CustomCooking

UN'APPLICAZIONE SOCIAL DI RICETTE CHE TI AIUTA A SCEGLIERE COSA PREPARARE IN BASE A QUELLO CHE HAI IN CUCINA

Gozzoli Alessio
0000666684
alessio.gozzoli@studio.unibo.it

Abrignani Federico
0000671199
federico.abrignani@studio.unibo.it

Ognibene Tommaso
0000672192
tommaso.ognibene@studio.unibo.it

Table of Contents

1 Analisi dei requisiti 2
1.1 Requisiti espressi in linguaggio naturale 2
1.2 Glossario dei termini 2
1.3 Strutturazione dei requisiti 2
1.4 Specifica operazioni 3
2 Progettazione concettuale 4
2.1 Identificazione delle entità e relazioni 4
2.2 Un primo scheletro dello schema 4
2.3 Sviluppo delle componenti dello scheletro 4
2.4 Unione delle componenti nello schema finale ridotto 5
2.5 Dizionario dei dati 5
2.6 Regole aziendali 6
3 Progettazione logica 6
3.1 Considerazioni sui volumi delle operazioni 6
3.2 Normalizzazione 7
4 Codifica SQL 7
4.1 Definizione dello schema 7
4.2 Codifica delle operazioni 10
5 Testing 12

1 Analisi dei requisiti

1.1 Requisiti espressi in linguaggio naturale

Si vuole realizzare una base di dati per un'app di ricette.

L'app deve permettere di scegliere una ricetta sulla base delle disponibilità dell'utente, in termini di prodotti alimentari e utensili da cucina.

La base di dati si compone delle entità *Recipe*, *Product*, *Tool*, *Account*, *Step*, e delle relazioni *Ingredient*, *Author*, *Rating*, *ToolAvailability*, *ProductAvailability*, *ToolSet*, *Sequence*.

Per quanto concerne *Recipe*, ci interessa rappresentare un identificatore univoco (idRecipe), nome, tipo, descrizione, cucina a cui appartiene, eventuale origine regionale, tempo stimato di preparazione, difficoltà secondo l'autore, difficoltà secondo le valutazioni degli utenti utilizzatori, eventuale nome e riferimento ad autore esterno della ricetta (ad esempio l'autore del libro di ricette dalla quale la ricetta è stata estrapolata).

Per quanto concerne *Product*, ci interessa rappresentare nome, classe del prodotto, eventuale URL di un'immagine didascalica.

Per quanto concerne *Tool*, ci interessa rappresentare nome ed eventuale URL di un'immagine didascalica.

Per quanto concerne *User*, ci interessa rappresentare username, email e password.

Per quanto concerne *Step*, ci interessa rappresentare un identificatore numerico, descrizione, tempo stimato, eventuale URL di un'immagine didascalica, eventuale URL di un video esplicativo.

Gli ingredienti per una ricetta possono essere opzionali o necessari, lo stesso vale per gli utensili.

1.2 Glossario dei termini

TERMINE	DESCRIZIONE	SINONIMI	COLLEGAMENTI
Account	Utente registrato		Recipe, ProductAvalaibilty, ToolAvailability, Rating, Author
Recipe	Algoritmo grastronomico		Account, Step, Tool, Product
Product	Prodotto alimentare		Ingredients, User,Recipe
Tool	Strumento per cucinare		Account, Recipe
Step	Passo atomico di una ricetta		Recipe
Ingredient	Prodotto alimentare relativo ad una ricetta		Product, Recipe
Sequence	Sequenza di step per preparare una ricetta		Recipe
ProductAvalaibilty	Disponibilità di un prodotto da parte di un utente		Account, Product
ToolAvailability	Disponibilità di uno strumento da parte di un utente		Account, Tool
Rating	Valutazione di una ricetta da parte di un utente		Account, Recipe
Author	Autore di una ricetta		Account, Recipe
ToolSet	Strumento relativo ad una ricetta		Tool, Recipe

1.3 Strutturazione dei requisiti

- Frasi di carattere generale:

Si vuole costruire una base di dati per un'applicazione mobile che consente di scegliere una ricetta sulla base delle proprie disponibilità in termini di prodotti e utensili, filtrando in termini di tempo, difficoltà e valutazione.

- Frasi relative a *Recipe*:
  Per le ricette, si intende rappresentare un identificatore numerico, nome, tipo, descrizione, cucina di appartenenza, origine geografica, tempo di preparazione, difficoltà secondo l'autore, difficoltà media secondo gli utenti, valutazione media secondo gli utenti, eventuale nome di un autore esterno, eventuale link alla fonte esterna.
- Frasi relative a *Product*:
  Per i prodotti alimentari, si intende rappresentare nome, classe, ed eventuale immagine didascalica.
- Frasi relative a *Tool*:
  Per gli utensili, si intende rappresentare nome ed eventuale immagine didascalica.
- Frasi relative a *Account*:
  Per gli utenti, si intende rappresentare username, email e password.
- Frasi relative a *Step*:
  Per il passo esecutivo di una ricetta, si intende rappresentare un identificatore numerico, descrizione, tempo stimato, eventuale immagine didascalica ed eventuale video esplicativo.
- Frasi relative ai componenti di una ricetta:
  Occorre rappresentare:
  - Gli ingredienti (necessari o opzionali),
  - Gli utensili (necessari o opzionali),
  - La sequenza dei passi esecutivi.
- Frasi relative alla provenienza e valutazione di una ricetta:
  Occorre rappresentare:
  - L'autore della ricetta,
  - La valutazione (voto, commento, difficoltà riscontrata) espressa dagli utenti.
- Frasi relative alle disponibilità di un utente:
  Occorre rappresentare:
  - Gli ingredienti disponibili,
  - Gli utensili disponibili.

