Data bases 2

Optional project

Riccardo Nannini 10626268

Antonio Ercolani 10621728

Specifications

An application deals with gamified consumer data collection. A user registers with a username, a password and an email. A registered user logs in and accesses a HOME PAGE where a “Questionnaire of the day” is published.

The HOME PAGE displays the name and the image of the “product of the day” and the product reviews by other users. The HOME PAGE comprises a link to access a QUESTIONNAIRE PAGE with a questionnaire divided in two sections: a section with a variable number of marketing questions about the product of the day (e.g., Q1: “Do you know the product?” Q2: Have you purchased the product before?” and Q3 “Would you recommend the product to a friend?”) and a section with fixed inputs for collecting statistical data about the user: age, sex, expertise level (low, medium high). The user fills in the marketing section, then accesses (with a *next* button) the statistical section where she can complete the questionnaire and submit it (with a *submit* button), cancel it (with a *cancel* button), or go back to the previous section and change the answers (with a *previous* button). All inputs of the marketing section are mandatory. All inputs of the statistical section are optional.

After successfully submitting the questionnaire, the user is routed to a page with a thanks and greetings message.

The database contains a table of offensive words. If any response of the user contains a word listed in the table, the transaction is rolled back, no data are recorded in the database, and the user’s account is blocked so that no questionnaires can be filled in by such account in the future.

Specifications

When the user submits the questionnaire one or more trigger compute the gamification points to assign to the user for the specific questionnaire, according to the following rule:

1. One point is assigned for every answered question of section 1 (remember that the number of questions can vary in different questionnaires).

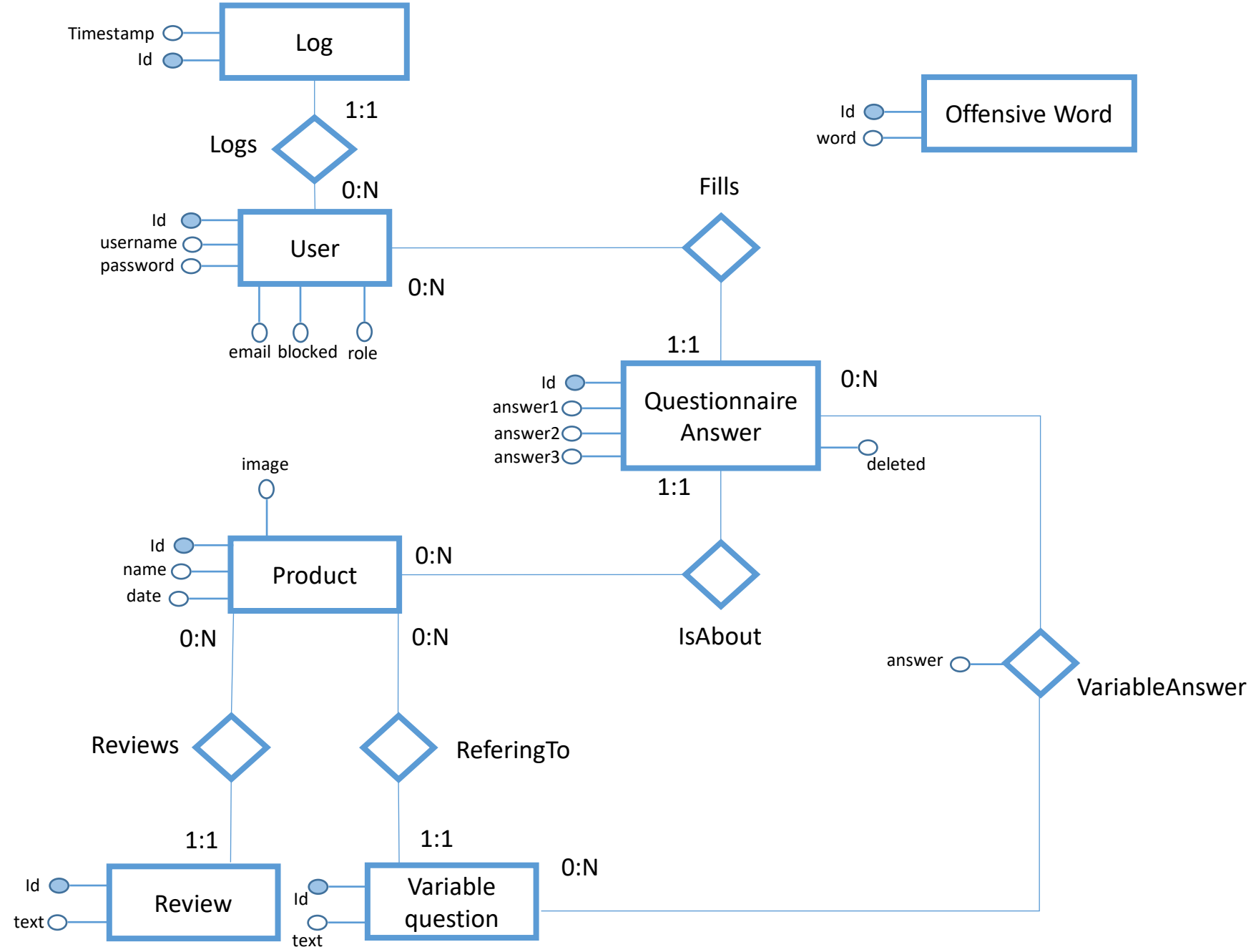
2. Two points are assigned for every answered optional question of section 2.

