name
Haoqi	kung pao chicken
Haoqi	NULL
Haoqi	NULL

Figure 5-3 dish search

As the user played game or attend course, there are something need to be changed in enrollment, gamezone and accomplishment. For example, if an app developer updates the game part like changing name, gamezone should be changed as well. What's more, if a certification program is cancelled, it should not be presented in the enrollment. To solve this kind of problems,3 kinds of triggers are created.

5.3. Social platform

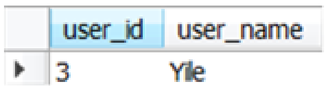


socialplatformid	user_user_id	description	like
001	1	cute	10000
002	2	hahaha	2333
003	3	best person in the world	1231
004	4	hadsome boy	1
005	5	owo	13

Figure 5-2 data stored in social platform

The social networking has following features:

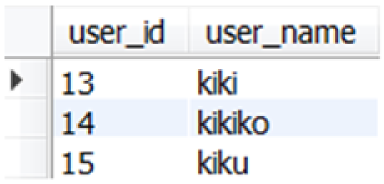
1) Search exact person's by eaxact ID or approximate name.



user_id	user_name
3	Yile

Figure 5-3 results by searching ID

If we already know a person's ID, we can know her name.

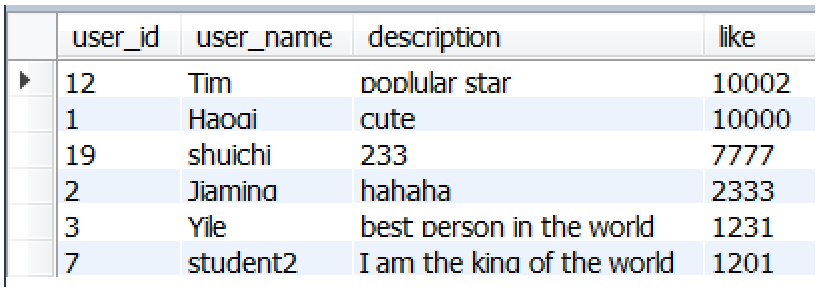


user_id	user_name
13	kiki
14	kikiko
15	kiku

Figure 5-3 results by searching name

If we only know a person's partial name, for example, Ki, we can get all user's id which have the information we want search. Since user_id and socialplatform_id is one to one relationship, we can easily find the id of zone via getting user's user_id.

2) Popular user in our system

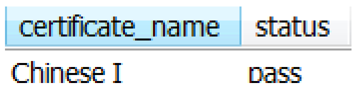


user_id	user_name	description	like
12	Tim	poplular star	10002
1	Haoqi	cute	10000
19	shuichi	233	7777
2	Jiaming	hahaha	2333
3	Yile	best person in the world	1231
7	student2	I am the king of the world	1201

Figure 4-4 results of searching popular star

Our platform encourages users to cook the dish and pass the certificate. More achievement they got, more probability they are going to receive like by other users. This function generates users who have most like in our system. We define people who have like over 1000 can be popular user and the search result is sort by decreasing order.

3) view certificates user has passed



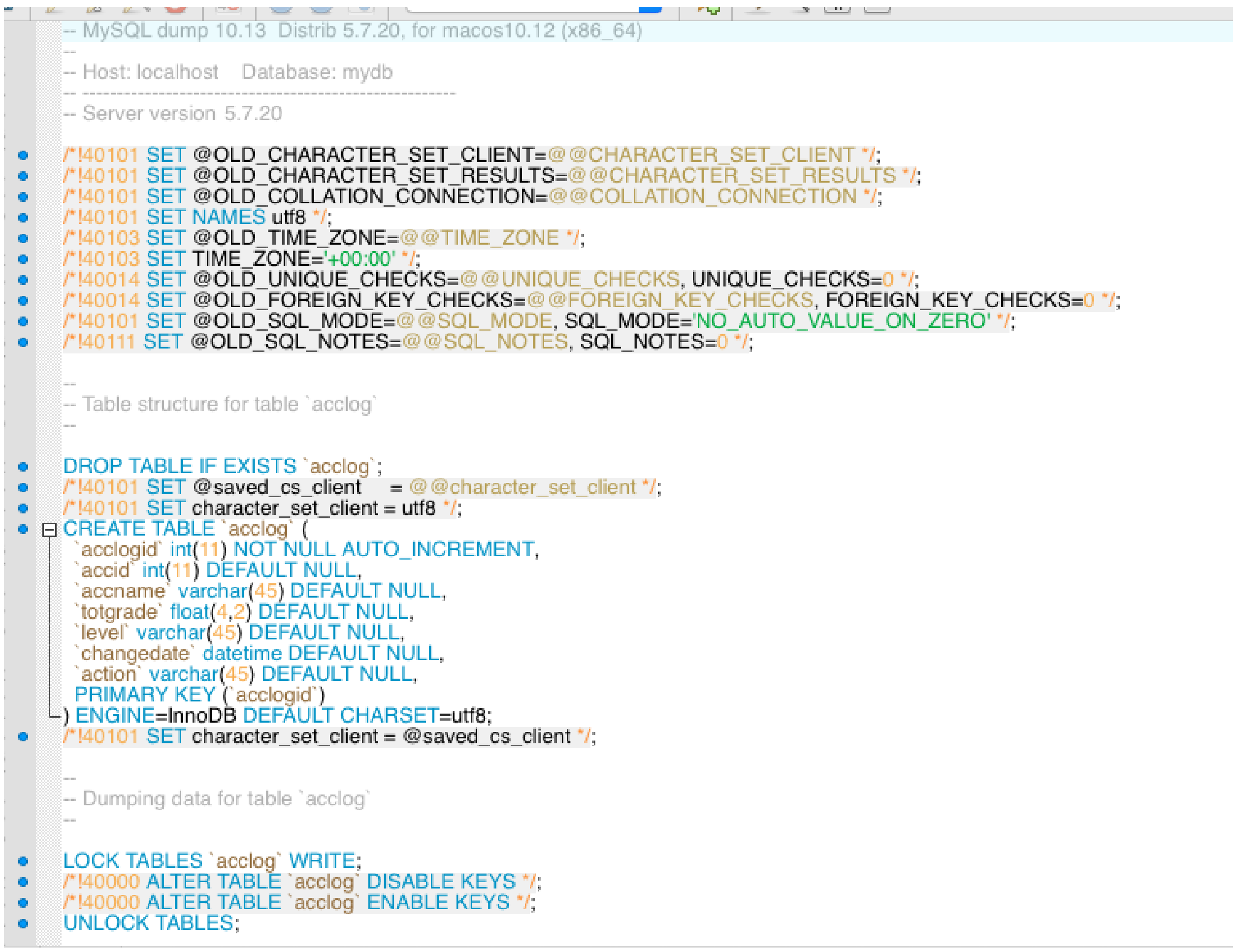
certificate_name	status
Chinese I	pass

Figure 5-5 results of showing the certificate Yile pass

At the time of viewing other people's page, people can also view all certificates this people passed. For example, Figure 5-5 demonstrates the certificate that Yile' passed. Actually, Yile passed Chinese I and is learning Japanese I this time. But in order to protect the privacy, people are only going to see the certificate has the passed status.
With the same method, the social platform can also view user's game score if necessary.