1.4 Specifica operazioni

- Inserire un nuovo *Account*.
- Inserire una nuova *Recipe*.
- Inserire un nuovo *Step*.
- Inserire un nuovo *Product*.
- Inserire un nuovo *Tool*.
- Aggiungere uno *Step* ad una *Recipe*.
- Aggiungere un *Product* ad una *Recipe*.
- Aggiungere un *Tool* ad una *Recipe*.
- Aggiungere un *Product* ad un *Account*.
- Aggiungere un *Tool* ad un *Account*.
- Valutare una *Recipe*.
- Visualizzare le ricette filtrate e ordinate sulla base di parametri specifici.

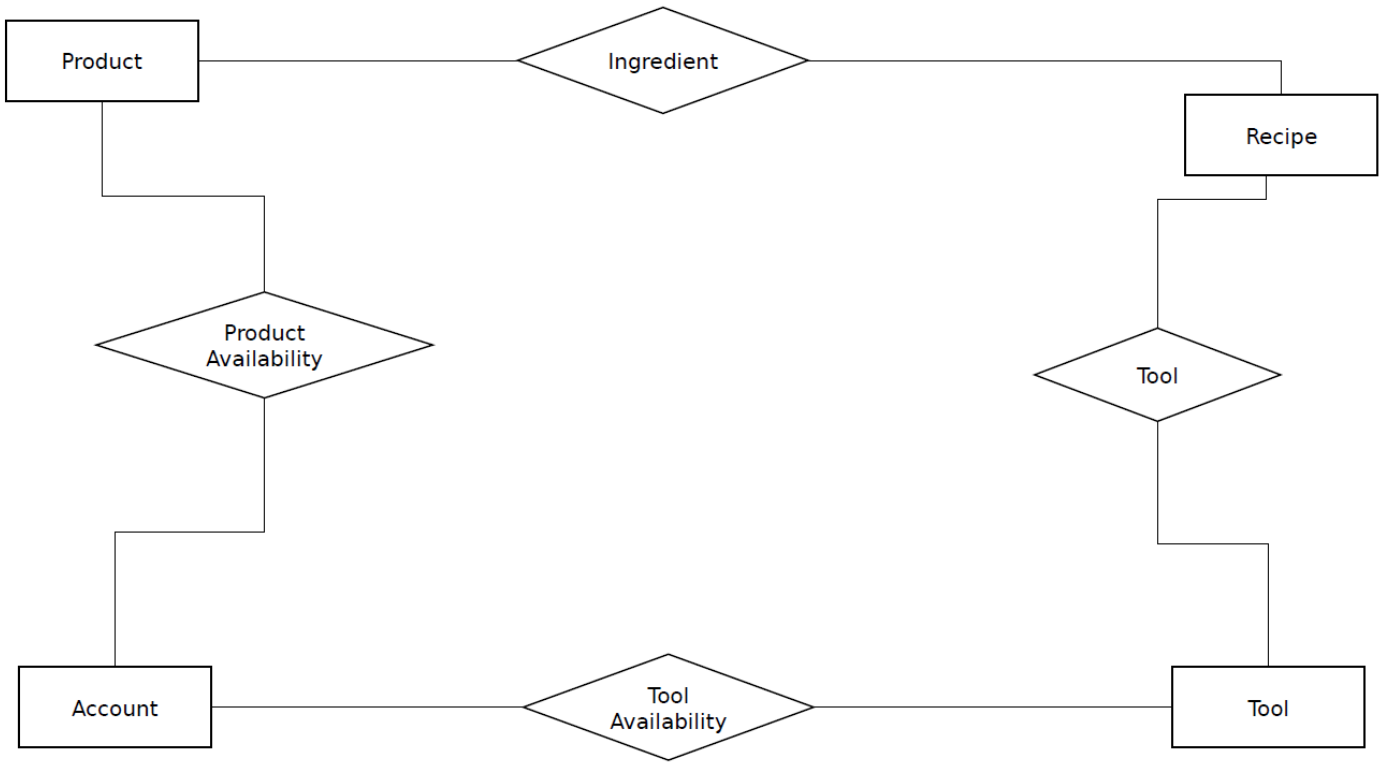
2 Progettazione concettuale

2.1 Identificazione delle entità e relazioni

Applicando la strategia *bottom*-up sono state identificate le seguenti entità e relazioni principali: *Account*, *Recipe*, *Product*, *Ingredient*, *Tool*, *Author*, *Rating*, *ProductAvailability*, *ToolAvailability* , *ToolSet*.