6. Backup

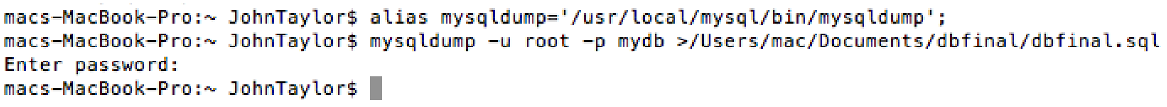
We use the mysqldump to backup and recover our database.



```
-- MySQL dump 10.13  Distrib 5.7.20, for macos10.12 (x86_64)
--
-- Host: localhost    Database: mydb
-- ------------------------------------------------------
-- Server version 5.7.20

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `acclog`
--

DROP TABLE IF EXISTS `acclog`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `acclog` (
  `acclogid` int(11) NOT NULL AUTO_INCREMENT,
  `accid` int(11) DEFAULT NULL,
  `accname` varchar(45) DEFAULT NULL,
  `totgrade` float(4,2) DEFAULT NULL,
  `level` varchar(45) DEFAULT NULL,
  `changedate` datetime DEFAULT NULL,
  `action` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`acclogid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `acclog`
--

LOCK TABLES `acclog` WRITE;
/*!40000 ALTER TABLE `acclog` DISABLE KEYS */;
/*!40000 ALTER TABLE `acclog` ENABLE KEYS */;
UNLOCK TABLES;
```

Figure 6-1

Here is a screenshot of the backup file, we use the mysqldump sentence to generate it, first we have to find how to access the mysqldump file, it locate at /usr/local/mysql/bin/mysqldump in mac. Then we give it a alias to access more easily:
alias mysqldump='/usr/local/mysql/bin/mysqldump', after that , we can use the mysqldump to backup it: mysqldump -u root -p mydb >/Users/mac/Documents/dbfinal/dbfinal.sql
After we input the password, the backup will generate automatically



```
macs-MacBook-Pro:~ JohnTaylor$ alias mysqldump='/usr/local/mysql/bin/mysqldump';
macs-MacBook-Pro:~ JohnTaylor$ mysqldump -u root -p mydb >/Users/mac/Documents/dbfinal/dbfinal.sql
Enter password:
macs-MacBook-Pro:~ JohnTaylor$
```

Figure 6-2

7. Conclusion

To saving people who feel pain in cooking, in this paper, we designed a system that can simulate the cooking process in the kitchen on Mysql. Based on simulation module, some functions are developed including social platform, certificate, training, gaming. To achieve the functions, we designed ER diagram with 22 tables, including ingredients, cooking methods, certificates, friend lists and social functions. Then we designed the algorithms that can calculate the evaluation of each accomplishments. Then we developed recipes, dishes, friend lists based on ER diagram. Finally, we designed a front-end based on website. With this VR kitchen system, many people can practice their cooking skills without feeling of painful, and they may love cooking for their families. Natural resources can be saved more. Also, many cooking lovers may be connected. They may share their accomplishments with friends.

There are still several points that we need to improve. First, we need to develop a front-end that satisfies daily use for people. Second, we need to introduce more cooking process and more accurate simulation algorithms. Moreover, many new technologies may be used in this system.

Reference

1. https://www.ars.usda.gov/northeast-area/beltsville-md/beltsville-human-nutrition-research-center/nutrient-data-laboratory/docs/usda-national-nutrient-database-for-standard-reference/

2. Martin, Jolie M., et al. "Modeling expert opinions on food healthfulness: a nutrition metric." *Journal of the American Dietetic Association* 109.6 (2009): 1088-1091.

3. Hoffer, Jeff, Ramesh Venkataraman, and Heikki Topi. *Modern database management*. Prentice Hall Press, 2015.

4. Vitrac, Olivier, Gilles Trystram, and Anne‐Lucie Raoult‐Wack. "Deep‐fat frying of food: Heat and mass transfer, transformations and reactions inside the frying material." *European Journal of Lipid Science and Technology* 102.8‐9 (2000): 529-538.

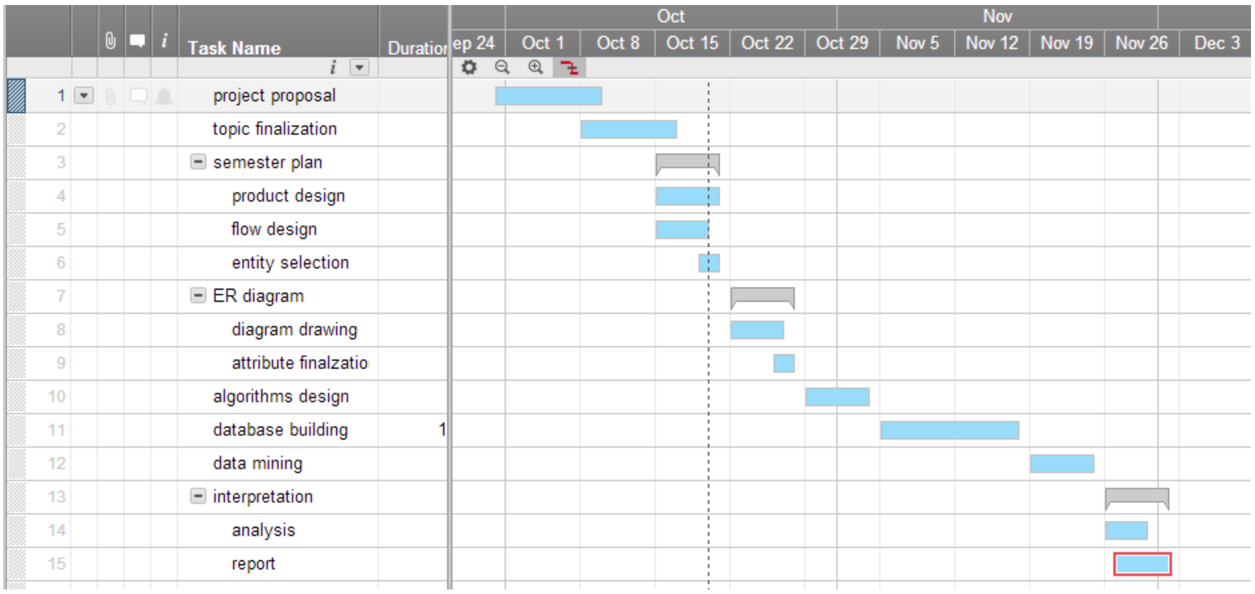
5. Virtual Cooking with Virtual Axel. https://www.youtube.com/watch?v=ywFBrcLs2M4

Appendix I Project Management

Member's Work

Yile Peng: sequential diagram, Gantt Chart, social platform, formatting
Haoqi Huang: user related parts,2 triggers and 2 procedures, use case and front-end design.
Tianyu Zhong: Establish ER diagram of the part of cooking process. Designed 5 stored procedures, 3 triggers, 1 view. Designed the grading algorithms of accomplishments. Tested the schema, queries, subqueries, triggers, views. Inserted data for ingredients, process, accomplishment, dish. Tested the foreign key references.
Jiaming Duan: game mode. Backup.

Gantt Chart



	Task Name	Duratior
1	project proposal	
2	topic finalization	
3	semester plan	
4	product design	
5	flow design	
6	entity selection	
7	ER diagram	
8	diagram drawing	
9	attribute finalzatio	
10	algorithms design	
11	database building	1
12	data mining	
13	interpretation	
14	analysis	
15	report	

Figure 7-1 Gantt Chart

As the Gantt shows, we started the brainstorming of project at the beginning of the semester and did the presentation on Dec 4th. Here, we finish the final part of report in Dec 10th.

Appendix II: Code

Schema of database:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `mydb` ;

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`user`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`user` ;

CREATE TABLE IF NOT EXISTS `mydb`.`user` (
  `user_id` INT NOT NULL,
  `user_name` VARCHAR(45) NULL,
  PRIMARY KEY (`user_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`certificate`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`certificate` ;

CREATE TABLE IF NOT EXISTS `mydb`.`certificate` (
  `certificate_id` INT NOT NULL,
  `certificate_name` VARCHAR(45) NULL,
  `certificate_des` VARCHAR(45) NULL,
  PRIMARY KEY (`certificate_id`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`enrollment`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`enrollment` ;

CREATE TABLE IF NOT EXISTS `mydb`.`enrollment` (
  `user_user_id` INT NOT NULL,
  `certificate_certificate_id` INT NOT NULL,
  `status` VARCHAR(45) NULL,
  PRIMARY KEY (`user_user_id`, `certificate_certificate_id`),
  INDEX `fk_certificate_has_user_user1_idx` (`user_user_id` ASC),
  INDEX `fk_certificate_has_user_certificate_idx`
(`certificate_certificate_id` ASC),
  CONSTRAINT `fk_certificate_has_user_certificate`
    FOREIGN KEY (`certificate_certificate_id`)
    REFERENCES `mydb`.`certificate` (`certificate_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_certificate_has_user_user1`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`course`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`course` ;

CREATE TABLE IF NOT EXISTS `mydb`.`course` (
  `course_id` INT NOT NULL,
  `certificate_certificate_id` INT NOT NULL,
  `course_name` VARCHAR(100) NULL,
  `course_difficulty` INT NULL,
  PRIMARY KEY (`course_id`, `certificate_certificate_id`),
  INDEX `fk_course_certificate1_idx` (`certificate_certificate_id` ASC),
  CONSTRAINT `fk_course_certificate1`
    FOREIGN KEY (`certificate_certificate_id`)
    REFERENCES `mydb`.`certificate` (`certificate_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`test`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`test` ;

CREATE TABLE IF NOT EXISTS `mydb`.`test` (
  `test_id` INT NOT NULL,
  `test_name` VARCHAR(45) NULL,
  `test_difficulty` VARCHAR(45) NULL,
  `certificate_certificate_id` INT NOT NULL,
  PRIMARY KEY (`test_id`),
  INDEX `fk_test_certificate1_idx` (`certificate_certificate_id` ASC),
  CONSTRAINT `fk_test_certificate1`
    FOREIGN KEY (`certificate_certificate_id`)
    REFERENCES `mydb`.`certificate` (`certificate_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`process`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`process` ;

CREATE TABLE IF NOT EXISTS `mydb`.`process` (
  `processid` INT NOT NULL,
  `processname` VARCHAR(45) NULL,
  PRIMARY KEY (`processid`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`dish`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`dish` ;

CREATE TABLE IF NOT EXISTS `mydb`.`dish` (
  `dish_id` INT NOT NULL,
  `dish_name` VARCHAR(45) NULL,
  `course_course_id` INT NOT NULL,
  `test_test_id` INT NOT NULL,
  `process_processid` INT NOT NULL,
  PRIMARY KEY (`dish_id`, `process_processid`),
  INDEX `fk_dish_course1_idx` (`course_course_id` ASC),
  INDEX `fk_dish_test1_idx` (`test_test_id` ASC),
  INDEX `fk_dish_process1_idx` (`process_processid` ASC),
  CONSTRAINT `fk_dish_course1`
    FOREIGN KEY (`course_course_id`)
    REFERENCES `mydb`.`course` (`course_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_dish_test1`
    FOREIGN KEY (`test_test_id`)
    REFERENCES `mydb`.`test` (`test_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_dish_process1`
    FOREIGN KEY (`process_processid`)
    REFERENCES `mydb`.`process` (`processid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`recipe`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`recipe` ;

CREATE TABLE IF NOT EXISTS `mydb`.`recipe` (
  `recipe_id` INT NOT NULL,
  `recipe_name` VARCHAR(45) NULL,
  `recipe_description` VARCHAR(45) NULL,
  `dish_dish_id` INT NOT NULL,
  `difficulty` INT NOT NULL,
  PRIMARY KEY (`recipe_id`, `dish_dish_id`),
  INDEX `fk_recipe_dish1_idx` (`dish_dish_id` ASC),
  CONSTRAINT `fk_recipe_dish1`
    FOREIGN KEY (`dish_dish_id`)
    REFERENCES `mydb`.`dish` (`dish_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`reminder`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `mydb`.`reminder` ;

CREATE TABLE IF NOT EXISTS `mydb`.`reminder` (
  `reminder_id` INT NOT NULL,
  `reminder_name` VARCHAR(45) NULL,
  `reminder_des` VARCHAR(45) NULL,
  `reminder_start` VARCHAR(45) NULL,
  `remider_end` VARCHAR(45) NULL,
  `recipe_recipe_id` INT NOT NULL,
  `sequence` INT NULL,
  PRIMARY KEY (`reminder_id`, `recipe_recipe_id`),
  INDEX `fk_reminder_recipe1_idx` (`recipe_recipe_id` ASC),
  CONSTRAINT `fk_reminder_recipe1`
    FOREIGN KEY (`recipe_recipe_id`)
    REFERENCES `mydb`.`recipe` (`recipe_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`ingredient`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`ingredient` ;

CREATE TABLE IF NOT EXISTS `mydb`.`ingredient` (
  `ingrid` INT NOT NULL,
  `ingrtypeid` INT NOT NULL,
  `ingrtypename` VARCHAR(100) NULL,
  `ingrname` VARCHAR(255) NULL,
  `water` FLOAT NULL,
  `energy` FLOAT NULL,
  `protein` FLOAT NULL,
  `fat` FLOAT NULL,
  `cbhy` FLOAT NULL,
  `fiber` FLOAT NULL,
  `sugar` FLOAT NULL,
  `calcium` FLOAT NULL,
  `iron` FLOAT NULL,
  `sodium` FLOAT NULL,
  `vc` FLOAT NULL,
  `va` FLOAT NULL,
  `fat_sat` FLOAT NULL,
  `chol` FLOAT NULL,
  `salty` FLOAT NULL,
  `sweet` FLOAT NULL,
  `sour` FLOAT NULL,
  `spicy` FLOAT NULL,
  `thick` FLOAT NULL,
  PRIMARY KEY (`ingrid`, `ingrtypeid`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`account`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`account` ;

CREATE TABLE IF NOT EXISTS `mydb`.`account` (
  `account_name` VARCHAR(10) NOT NULL,
  `password` VARCHAR(255) NULL,
  `user_user_id` INT NOT NULL,
  PRIMARY KEY (`account_name`, `user_user_id`),
  INDEX `fk_account_user1_idx` (`user_user_id` ASC),
  CONSTRAINT `fk_account_user1`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`game_data`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`game_data` ;

CREATE TABLE IF NOT EXISTS `mydb`.`game_data` (
  `game_progress` INT NOT NULL,
  `game_background` VARCHAR(400) NOT NULL,
  `dish_id` INT NOT NULL,
  PRIMARY KEY (`game_progress`, `dish_id`),
  INDEX `fk_game_data_dish1_idx` (`dish_id` ASC),
  CONSTRAINT `fk_game_data_dish1`
    FOREIGN KEY (`dish_id`)
    REFERENCES `mydb`.`dish` (`dish_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`accomplishment`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`accomplishment` ;

CREATE TABLE IF NOT EXISTS `mydb`.`accomplishment` (
  `accid` INT NOT NULL AUTO_INCREMENT,
  `user_user_id` INT NOT NULL,
  `accname` VARCHAR(45) NULL,
  `dish_dish_id` INT NULL,
  `date` DATE NULL,
  `flavor` FLOAT(4,2) NULL,
  `nutr` FLOAT(4,2) NULL,
  `texture` FLOAT(4,2) NULL,
  `totgrade` FLOAT(4,2) NULL,
  `level` VARCHAR(45) NULL,
  `comment` VARCHAR(200) NULL,
  PRIMARY KEY (`accid`, `user_user_id`),
  INDEX `fk_accomplishment_user1_idx` (`user_user_id` ASC),
  INDEX `fk_accomplishment_dish1_idx` (`dish_dish_id` ASC),
  CONSTRAINT `fk_accomplishment_user1`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_accomplishment_dish1`
    FOREIGN KEY (`dish_dish_id`)
    REFERENCES `mydb`.`dish` (`dish_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`game`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`game` ;