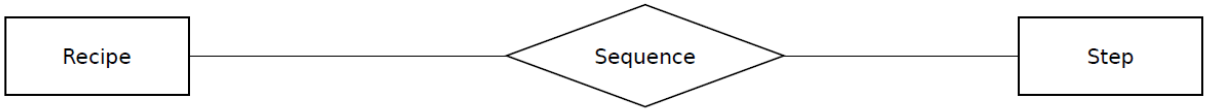
2.2 Un primo scheletro dello schema

A partire da questo elenco è stato modellato il seguento primo scheletro di schema contettuale:



2.3 Sviluppo delle componenti dello scheletro

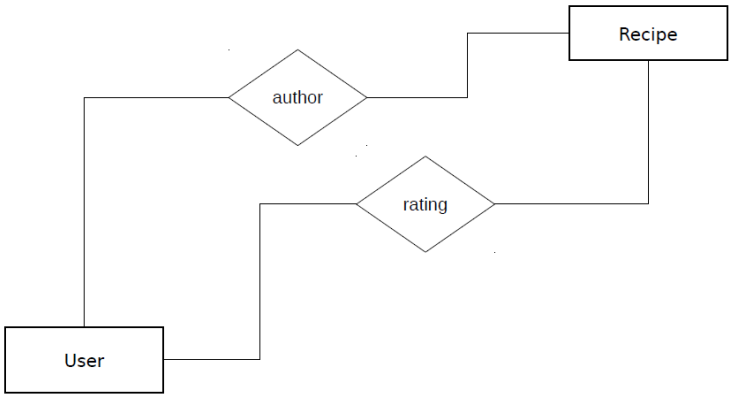
Ogni ricetta è costituita da uno o più step. È stato stabilito di rappresentare la ricetta mediante due differenti entità:



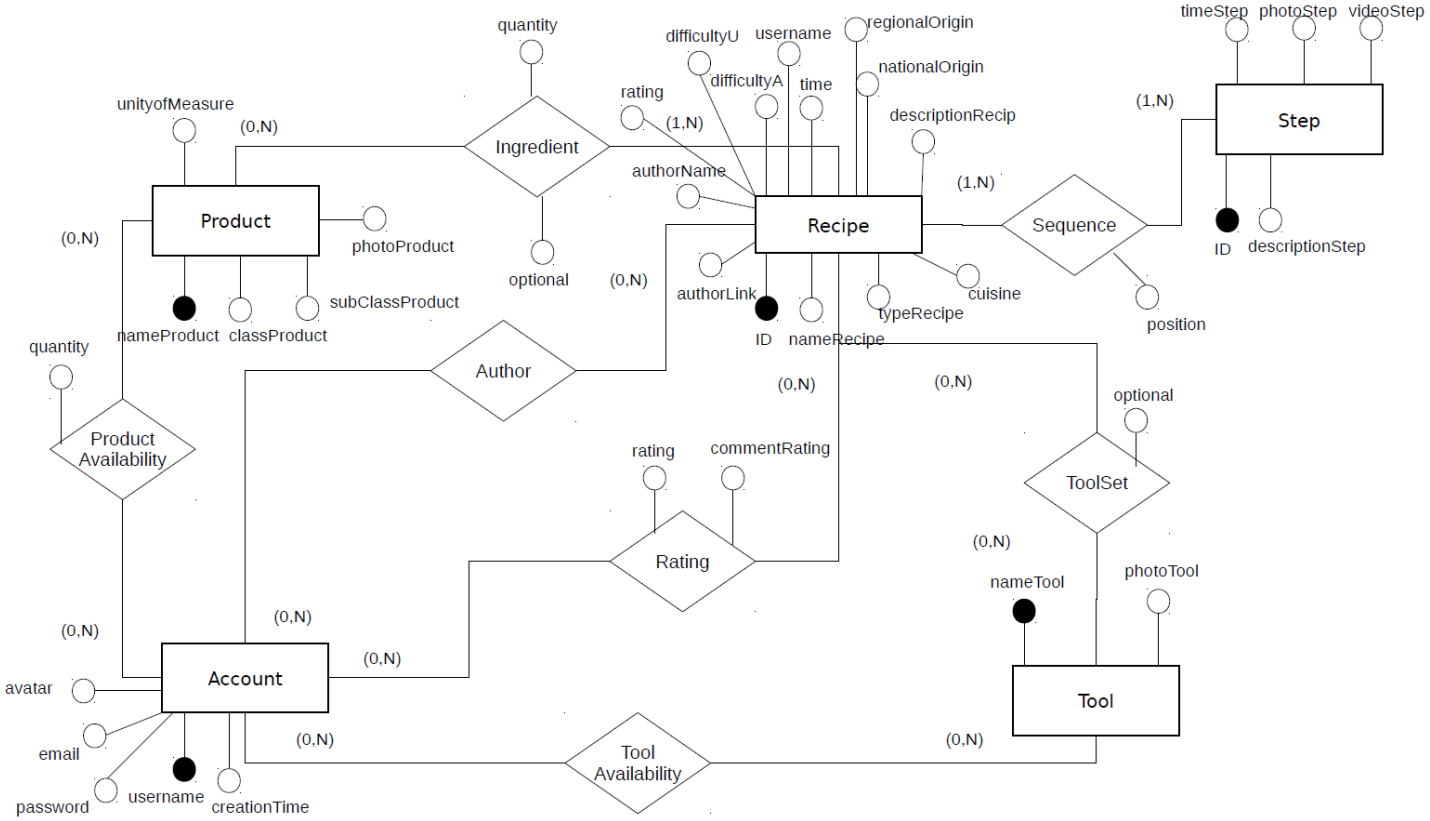
La nuova relazione (*Sequence*) è finalizzata a suddividere le ricette in fasi esecutive atomiche, affinché queste possano essere condivise, evitando ripetizioni, tra molteplici ricette. La nuova entità (*Step*) rappresenta la singola fase esecutiva.