CREATE TABLE IF NOT EXISTS `mydb`.`game` (
  `game_id` INT NOT NULL,
  `game_progress` INT NOT NULL,
  `accid` INT NOT NULL,
  `status` TINYINT(1) NOT NULL,
  `game_data_game_progress` INT NOT NULL,
  `game_data_dish_id` INT NOT NULL,
  `accomplishment_accid` INT NOT NULL,
  `accomplishment_user_user_id` INT NOT NULL,
  PRIMARY KEY (`game_id`, `game_data_game_progress`, `game_data_dish_id`),
  INDEX `fk_game_game_data1_idx` (`game_data_game_progress` ASC,
`game_data_dish_id` ASC),
  INDEX `fk_game_accomplishment1_idx` (`accomplishment_accid` ASC,
`accomplishment_user_user_id` ASC),
  CONSTRAINT `fk_game_game_data1`
    FOREIGN KEY (`game_data_game_progress` , `game_data_dish_id`)
    REFERENCES `mydb`.`game_data` (`game_progress` , `dish_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_game_accomplishment1`
    FOREIGN KEY (`accomplishment_accid` , `accomplishment_user_user_id`)
    REFERENCES `mydb`.`accomplishment` (`accid` , `user_user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`gamezone`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`gamezone` ;

CREATE TABLE IF NOT EXISTS `mydb`.`gamezone` (
  `user_user_id` INT NOT NULL,
  `game_game_id` INT NOT NULL,
  `game-totalGrade` INT NOT NULL,
  PRIMARY KEY (`user_user_id`, `game_game_id`),
  INDEX `fk_user_has_game_game1_idx` (`game_game_id` ASC),
  INDEX `fk_user_has_game_user1_idx` (`user_user_id` ASC),
  CONSTRAINT `fk_user_has_game_user1`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_has_game_game1`
    FOREIGN KEY (`game_game_id`)
    REFERENCES `mydb`.`game` (`game_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`friendlist`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`friendlist` ;

CREATE TABLE IF NOT EXISTS `mydb`.`friendlist` (
  `user_user_id` INT NOT NULL,
  `friendid` INT NOT NULL,
  `relationship` VARCHAR(45) NULL,
  `lengthoftime` VARCHAR(45) NULL,
  PRIMARY KEY (`user_user_id`, `friendid`),
  INDEX `fk_user_has_user_user4_idx` (`friendid` ASC),
  INDEX `fk_user_has_user_user3_idx` (`user_user_id` ASC),
  CONSTRAINT `fk_user_has_user_user3`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_has_user_user4`
    FOREIGN KEY (`friendid`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`friendlist`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`friendlist` ;

CREATE TABLE IF NOT EXISTS `mydb`.`friendlist` (
  `user_user_id` INT NOT NULL,
  `friendid` INT NOT NULL,
  `relationship` VARCHAR(45) NULL,
  `lengthoftime` VARCHAR(45) NULL,
  PRIMARY KEY (`user_user_id`, `friendid`),
  INDEX `fk_user_has_user_user4_idx` (`friendid` ASC),
  INDEX `fk_user_has_user_user3_idx` (`user_user_id` ASC),
  CONSTRAINT `fk_user_has_user_user3`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_has_user_user4`
    FOREIGN KEY (`friendid`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`friendlist`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`friendlist` ;

CREATE TABLE IF NOT EXISTS `mydb`.`friendlist` (
  `user_user_id` INT NOT NULL,
  `friendid` INT NOT NULL,
  `relationship` VARCHAR(45) NULL,
  `lengthoftime` VARCHAR(45) NULL,
  PRIMARY KEY (`user_user_id`, `friendid`),
  INDEX `fk_user_has_user_user4_idx` (`friendid` ASC),
  INDEX `fk_user_has_user_user3_idx` (`user_user_id` ASC),
  CONSTRAINT `fk_user_has_user_user3`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_has_user_user4`
    FOREIGN KEY (`friendid`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`socialplatform`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`socialplatform` ;

CREATE TABLE IF NOT EXISTS `mydb`.`socialplatform` (
  `socialplatformid` INT(10) NOT NULL,
  `user_user_id` INT NOT NULL,
  `description` VARCHAR(45) NULL,
  `like` INT NULL,
  PRIMARY KEY (`socialplatformid`, `user_user_id`),
  INDEX `fk_socialplatform_user1_idx` (`user_user_id` ASC),
  CONSTRAINT `fk_socialplatform_user1`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`user_has_socialplatform`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`user_has_socialplatform` ;

CREATE TABLE IF NOT EXISTS `mydb`.`user_has_socialplatform` (
  `user_user_id` INT NOT NULL,
  `user_socialplatform_socialplatformid` INT(10) NOT NULL,
  `socialplatform_socialplatformid` INT(10) NOT NULL,
  PRIMARY KEY (`user_user_id`, `user_socialplatform_socialplatformid`,
`socialplatform_socialplatformid`),
  INDEX `fk_user_has_socialplatform_socialplatform1_idx`
(`socialplatform_socialplatformid` ASC),
  INDEX `fk_user_has_socialplatform_user1_idx` (`user_user_id` ASC,
`user_socialplatform_socialplatformid` ASC),
  CONSTRAINT `fk_user_has_socialplatform_user1`
    FOREIGN KEY (`user_user_id`)
    REFERENCES `mydb`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_has_socialplatform_socialplatform1`
    FOREIGN KEY (`socialplatform_socialplatformid`)
    REFERENCES `mydb`.`socialplatform` (`socialplatformid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`ingredient_has_process`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`ingredient_has_process` ;

CREATE TABLE IF NOT EXISTS `mydb`.`ingredient_has_process` (
  `ingredient_ingrid` INT NOT NULL,
  `process_processid` INT NOT NULL,
  PRIMARY KEY (`ingredient_ingrid`, `process_processid`),
  INDEX `fk_ingredient_has_process_process1_idx` (`process_processid` ASC),
  INDEX `fk_ingredient_has_process_ingredient1_idx` (`ingredient_ingrid`
ASC),
  CONSTRAINT `fk_ingredient_has_process_ingredient1`
    FOREIGN KEY (`ingredient_ingrid`)
    REFERENCES `mydb`.`ingredient` (`ingrid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ingredient_has_process_process1`
    FOREIGN KEY (`process_processid`)
    REFERENCES `mydb`.`process` (`processid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`accomplishment_has_ingredient`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`accomplishment_has_ingredient` ;

CREATE TABLE IF NOT EXISTS `mydb`.`accomplishment_has_ingredient` (
  `accomplishment_accid` INT NOT NULL,
  `ingredient_ingrid` INT NOT NULL,
  `weight` INT NULL,
  `size` VARCHAR(45) NULL DEFAULT 'S',
  `ingredient_ingrtypeid` INT NOT NULL,
  PRIMARY KEY (`accomplishment_accid`, `ingredient_ingrid`,
`ingredient_ingrtypeid`),
  INDEX `fk_accomplishment_has_ingredient_ingredient1_idx`
(`ingredient_ingrid` ASC, `ingredient_ingrtypeid` ASC),
  INDEX `fk_accomplishment_has_ingredient_accomplishment1_idx`
(`accomplishment_accid` ASC),
  CONSTRAINT `fk_accomplishment_has_ingredient_accomplishment1`
    FOREIGN KEY (`accomplishment_accid`)
    REFERENCES `mydb`.`accomplishment` (`accid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_accomplishment_has_ingredient_ingredient1`
    FOREIGN KEY (`ingredient_ingrid` , `ingredient_ingrtypeid`)
    REFERENCES `mydb`.`ingredient` (`ingrid` , `ingrtypeid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`accomplishment_has_process`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`accomplishment_has_process` ;