Ogni ricetta è collegata ad un utente che l'ha inserita e creata. Si prevede inoltre la possibilità di inserire ricette create da autori noti.

Inoltre, ogni utente che sceglie una ricetta può esprimere un voto, scrivere un eventuale commento ed un eventuale giudizio sulla difficoltà riscontrata.



2.4 Unione delle componenti nello schema finale ridotto



2.5 Dizionario dei dati

Entità:

NOME ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
Recipe	Ricetta	idRecipe, nameRecipe, typeRecipe, descriptionRecipe, cuisine, regionalOrigin, preparationTime, difficultyAuthor, difficultyUsers, authorName, authorLink, username, rating	idRecipe
Product	Prodotto alimentare	nameProduct, classProduct, unityOfMeasure, subclassProduct, photoProduct	nameProduct
Account	Utente registrato	username, pass, email, creationTime	username
Tool	Strumento per cucinare	nameTool, photoTool	nameTool
Step	Passo esecutivo di una ricetta	idStep, descriptionStep, timeStep, photoStep, videoStep	idStep

Relazioni:

NOME RELAZIONE	DESCRIZIONE	ENTITÀ COINVOLTE	ATTRIBUTI
Rating	Valutazione di una ricetta	Account (0,N) - Recipe(0,N)	username, idRecipe, commentRating, rating, difficulty
Ingredient	Ingrediente di una ricetta	Product(1,N) - Recipe(1,N)	idRecipe, nameProduct, quantity, optional
Sequence	Passo esecutivo di una ricetta	Recipe(1,N) - Step(1,N)	
ToolSet	Strumento usato in una ricetta	Tool(0,N) - Recipe(0,N)	idRecipe, nameTool, optional
ToolAvailability	Strumento nelle disponibilità di un utente	Account (0,N) - Tool(0,N)	username, nameTool
ProductAvailability	Prodotto nelle disponibilità di un utente	Account (0,N) - Product(0,N)	username, nameProduct, quantity
Author	Autore di una ricetta	User(0,N) - Recipe(0,N)	idRecipe, username

2.6 Regole aziendali

Regole di vincolo

(RV1) L'attributo *classProduct* può assumere i seguenti valori: *Meat*, *Vegetable*, *Mushrooms*, *Fruit*, *Fish*, *Spices*, *Cereals*, *Dairy product*, *Alcohol*, *Flour*.

(RV2) L'attributo *typeRecipe* può assumere i seguenti valori: *Starter*, *First course*, *Second course*, *Dessert*, *All in one*.

(RV3) L'attributo *cuisine* può assumere come valore un insieme predefinito ma estendibile di tipi di cucine.

(RV4) L'attributo *regionalOrigin* può assumere come valore un insieme predefinito ma estendibile di regioni geografiche.

(RV5) Il *rating* di una ricetta deve essere compreso tra 1 e 4.

(RV6) La *difficulty* di una ricetta deve essere compresa tra 1 e 4.

Regole di derivazione

(RD1) L'attributo *ratingUsers* di una ricetta deve essere calcolato come media dei giudizi espressi dagli utenti per quella ricetta.

(RD2) L'attributo *preparationTime* di una ricetta deve essere calcolato come somma dei tempi stimati per ogni passo della ricetta.

3 Progettazione logica

3.1 Considerazioni sui volumi delle operazioni

Al fine di incrementare la scalabilità e la portabilità della base di dati, è stato fatto impiego delle *stored procedures*, in particolare in relazione alle operazioni che si stima saranno maggiormente eseguite a regime.

I benefici delle *stored procedures* in MySQL 5 sono i seguenti:

- Incapsulare funzionalità in procedure eseguibili da differenti applicazioni scritte in differenti linguaggi.
- Isolare gli utenti dalle tabelle, autorizzando l'accesso alle *stored procedures* ma non direttamente alle tabelle.
- Migliorare la *performance*.

Gli attributi *difficultyUsers*, *preparationTime*, *rating* dell'entità *Recipe* vengono generati automaticamente mediante dei *triggers*:

- *difficultyUsers* rappresenta la media delle difficoltà attribuite dagli utenti alla ricetta;
- *preparationTime* rappresenta la sommatoria dei tempi dei passi della ricetta;
- *rating* rappresenta la media dei voti attribuiti dagli utenti alla ricetta.

Tali ridondanze sono state introdotte al fine di migliorare l'efficienza delle query. Questa scelta si fonda sulla considerazione che la query che restituisce le ricette in base ai parametri di difficoltà, tempo di preparazione e valutazione, sarà una delle query maggiormente utilizzate. Di conseguenza è ragionevole calcolare una sola volta questi valori, inserendoli in attributi appositi, piuttosto che calcolarli *on-the-fly* ad ogni query.

3.2 Normalizzazione

Escludendo le dipendenze funzionali banali, o quelle contenenti chiavi le quali rispetterebbero comunque i vincoli di Boyce-Codd, si osserva che:

NOME ENTITÀ	COMMENTO
Account	Non contiene dipendenze funzionali.
Product	Non contiene dipendenze funzionali.
Tool	Non contiene dipendenze funzionali.

Step	Non contiene dipendenze funzionali. Occorre sottolineare che la chiave ideale sarebbe stata *descriptionStep*, tuttavia la sua dimensione supera 767 bytes, che è la dimensione massima per una chiave in InnoDB. Una soluzione percorribile poteva essere quella di impiegare come chiave l'*hash* della descrizione, ma, per evitare casi di collisioni tra *hash*, si è optato per una chiave numerica incrementale. Di conseguenza, quando l'utente tenterà di inserire un nuovo step, l'applicazione effettuerà una query per garantire l'unicità di *descriptionStep*:
	
	`create function isStepUnique(description varchar(300)) returns boolean`
	`begin`
	`  return (select count(*) from Step where descriptionStep = description) = 0;`
	`end $`
	
	`drop procedure if exists addStep $`
	`create procedure addStep(in descriptionStep varchar(300), timeStep int, photoStep varchar(200), videoStep varchar(200))`
	`language sql`
	`sql security invoker`
	`comment 'Add a step'`
	`begin`
	`  if isStepUnique(descriptionStep) then`
	`  Insert`
	`  into Step (descriptionStep, timeStep, photoStep, videoStep)`
	`  values (descriptionStep, timeStep, photoStep, videoStep);`
	`  end if;`
	`end $`

Recipe	Potrebbe sembrare che *authorName* e/o *authorLink* possano dipendere funzionalmente da *nameRecipe*, ma ci preme ricordare che gli utenti potranno inserire più versioni della stessa ricetta utilizzando lo stesso nome, pertanto nessun attributo può dipendere funzionalmente da *nameRecipe*.

4 Codifica SQL

4.1 Definizione dello schema

```
drop database if exists CustomCooking;
create database CustomCooking;
use CustomCooking;

```

```
5.  create table Account (
6.      username varchar(30) not null,
7.      pass varchar(32) not null,
8.      email varchar(100),
9.      creationTime datetime,
10.     avatar varchar(200),
11.
12.     primary key(username)
13. ) engine=InnoDB;
14.
15. create table Recipe (
16.     idRecipe int not null auto_increment,
17.     nameRecipe varchar(30) not null,
18.     typeRecipe varchar(30) not null,
19.     descriptionRecipe text,
20.     cuisine varchar(30),
21.     regionalOrigin varchar(30),
22.     preparationTime int not null,
23.     difficultyAuthor int not null,
24.     difficultyUsers double,
25.     authorName varchar(30),
26.     authorLink varchar(300),
27.     username varchar(30) not null,
28.     rating double,
29.
30.     primary key(idRecipe),
31.     foreign key(username) references Account(username)
32. ) engine=InnoDB;
33.
34. create table Step (
35.     idStep int not null auto_increment,
36.     descriptionStep varchar(300) not null,
37.     timeStep int not null,
38.     photoStep varchar(200),
39.     videoStep varchar(200),
40.
41.     primary key(idStep)
42. ) engine=InnoDB;
43.
44. create table Product (
45.     nameProduct varchar(30) not null,
46.     classProduct varchar(30) not null,
47.     unityOfMeasure varchar(10) not null,
48.     subclassProduct varchar(30),
49.     photoProduct varchar(200),
50.
51.     primary key(nameProduct)
52. ) engine=InnoDB;
53.
54. create table Tool (
55.     nameTool varchar(30) not null,
56.     photoTool varchar(200),
57.
58.     primary key(nameTool)
59. ) engine=InnoDB;
60.
61. create table Ingredient (
62.     idRecipe int not null,
63.     nameProduct varchar(30) not null,
64.     quantity double not null,
65.     optional boolean not null default false,
66.
67.     primary key(idRecipe, nameProduct),
68.     foreign key(idRecipe) references Recipe(idRecipe),
69.     foreign key(nameProduct) references Product(nameProduct)
```

```
70. ) engine=InnoDB;
71.
72. create table ProductAvailability (
73.     username varchar(30) not null,
74.     nameProduct varchar(30) not null,
75.     quantity double not null,
76.
77.     primary key(username, nameProduct),
78.     foreign key(username) references Account(username),
79.     foreign key(nameProduct) references Product(nameProduct)
80. ) engine=InnoDB;
81.
82. create table ToolAvailability (
83.     username varchar(30) not null,
84.     nameTool varchar(30) not null,
85.
86.     primary key(username, nameTool),
87.     foreign key(username) references Account(username),
88.     foreign key(nameTool) references Tool(nameTool)
89. ) engine=InnoDB;
90.
91. create table Rating (
92.     username varchar(30) not null,
93.     idRecipe int not null,
94.     commentRating text,
95.     rating int not null,
96.     difficulty int,
97.
98.     primary key(username, idRecipe),
99.     foreign key(username) references Account(username),
100.    foreign key(idRecipe) references Recipe(idRecipe)
101.) engine=InnoDB;
102.
103.create table Sequence (
104.    idRecipe int not null,
105.    idStep int not null,
106.    position int not null,
107.
108.    primary key(idRecipe, idStep),
109.    foreign key(idRecipe) references Recipe(idRecipe),
110.    foreign key(idStep) references Step(idStep)
111.) engine=InnoDB;
112.
113.create table ToolSet (
114.    idRecipe int not null,
115.    nameTool varchar(30) not null,
116.    optional boolean not null,
117.
118.    primary key(idRecipe, nameTool),
119.    foreign key(idRecipe) references Recipe(idRecipe),
120.    foreign key(nameTool) references Tool(nameTool)
121.) engine=InnoDB;
122.
123.create table Author (
124.    idRecipe int not null,
125.    username varchar(30) not null,
126.
127.    primary key(idRecipe, username),
128.    foreign key(idRecipe) references Recipe(idRecipe),
129.    foreign key(username) references Account(username)
130.) engine=InnoDB;
```