CREATE TABLE IF NOT EXISTS `mydb`.`accomplishment_has_process` (
  `accomplishment_accid` INT NOT NULL,
  `process_processid` INT NOT NULL,
  `time` INT NULL,
  `initialtemp` INT NULL,
  PRIMARY KEY (`accomplishment_accid`, `process_processid`),
  INDEX `fk_accomplishment_has_process_process1_idx` (`process_processid`
ASC),
  INDEX `fk_accomplishment_has_process_accomplishment1_idx`
(`accomplishment_accid` ASC),
  CONSTRAINT `fk_accomplishment_has_process_accomplishment1`
    FOREIGN KEY (`accomplishment_accid`)
    REFERENCES `mydb`.`accomplishment` (`accid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_accomplishment_has_process_process1`
    FOREIGN KEY (`process_processid`)
    REFERENCES `mydb`.`process` (`processid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`recipe_has_ingredient`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`recipe_has_ingredient` ;

CREATE TABLE IF NOT EXISTS `mydb`.`recipe_has_ingredient` (
  `recipe_recipe_id` INT NOT NULL,
  `recipe_dish_dish_id` INT NOT NULL,
  `ingredient_ingrid` INT NOT NULL,
  PRIMARY KEY (`recipe_recipe_id`, `recipe_dish_dish_id`,
`ingredient_ingrid`),
  INDEX `fk_recipe_has_ingredient_ingredient1_idx` (`ingredient_ingrid` ASC),
  INDEX `fk_recipe_has_ingredient_recipe1_idx` (`recipe_recipe_id` ASC,
`recipe_dish_dish_id` ASC),
  CONSTRAINT `fk_recipe_has_ingredient_recipe1`
```

```
    FOREIGN KEY (`recipe_recipe_id` , `recipe_dish_dish_id`)
    REFERENCES `mydb`.`recipe` (`recipe_id` , `dish_dish_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_recipe_has_ingredient_ingredient1`
    FOREIGN KEY (`ingredient_ingrid`)
    REFERENCES `mydb`.`ingredient` (`ingrid`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`userlog`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`userlog` ;

CREATE TABLE IF NOT EXISTS `mydb`.`userlog` (
  `userlogid` INT NOT NULL AUTO_INCREMENT,
  `user_id` INT NULL,
  `user_name` VARCHAR(45) NULL,
  `action` VARCHAR(45) NULL,
  `changedate` DATETIME NULL,
  PRIMARY KEY (`userlogid`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`accountlog`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`accountlog` ;

CREATE TABLE IF NOT EXISTS `mydb`.`accountlog` (
  `accountlogid` INT NOT NULL AUTO_INCREMENT,
  `account_name` VARCHAR(45) NULL,
  `password` VARCHAR(45) NULL,
  `changedate` DATETIME NULL,
  `action` VARCHAR(45) NULL,
  PRIMARY KEY (`accountlogid`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`acclog`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `mydb`.`acclog` ;

CREATE TABLE IF NOT EXISTS `mydb`.`acclog` (
  `acclogid` INT NOT NULL AUTO_INCREMENT,
  `accid` INT NULL,
  `accname` VARCHAR(45) NULL,
  `totgrade` FLOAT(4,2) NULL,
  `level` VARCHAR(45) NULL,
  `changedate` DATETIME NULL,
  `action` VARCHAR(45) NULL,
  PRIMARY KEY (`acclogid`))
ENGINE = InnoDB;

USE `mydb` ;

-- -----------------------------------------------------
-- Placeholder table for view `mydb`.`useracc`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`useracc` (`user_id` INT, `user_name` INT,
`accid` INT, `accname` INT, `date` INT, `flavor` INT, `nutr` INT, `texture`
INT, `level` INT, `comment` INT);

-- -----------------------------------------------------
-- Placeholder table for view `mydb`.`accinfo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`accinfo` (`accid` INT, `user_id` INT,
`accname` INT, `date` INT, `flavor` INT, `nutr` INT, `texture` INT, `level`
INT, `comment` INT, `ingrname` INT, `ingrtypename` INT, `weight` INT, `size`
INT, `processname` INT, `time` INT, `initialtemp` INT);

-- -----------------------------------------------------
-- View `mydb`.`useracc`
-- -----------------------------------------------------
DROP VIEW IF EXISTS `mydb`.`useracc` ;
DROP TABLE IF EXISTS `mydb`.`useracc`;
USE `mydb`;
CREATE  OR REPLACE VIEW useracc AS

SELECT user_id,user_name,accid,accname,date,flavor,nutr,texture,level,comment
FROM user u INNER JOIN accomplishment a ON u.user_id=a.user_user_id
ORDER BY user_id;

-- -----------------------------------------------------
-- View `mydb`.`accinfo`
-- -----------------------------------------------------
```

```
DROP VIEW IF EXISTS `mydb`.`accinfo` ;
DROP TABLE IF EXISTS `mydb`.`accinfo`;
USE `mydb`;
CREATE  OR REPLACE VIEW accinfo AS
    SELECT
        accid,
        user_id,
        accname,
        date,
        flavor,
        nutr,
        texture,
        level,
        comment,
        ingrname,
        ingrtypename,
        weight,
        size,
        processname,
        time,
        initialtemp
    FROM
        user u
            INNER JOIN
        accomplishment a ON u.user_id = a.user_user_id
            INNER JOIN
        accomplishment_has_process ahp ON a.accid = ahp.accomplishment_accid
            INNER JOIN
        process p ON ahp.process_processid = p.processid
            INNER JOIN
        accomplishment_has_ingredient ahi ON a.accid =
ahi.accomplishment_accid
            INNER JOIN
        ingredient i ON i.ingrid = ahi.ingredient_ingrid
    ORDER BY accid;
USE `mydb`;

DELIMITER $$

USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`user_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`user_BEFORE_UPDATE` BEFORE
UPDATE ON `user` FOR EACH ROW
BEGIN
```

```
INSERT INTO userlog
    SET action = 'update',
     user_id = OLD.user_id,
        user_name = OLD.user_name,
        changedate = NOW();
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`account_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`account_BEFORE_UPDATE` BEFORE
UPDATE ON `account` FOR EACH ROW
BEGIN
INSERT INTO accountlog
    SET action = 'update',
     account_name = OLD.account_name,
        password = OLD.password,
        changedate = NOW();
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`accomplishment_AFTER_INSERT` $$
USE `mydb`$$
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`accomplishment_AFTER_INSERT`
AFTER INSERT ON `accomplishment` FOR EACH ROW
BEGIN
INSERT INTO acclog
    SET action = 'insert',
     accid = new.accid,
        accname = new.accname,
        changedate = NOW();
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`accomplishment_AFTER_UPDATE` $$
USE `mydb`$$
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`accomplishment_AFTER_UPDATE`
AFTER UPDATE ON `accomplishment` FOR EACH ROW
BEGIN
    INSERT INTO acclog
    SET action = 'update',
     accid = OLD.accid,
        accname = OLD.accname,
        totgrade=NEW.totgrade,
        level=NEW.level,
        changedate = NOW();
END$$


DELIMITER ;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Privileges

```
Create a Supervisor and let him have the right to all table in mydb
select Host,User from mysql.user;
Drop  User 'SupervisorHHQ'@'localhost';
Create User 'SupervisorHHQ'@'localhost' identified By 'hqhandsome';
grant all on mydb.* to 'SupervisorHHQ'@'localhost';
grant create,show view on mydb to 'SupervisorHHQ'@'localhost';
show grants for 'SupervisorHHQ'@'localhost' ;


#Create a Student and let him have the right to choose game,course and dish
and view his or his
Drop  User  'StudentJM'@'%';
Create User 'StudentJM'@'%'identified By'JMLearning';
Grant update on mydb.account to 'StudentJM'@'%';
Grant Select,insert,update,delete on mydb.user to 'StudentJM'@'%';
Grant Select,insert,delete on mydb.friendlist to 'StudentJM'@'%';

Grant select(course_id, course_name, course_difficulty) on mydb.course to
'StudentJM'@'%';
Grant select(certificate_id, certificate_name) on mydb.certificate to
'StudentJM'@'%';
Grant select(game_id, dish_id) on mydb.game to 'StudentJM'@'%';
Grant Select on mydb.accomplishment to 'StudentJM'@'%';
Grant Select on mydb.enrollment to 'StudentJM'@'%';
Grant Select on mydb.enrollment to 'StudentJM'@'%';
revoke all on mydb.* from 'StudentJm'@'%';
```

```
#Create a Instructer that is in charge of the dish,reminder,recipe
Drop  User  'InstructorTY'@'localhost';
Grant Select,insert,update,delete on mydb.recipe to
'InstructorTY'@'localhost';
Grant Select,insert,update,delete on mydb.reminder to
'InstructorTY'@'localhost';
Grant Select,insert,update,delete on mydb.dish to 'InstructorTY'@'localhost';
```

Certificate

```
#show all certificate
SELECT
    certificate_name, certificate_des
FROM
    certificate;

select course_name from course;

select course_name from course where certificate_certificate_id=1;
select * from dish;
select dish_name,course_id,course_difficulty from dish
 inner join course on course.course_id=dish.course_course_id where
course_difficulty in (1,2);

#search by difficulty
select dish_name,difficulty from recipe
inner join dish on recipe.dish_dish_id=dish.dish_id where difficulty in
(1,2);
select * from recipe;

#get the course name of a dish
select course.course_name from course
inner join dish on course.course_id=dish.course_course_id
where dish_name='spicy beef';

#same function using subquery
select course_name from course
where exists (select course_name from dish
where course.course_id=dish.course_course_id and dish_name='spicy beef');

select course.course_name,dish.dish_name,recipe.difficulty from dish
inner join course on course.course_id=dish.course_course_id
inner join recipe on dish.dish_id=recipe.dish_dish_id
```

```
where dish_name='spicy beef';

#spicy
select dish_name,course_name from dish
inner join course on  course.course_id=dish.course_course_id
 where dish_name like '%spicy%'
 and course_difficulty in (1,2);
 #number of cooking process
 select * from process;
 select process.processname,count(dish.dish_id) headcount
 from dish_has_process
 inner join process on process.processid=dish_has_process.process_processid
 inner join dish on dish.dish_id=dish_has_process.dish_dish_id
 group by processid;
 select * from dish;

 #details about enrollment
SELECT
    user_name, status, certificate_name
FROM
    enrollment
        INNER JOIN
    user ON user.user_id = enrollment.user_user_id
        INNER JOIN
    certificate ON enrollment.certificate_certificate_id =
certificate.certificate_id
ORDER BY user_name;

#how many certificates user enrolled
SELECT
    user_name, COUNT(certificate_name) numbers
FROM
    enrollment
        INNER JOIN
    user ON user.user_id = enrollment.user_user_id
        INNER JOIN
    certificate ON enrollment.certificate_certificate_id =
certificate.certificate_id
GROUP BY user_name;

#show reminder during cooking process
select * from reminder;
SELECT
    sequence, reminder_start, reminder_des
FROM
```

```
    reminder
    inner join recipe on recipe.recipe_id=reminder.recipe_recipe_id
    where recipe_name='kung pao'
ORDER BY sequence;
```

Cooking Process

Stores procedure 1: add accomplishment

```
DROP PROCEDURE IF EXISTS addacc;
DELIMITER //
CREATE PROCEDURE addacc(
IN userid INT,
IN acname VARCHAR(100),
IN acprocess VARCHAR(20),
IN prtime INT,
IN initemp INT,
OUT aid INT)

BEGIN

INSERT IGNORE INTO accomplishment(user_user_id,accname,date)
values(userid,acname,current_date());

SELECT
    accid
INTO aid FROM
    accomplishment a
WHERE
    a.user_user_id = userid
        AND a.accname = acname;

INSERT INTO
accomplishment_has_process(accomplishment_accid,process_processid,time,initia
ltemp)
SELECT
    aid, processid, prtime, initemp
FROM
    process p
WHERE
    p.processname = acprocess;
SELECT aid;
END//
DELIMITER ;
```

Stored procedure 2: add ingredients

```
DROP PROCEDURE IF EXISTS addingr;
DELIMITER //
CREATE PROCEDURE addingr(
IN accid INT,
IN ingrname VARCHAR(100),
IN ingrwgt FLOAT,
IN ingrsize VARCHAR(10))

BEGIN


INSERT INTO accomplishment_has_ingredient
(accomplishment_accid,ingredient_ingrid,ingredient_ingrtypeid,weight,size)
SELECT
    accid, ingrid, ingrtypeid, ingrwgt, ingrsize
FROM
    ingredient i
WHERE
    i.ingrname = ingrname;

END//
DELIMITER ;
```

Stored procedure 3: gradeflavor

```
DROP PROCEDURE IF EXISTS gradeflavor;
DELIMITER //
CREATE PROCEDURE gradeflavor(
IN accompid INT,
OUT flagrade FLOAT,
OUT flalevel VARCHAR(10),
OUT comment VARCHAR(200))
BEGIN

DECLARE accsalty FLOAT;
DECLARE accsweet FLOAT;
DECLARE accsour  FLOAT;
DECLARE accspicy FLOAT;
DECLARE accthick FLOAT;
DECLARE saltyscore FLOAT;
DECLARE sweetscore FLOAT;
DECLARE sourscore FLOAT;
DECLARE spicyscore FLOAT;
DECLARE thickscore FLOAT;
DECLARE saltycom VARCHAR(100);
DECLARE sweetcom VARCHAR(100);
DECLARE sourcom VARCHAR(100);
DECLARE spicycom VARCHAR(100);
DECLARE thickcom VARCHAR(100);

SELECT
    SUM(salty * weight) / SUM(weight)
INTO accsalty FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;

SELECT
    SUM(sweet * weight) / SUM(weight)
INTO accsweet FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;

SELECT
    SUM(sour * weight) / SUM(weight)
INTO accsour FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(spicy * weight) / SUM(weight)
INTO accspicy FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
```

```
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(thick * weight) / SUM(weight)
INTO accthick FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;

if (accsalty>=0 AND accsalty<= 100) then
      set saltyscore = 0; set saltycom='no salt,';
elseif (accsalty>100 AND accsalty<=300) then
      set saltyscore =0.05*accsalty-5; set saltycom='not enought salt,';
elseif (accsalty>300 AND accsalty<=700) then
      set saltyscore=10; set saltycom='perfectly salty!';
elseif (accsalty>700 AND accsalty<=1000) then
      set saltyscore=(-1/30)*(accsalty-1000); set saltycom='a little
oversalty,';
else
      set saltyscore=0; set saltycom='tooooooo salty!';

end if;

if (accsweet>=0 AND accsweet<=3) then
      set sweetscore=(4/3)*(accsweet+9/2); set sweetcom='slightly sweet,';
elseif (accsweet>3 AND accsweet<=10) then
      set sweetscore=10; set sweetcom='perfectly sweet!';
elseif (accsweet>10 AND accsweet<=20) then
      set sweetscore=(-1)*accsweet+20; set sweetcom='slightly oversweet,';
else
      set sweetscore=0; set sweetcom='oversweet!';
end if;

if (accsour>=0 AND accsour<=30) then
      set sourscore=0.1*(accsour+70); set sourcom='slightly sour,';
elseif (accsour>30 AND accsour<=60) then
      set sourscore=10; set sourcom='perfectly sour,';
elseif (accsour>60 AND accsour<=90) then
      set sourscore=(-1/3)*(accsour-90); set sourcom='slightly oversour,';
else
      set sourscore=0; set sourcom='oversour!';
end if;
```

```
if (accspicy>=0 AND accspicy<=30) then
      set spicyscore=0.1*(accspicy+70); set spicycom='slightly spicy,';
elseif (accspicy>30 AND accspicy<=60) then
      set spicyscore=10; set spicycom='perfectly spicy,';
elseif (accspicy>60 AND accspicy<=90) then
      set spicyscore=(-1/3)*(accspicy-90); set spicycom='slightly oversour,';
else
      set spicyscore=0; set spicycom='oversour!';
end if;

if (accthick>=0 AND accthick<=5) then
      set thickscore=accthick+5; set thickcom='freshness.';
elseif (accthick>5 AND accthick<=15) then
      set thickscore=10; set thickcom='perfect thickness.';
elseif (accthick>15 AND accthick<=25) then
      set thickscore=(-1)*accthick+25; set thickcom='slightly oily.';
else
      set thickscore=0; set thickcom='too oily!';
end if;

set
flagrade=0.6*saltyscore+0.2*sweetscore+0.1*thickscore+0.05*spicyscore+0.05*so
urscore;
set comment=concat(saltycom,' ', sweetcom,' ' ,sourcom,' ',spicycom, '
',thickcom);

if (flagrade>=0 AND flagrade<=2) then
      set flalevel='E';
elseif (flagrade>2 AND flagrade<=4) then
      set flalevel='D';
elseif (flagrade>4 AND flagrade<6) then
      set flalevel='C';
elseif (flagrade>6 AND flagrade<=8) then
      set flalevel='B';
else
      set flalevel='A';
END IF;

END //

DELIMITER ;
```