4.2 Codifica delle operazioni

```
/* Stored Procedures */
DELIMITER $

create function isStepUnique(description varchar(300)) returns boolean
begin
    return (select count(*) from Step where descriptionStep = description) = 0;
end $

drop procedure if exists addAuthor $
create procedure addAuthor(in username varchar(30), pass varchar(30), email varchar(30))
language sql
sql security invoker
comment 'Add an author'
begin
    insert into Account values (username, md5(pass), email, now());
end $

drop procedure if exists addProduct $
create procedure addProduct(in nameProduct varchar(30), classProduct varchar(30), unityOfMeasure varch
ar(10), subclassProduct varchar(30), photoProduct varchar(200))
language sql
sql security invoker
comment 'Add a product'
begin
    insert into Product values (nameProduct, classProduct, unityOfMeasure, subclassProduct, photoProdu
ct);
end $

drop procedure if exists addTool $
create procedure addTool(in nameTool varchar(30), photoTool varchar(200))
language sql
sql security invoker
comment 'Add a tool'
begin
    insert into Tool values (nameTool, photoTool);
end $

drop procedure if exists addIngredient $
create procedure addIngredient(in idRecipe integer, nameProduct varchar(30), quantity double, optional
 boolean)
language sql
sql security invoker
comment 'Add an ingredient'
begin
    insert into Ingredient values (idRecipe, nameProduct, quantity, optional);
end $

drop procedure if exists addAvailability $
create procedure addAvailability(in username varchar(30), nameProduct varchar(30), quantity double)
language sql
sql security invoker
comment 'Add an availability'
begin
    insert into ProductAvailability values (username, nameProduct, quantity);
end $

drop procedure if exists addStep $
create procedure addStep(in descriptionStep varchar(300), timeStep int, photoStep varchar(200), videoS
tep varchar(200))
language sql
```

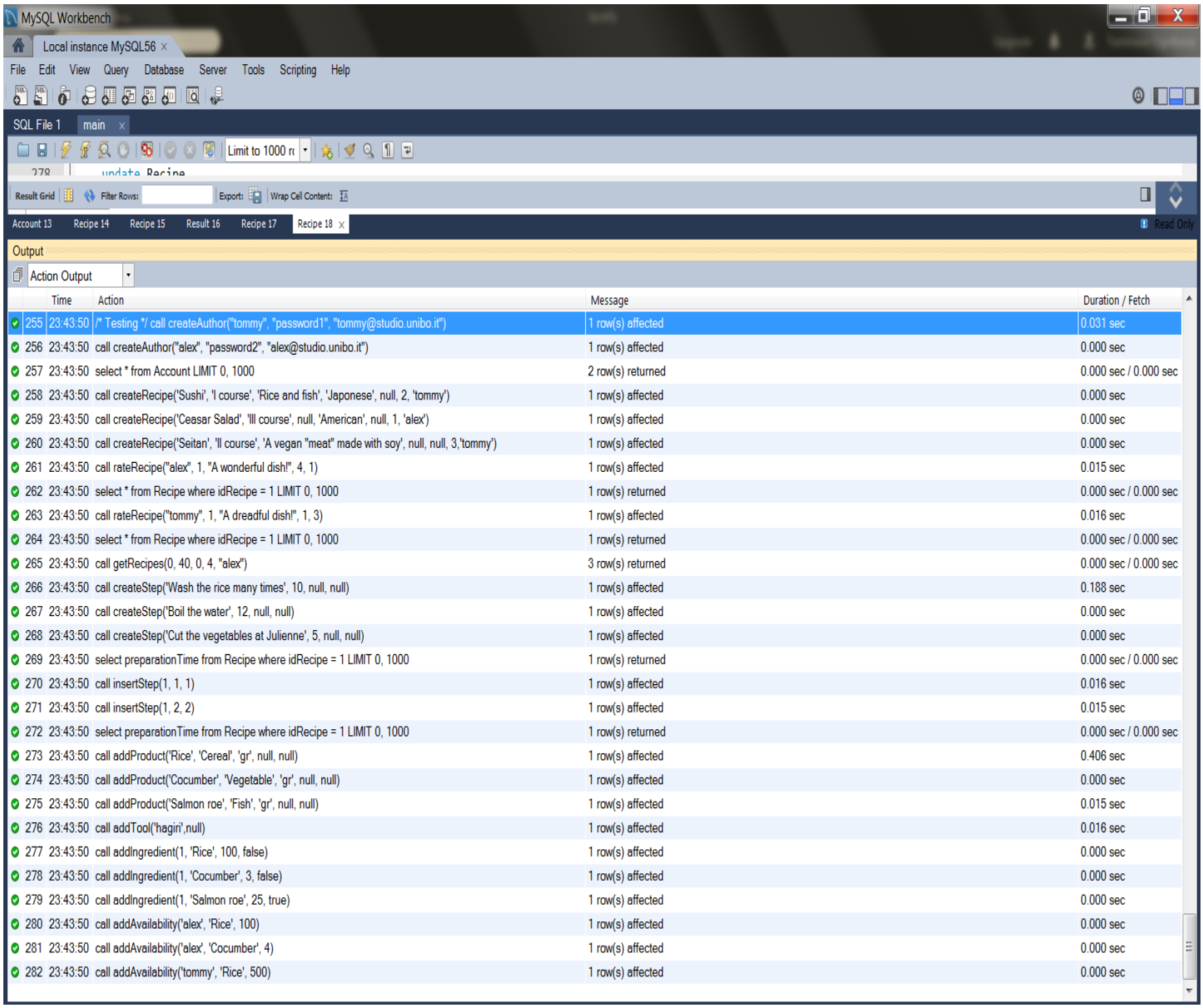
```
sql security invoker
comment 'Add a step'
begin
    if isStepUnique(descriptionStep) then
        Insert
           into Step (descriptionStep, timeStep, photoStep, videoStep)
           values (descriptionStep, timeStep, photoStep, videoStep);
    end if;
end $

drop procedure if exists insertStep $
create procedure insertStep(in idRecipe integer, idStep integer, position integer)
language sql
sql security invoker
comment 'Insert a step in a sequence'
begin
    insert into Sequence values (idRecipe, idStep, position);
end $

drop procedure if exists createRecipe $
create procedure createRecipe(in nameRecipe varchar(30), typeRecipe varchar(30), descriptionRecipe tex
t,
    cuisine varchar(30), regionalOrigin varchar(30), difficulty integer, username varchar(30))
language sql
sql security invoker
comment 'Create a recipe'
begin
    insert into Recipe (nameRecipe, typeRecipe, descriptionRecipe, cuisine, regionalOrigin, preparatio
nTime, difficultyAuthor, username)
        values (nameRecipe, typeRecipe, descriptionRecipe, cuisine, regionalOrigin, 0, difficultyAutho
r, username);
end $

drop procedure if exists rateRecipe $
create procedure rateRecipe(in username varchar(30), idRecipe int, commentRating text, rating int, dif
ficulty int)
language sql
sql security invoker
comment 'Rate a recipe'
begin
    insert into Rating values (username, idRecipe, commentRating, rating, difficulty);
end $

drop procedure if exists getRecipes $
create procedure getRecipes(in timeMin int, timeMax int, diffMin int, diffMax int, username varchar(30
))
language sql
sql security invoker
comment
'Get the possible recipes for a given user accordingly by time, difficulty and avaiability, ordered by
 rating'
begin
    select Recipe.*
    from Recipe
    where
        preparationTime between timeMin and timeMax and
        (
            (difficultyAuthor between diffMin and diffMax) or
            (difficultyUsers between diffMin and diffMax)
        ) and
        idRecipe not in (
            select idRecipe
            from Ingredient i
            where
                i.optional = false and
```

```
                not exists (
                    select *
                    from ProductAvailability a
                    where
                        a.username = username and
                        i.nameProduct = a.nameProduct and
                        i.quantity <= a.quantity))
    order by rating desc;
end $

/* Automatically update the overall rating of a recipe. */
drop trigger if exists updateRating $
create trigger updateRating
after insert on Rating
for each row
begin
    declare overallRating double;
    declare overallDifficulty double;

    select avg(rating) into overallRating
    from Rating
    where idRecipe = new.idRecipe;

    update Recipe
    set rating = overallRating
    where idRecipe = new.idRecipe;

    if (new.difficulty != null) then
        select avg(difficulty) into overallDifficulty
        from Rating
        where idRecipe = new.idRecipe;

        update Recipe
        set difficultyUsers = overallDifficulty
        where idRecipe = new.idRecipe;
    end if;
end $

/* Automatically update the overall time needed for a recipe. */
drop trigger if exists updateTime $
create trigger updateTime
after insert on Sequence
for each row
begin
    declare overallTime int;

    select sum(timeStep) into overallTime
    from Sequence natural join Step
    where idRecipe = new.idRecipe;

    update Recipe
    set preparationTime = overallTime
    where idRecipe = new.idRecipe;
end $

DELIMITER;
```