Stored procedure 4: grade nutrition

```
DROP PROCEDURE IF EXISTS gradenutr;
DELIMITER //
CREATE PROCEDURE gradenutr(
IN accompid INT,
OUT nutrgrade FLOAT,
OUT nutrlevel VARCHAR(10),
OUT nutrcomment VARCHAR(100))

BEGIN

DECLARE accfat FLOAT;
DECLARE accsat_fat FLOAT;
DECLARE accchole FLOAT;
DECLARE accsodium FLOAT;
DECLARE acccarbhy FLOAT;
DECLARE accfiber FLOAT;
DECLARE accsugar FLOAT;
DECLARE accprotein FLOAT;
DECLARE accvma FLOAT;
DECLARE accvmc FLOAT;
DECLARE acccal FLOAT;
DECLARE acciron FLOAT;


SELECT
    SUM(weight * Fat) / SUM(weight)
INTO accfat FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;

SELECT
    SUM(weight * fat_sat) / SUM(weight)
INTO accsat_fat FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
```

```
SELECT
    SUM(weight * chol) / SUM(weight)
INTO accchole FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(weight * sodium) / SUM(weight)
INTO accsodium FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(weight * cbhy) / SUM(weight)
INTO acccarbhy FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(weight * fiber) / SUM(weight)
INTO accfiber FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(weight * sugar) / SUM(weight)
INTO accsugar FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
```

```
SELECT
    SUM(weight * protein) / SUM(weight)
INTO accprotein FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(weight * va) / 5000
INTO accvma FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(weight * vc) / 60
INTO accvmc FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(weight * calcium) / 1000
INTO acccal FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
SELECT
    SUM(weight * iron) / 18
INTO acciron FROM
    accomplishment_has_ingredient ahi
        INNER JOIN
    ingredient i ON ahi.ingredient_ingrid = i.ingrid
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;
```

```
set nutrgrade=5.710-0.0538*accfat-0.423*accsat_fat-0.00398*accchole-
0.00254*accsodium
-0.0300*acccarbhy+0.561*accfiber-
0.0245*accsugar+0.123*accprotein+0.00562*accvma
+0.0137*accvmc+0.0685*acccal-0.0186*acciron;

if (nutrgrade>=0 AND nutrgrade<=2) then
      set nutrlevel='E';set nutrcomment='unhealthy.';
elseif (nutrgrade>2 AND nutrgrade<=4) then
      set nutrlevel='D';set nutrcomment='not very healthy.';
elseif (nutrgrade>4 AND nutrgrade<=6) then
      set nutrlevel='C'; set nutrcomment='fairly healthy.';
elseif (nutrgrade>6 AND nutrgrade<=8) then
      set nutrlevel='B';set nutrcomment='healthy.';
else
      set nutrlevel='A';set nutrcomment='very healthy!';
END if;

END //

DELIMITER ;
```

Stored procedure 5: grade texture

```
DROP PROCEDURE IF EXISTS gradetexture;
DELIMITER //
CREATE PROCEDURE gradetexture(
IN accompid INT,
OUT texgrade FLOAT,
OUT texlevel VARCHAR(10),
OUT texcomment VARCHAR(100))

BEGIN

DECLARE accproc VARCHAR(20);
DECLARE acctime INT;
DECLARE initemp INT;
DECLARE p INT;
DECLARE t INT;
DECLARE s INT;
DECLARE iniscore FLOAT;

SELECT
```

```
    AVG(CASE processname
        WHEN 'boil' THEN 1.5
        WHEN 'steam' THEN 1
        WHEN 'bake' THEN 2
        WHEN 'fry' THEN 3
        WHEN 'grill' THEN 3
        ELSE 0
    END)
INTO p FROM
    accomplishment a
        INNER JOIN
    accomplishment_has_process ahp ON a.accid = ahp.accomplishment_accid
        INNER JOIN
    process p ON ahp.process_processid = p.processid
WHERE
    a.accid = accompid
GROUP BY accid;


SELECT
    SUM(time)
INTO acctime FROM
    accomplishment_has_process
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;

SELECT
    AVG(initialtemp)
INTO initemp FROM
    accomplishment_has_process
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;

SELECT
    SUM(weight * (CASE size
        WHEN 'S' THEN 3
        WHEN 'M' THEN 2
        WHEN 'L' THEN 1
    END)) / SUM(weight)
INTO s FROM
    accomplishment_has_ingredient
WHERE
    accomplishment_accid = accompid
```

```
GROUP BY accomplishment_accid;

SELECT
    SUM(weight * (CASE
        WHEN ingredient_ingrtypeid = 10 THEN 1
        WHEN ingredient_ingrtypeid = 13 THEN 1
        WHEN ingredient_ingrtypeid = 17 THEN 1
        WHEN ingredient_ingrtypeid = 5 THEN 2
        WHEN ingredient_ingrtypeid = 15 THEN 2
        ELSE 3
    END)) / SUM(weight)
INTO t FROM
    accomplishment_has_ingredient
WHERE
    accomplishment_accid = accompid
GROUP BY accomplishment_accid;

SET iniscore=(initemp/100)*p*t*s*0.3*acctime;

IF
       iniscore>=0 AND iniscore<=10 THEN
      SET texgrade=0; SET texcomment='raw';
      ELSEIF iniscore>10 AND iniscore<=25 THEN
      SET texgrade=(8/15)*(iniscore-10); SET texcomment='not raw but
undercooked';
      ELSEIF iniscore>25 AND iniscore<=30 THEN
      SET texgrade=(2/5)*(iniscore-5); SET texcomment='wellcooked!';
      ELSEIF iniscore>30 AND iniscore<=35 THEN
      SET texgrade=(-2/5)*iniscore+22; SET texcomment='Slightly overcooked';
      ELSEIF iniscore>35 AND iniscore<50 THEN
      SET texgrade=(-8/15)*iniscore+(80/3); SET texcomment='charcoal';
      ELSE
      SET texgrade=0; SET texcomment='Please practice more!';
END IF;

IF
       texgrade>=0 AND texgrade<=2 THEN
      SET texlevel='E';
      ELSEIF texgrade>2 AND texgrade<=4 THEN
      SET texlevel='D';
      ELSEIF texgrade>4 AND texgrade<=6 THEN
      SET texlevel='C';
      ELSEIF texgrade>6 AND texgrade<=8 THEN
      SET texlevel='B';
      ELSEIF texgrade>8 AND texgrade<=10 THEN
      SET texlevel='A';
END IF;

END //
DELIMITER ;
```