5 Testing

```
call addAuthor("tommy", "password1", "tommy@studio.unibo.it");
call addAuthor("alex", "password2", "alex@studio.unibo.it");

select * from Account;
```

```
5.
6.  call createRecipe('Sushi', 'I course', 'Rice and fish', 'Japonese', null, 2, 'tommy');
7.  call createRecipe('Ceasar Salad', 'III course', null, 'American', null, 1, 'alex');
8.  call createRecipe('Seitan', 'II course', 'A vegan "meat" made with soy', null, null, 3,'tommy');
9.
10. call rateRecipe("alex", 1, "A wonderful dish!", 4, 1);
11.
12. select * from Recipe where idRecipe = 1;
13.
14. call rateRecipe("tommy", 1, "A dreadful dish!", 1, 3);
15.
16. select * from Recipe where idRecipe = 1;
17.
18. call getRecipes(0, 40, 0, 4, "alex");
19.
20. call createStep('Wash the rice many times', 10, null, null);
21. call createStep('Boil the water', 12, null, null);
22. call createStep('Cut the vegetables at Julienne', 5, null, null);
23.
24. select preparationTime from Recipe where idRecipe = 1;
25.
26. call insertStep(1, 1, 1);
27. call insertStep(1, 2, 2);
28.
29. select preparationTime from Recipe where idRecipe = 1;
30.
31. call addProduct('Rice', 'Cereal', 'gr', null, null);
32. call addProduct('Cocumber', 'Vegetable', 'gr', null, null);
33. call addProduct('Salmon roe', 'Fish', 'gr', null, null);
34.
35. call addTool('hangiri',null);
36.
37. call addIngredient(1, 'Rice', 100, false);
38. call addIngredient(1, 'Cocumber', 3, false);
39. call addIngredient(1, 'Salmon roe', 25, true);
40.
41. call addAvailability('alex', 'Rice', 100);
42. call addAvailability('alex', 'Cocumber', 4);
43. call addAvailability('tommy', 'Rice', 500);
```


MySQL Workbench
Local instance MySQL56
File Edit View Query Database Server Tools Scripting Help
SQL File 1 main
Limit to 1000 r
Result Grid Filter Rows: Export: Wrap Cell Content:
Account 13 Recipe 14 Recipe 15 Result 16 Recipe 17 Recipe 18
Read Only
Output
Action Output
Time Action Message Duration / Fetch
255 23:43:50 /* Testing */ call createAuthor("tommy", "password1", "tommy@studio.unibo.it") 1 row(s) affected 0.031 sec
256 23:43:50 call createAuthor("alex", "password2", "alex@studio.unibo.it") 1 row(s) affected 0.000 sec
257 23:43:50 select * from Account LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
258 23:43:50 call createRecipe('Sushi', 'I course', 'Rice and fish', 'Japonese', null, 2, 'tommy') 1 row(s) affected 0.000 sec
259 23:43:50 call createRecipe('Ceasar Salad', 'III course', null, 'American', null, 1, 'alex') 1 row(s) affected 0.000 sec
260 23:43:50 call createRecipe('Seitan', 'II course', 'A vegan "meat" made with soy', null, null, 3,'tommy') 1 row(s) affected 0.000 sec
261 23:43:50 call rateRecipe("alex", 1, "A wonderful dish!", 4, 1) 1 row(s) affected 0.015 sec
262 23:43:50 select * from Recipe where idRecipe = 1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
263 23:43:50 call rateRecipe("tommy", 1, "A dreadful dish!", 1, 3) 1 row(s) affected 0.016 sec
264 23:43:50 select * from Recipe where idRecipe = 1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
265 23:43:50 call getRecipes(0, 40, 0, 4, "alex") 3 row(s) returned 0.000 sec / 0.000 sec
266 23:43:50 call createStep('Wash the rice many times', 10, null, null) 1 row(s) affected 0.188 sec
267 23:43:50 call createStep('Boil the water', 12, null, null) 1 row(s) affected 0.000 sec
268 23:43:50 call createStep('Cut the vegetables at Julienne', 5, null, null) 1 row(s) affected 0.000 sec
269 23:43:50 select preparationTime from Recipe where idRecipe = 1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
270 23:43:50 call insertStep(1, 1, 1) 1 row(s) affected 0.016 sec
271 23:43:50 call insertStep(1, 2, 2) 1 row(s) affected 0.015 sec
272 23:43:50 select preparationTime from Recipe where idRecipe = 1 LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
273 23:43:50 call addProduct('Rice', 'Cereal', 'gr', null, null) 1 row(s) affected 0.406 sec
274 23:43:50 call addProduct('Cocumber', 'Vegetable', 'gr', null, null) 1 row(s) affected 0.000 sec
275 23:43:50 call addProduct('Salmon roe', 'Fish', 'gr', null, null) 1 row(s) affected 0.015 sec
276 23:43:50 call addTool('hagiri',null) 1 row(s) affected 0.016 sec
277 23:43:50 call addIngredient(1, 'Rice', 100, false) 1 row(s) affected 0.000 sec
278 23:43:50 call addIngredient(1, 'Cocumber', 3, false) 1 row(s) affected 0.000 sec
279 23:43:50 call addIngredient(1, 'Salmon roe', 25, true) 1 row(s) affected 0.000 sec
280 23:43:50 call addAvailability('alex', 'Rice', 100) 1 row(s) affected 0.000 sec
281 23:43:50 call addAvailability('alex', 'Cocumber', 4) 1 row(s) affected 0.000 sec
282 23:43:50 call addAvailability('tommy', 'Rice', 500) 1 row(s) affected 0.000 sec