Stored procedure 6: total grading

```
DROP PROCEDURE IF EXISTS totgrade;
DELIMITER //
CREATE PROCEDURE totgrade(
IN accompid INT)

BEGIN

DECLARE ovacomment VARCHAR(100);
DECLARE grade FLOAT;
DECLARE totlevel VARCHAR(10);
DECLARE totcomment VARCHAR(1000);

CALL gradeflavor(accompid,@flagrade,@flalevel,@flacomment);
CALL gradenutr(accompid,@nutrgrade,@nutrlevel,@nutrcomment);
CALL gradetexture(accompid,@texgrade,@texlevel,@texcomment);

SET grade= 0.5*@flagrade+0.15*@nutrgrade+0.35*@texgrade;

IF grade>=0 AND grade<=2 THEN
      SET totlevel='E'; SET ovacomment='Bad accomplishment.';
ELSEIF grade>2 AND grade<=4 THEN
      SET totlevel='D'; SET ovacomment='Not very good.';
ELSEIF grade>4 AND grade<=6 THEN
      SET totlevel='C'; SET ovacomment='fairly good.';
ELSEIF grade>6 AND grade<=8 THEN
      SET totlevel='D'; SET ovacomment='Pretty good.';
ELSE
      SET totlevel='A'; SET ovacomment='Excellent!';
END IF;

SET totcomment=concat(ovacomment,' ',@flacomment,' ',@nutrcomment,'
',@texcomment);

UPDATE accomplishment a
SET totgrade=grade, level=totlevel, comment=totcomment, flavor=@flagrade,
nutr=@nutrgrade, texture=@texgrade
WHERE a.accid=accompid;
```

```
SELECT * FROM accomplishment a WHERE a.accid=accompid;

END //
DELIMITER ;
```

View of accomplishment information:

```
CREATE VIEW accinfo AS
    SELECT
        accid,
        user_id,
        accname,
        date,
        flavor,
        nutr,
        texture,
        level,
        comment,
        ingrname,
        ingrtypename,
        weight,
        size,
        processname,
        time,
        initialtemp
    FROM
        user u
            INNER JOIN
        accomplishment a ON u.user_id = a.user_user_id
            INNER JOIN
        accomplishment_has_process ahp ON a.accid = ahp.accomplishment_accid
            INNER JOIN
        process p ON ahp.process_processid = p.processid
            INNER JOIN
        accomplishment_has_ingredient ahi ON a.accid =
ahi.accomplishment_accid
            INNER JOIN
        ingredient i ON i.ingrid = ahi.ingredient_ingrid
    ORDER BY accid;
```

Trigger: documenting information of insert of accomplishment

```
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`accomplishment_AFTER_INSERT`
AFTER INSERT ON `accomplishment` FOR EACH ROW
BEGIN
INSERT INTO acclog
    SET action = 'insert',
     accid = new.accid,
        accname = new.accname,
        changedate = NOW();
END
```

Trigger: documenting information of update of accomplishment

```
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`accomplishment_AFTER_UPDATE`
AFTER UPDATE ON `accomplishment` FOR EACH ROW
BEGIN
    INSERT INTO acclog
    SET action = 'update',
     accid = OLD.accid,
        accname = OLD.accname,
        totgrade=NEW.totgrade,
        level=NEW.level,
        changedate = NOW();
END
```

Game mode

Store Procedure 1:

```
DROP PROCEDURE IF EXISTS showallgame;
DELIMITER //
CREATE PROCEDURE showallgame(IN userid INT)
begin
SELECT game_game_id,`game-totalGrade`
FROM gamezone
WHERE user_user_id=userid;
end //
DELIMITER ;
call showallgame(1);
```

Store Procedure 2:

```
DROP PROCEDURE IF EXISTS showgamedetail;
```

```
DELIMITER //
CREATE PROCEDURE showgamedetail(IN gameid INT)
begin
SELECT *
FROM game
WHERE game_id=gameid and statuss=true;
end //
DELIMITER ;
call showgamedetail(01);
```

Store Procedure 3:

```
use mydb;
DROP PROCEDURE IF EXISTS gradesituatio;
DELIMITER //
CREATE PROCEDURE gradesituation(IN gameid INT)
begin

SELECT game.game_data_game_progress,accomplishment.totgrade
FROM game
inner join
accomplishment on game.accomplishment_accid=accomplishment.accid
where game.game_id=gameid;

end //
DELIMITER ;
call gradesituatio(3);
```

Store procedure 4:

```
use mydb;
DROP PROCEDURE IF EXISTS showgameprocess;
DELIMITER //
CREATE PROCEDURE showgameprocess(IN processid INT)
begin
SELECT game_background,dish_id
FROM game_data
WHERE game_progress=processid;
end //
DELIMITER ;
call showgameprocess(1);
```

Store Procedure 5:

```
use mydb;
DROP PROCEDURE IF EXISTS totalgrade;
DELIMITER //
CREATE PROCEDURE totalgrade(IN gameid INT)
begin
select sum(totgrade) AS totalgradee
from(
SELECT game.game_data_game_progress,accomplishment.totgrade
FROM game
inner join
accomplishment on game.accomplishment_accid=accomplishment.accid
where game.game_id=gameid
) as result;

end //
DELIMITER ;
call totalgrade(3);
```

View 1:

```
DROP view IF EXISTS ranking;
CREATE VIEW ranking AS
    SELECT
        game_game_id,`game-totalGrade`
    FROM
        gamezone

        order by
        `game-totalGrade` DESC
        limit 0,3;

        select * from ranking;
```

Trigger 1:

```
DELIMITER //
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`game_AFTER_UPDATE` AFTER UPDATE
ON `game` FOR EACH ROW
BEGIN

update gamezone
SET game_totalGrade=(select sum(totgrade)
from(
SELECT game.game_data_game_progress,accomplishment.totgrade
FROM game
inner join
accomplishment on game.accomplishment_accid=accomplishment.accid
where game.game_id=NEW.game_id
) as result)
where gamezone.game_game_id=NEW.game_id;

END//
DELIMITER ;
```

Social Network

Friend searching:

```
#find all friends for Haoqi
drop procedure if exists friend_search;
Delimiter //
create procedure friend_search(in p_name varchar(45))
begin
Select user_name
from user
where user_id=all(
select friendid from friend_list
where user_id= (select user_id from
user where user_name=p_name));

end //
delimiter ;
call friend_search('Haoqi');
```

Show all the dish for specific user

```
#find all dishes for one users
drop procedure if exists show_dishes;
Delimiter //
create procedure show_dishes(in p_name varchar(45))
begin
Select user_name,dish_name
from user u
inner join
accomplishment a on u.user_id=a.user_user_id
left join
```

```
dish d on a.dish_dish_id=d.dish_id
where user_name=p_name;

end //
delimiter ;
call show_dishes('Haoqi');
```

Trigger for user's change

```
#delete enrollment if the certification is cancelled
drop trigger if exists enrollment_delete;
DELIMITER //
create trigger enrollment_delete
after delete on certification
for each row
begin
 delete from enrollment where certification_certification_id =
 old.certification_id;
end;
//
Delimiter;
#Sync game update
drop trigger if exists game_update;
DELIMITER //
create
 trigger game_update
 before update
 on game
 for each row
 begin
 update gamezone set game_game_id=new.game_id
end//
delimiter ;
```

Search for user by name

```
SELECT
    *
FROM
    user
WHERE
    user_name LIKE '%ki%';
# search for user by id
SELECT
```

```
    *
FROM
    user
WHERE
    user_id=3;
```

View Users whose zone has like over 1000

```
SELECT
      user.user_id,user.user_name,socialplatform.description,socialplatform.l
ike
FROM
    socialplatform
        INNER JOIN
        user ON  user.user_id=socialplatform.user_user_id
WHERE
    socialplatform.like > 1000
order by socialplatform.like desc;
```

View user's certificated passed

```
SELECT
    certificate_name, enrollment.status
FROM
    enrollment
        INNER JOIN
    user ON user.user_id = enrollment.user_user_id
        INNER JOIN
    certificate ON certificate.certificate_id =
enrollment.certificate_certificate_id
WHERE
    enrollment.status = 'pass'
        AND user.user_id = 3;
```