When the user cancels the questionnaire, no responses are stored in the database. However, the database retains the information that the user X has logged in at a given date and time.

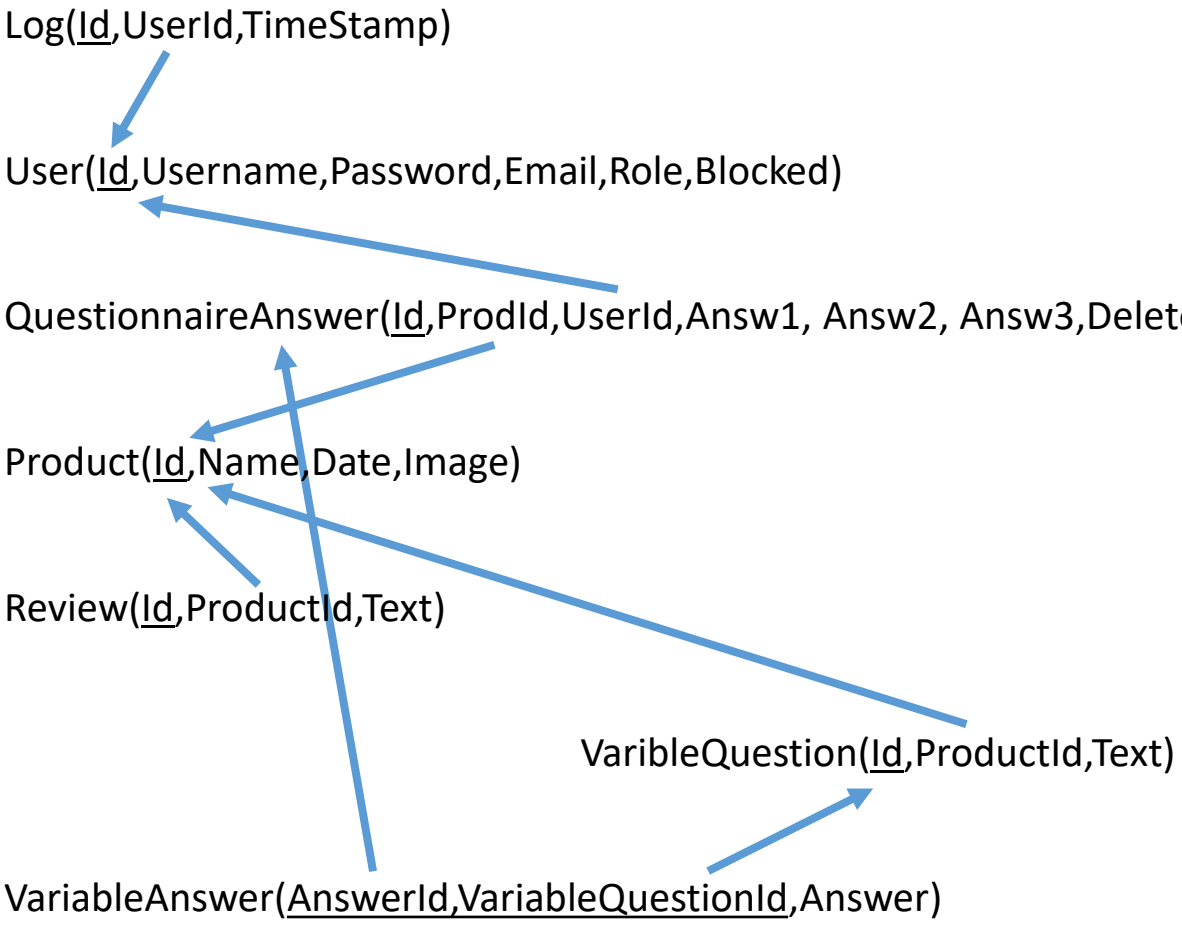
The user can access a LEADERBOARD page, which shows a list of the usernames and points of all the users who filled in the questionnaire of the day, ordered by the number of points (descending).

The administrator can access a dedicated application on the same database, which features the following pages

- A CREATION page for inserting the product of the day for the current date or for a posterior date and for creating a variable number of marketing questions about such product.
- An INSPECTION page for accessing the data of a past questionnaire. The visualized data for a given questionnaire include
    - List of users who submitted the questionnaire.
    - List of users who cancelled the questionnaire.
    - Questionnaire answers of each user.
- A DELETION page for ERASING the questionnaire data and the related responses and points of all users who filled in the questionnaire. Deletion should be possible only for a date preceding the current date.


Timestamp
Id
Log
1:1
Logs
0:N
Id
username
password
User
email
blocked
role
0:N
Fills
1:1
Id
answer1
answer2
answer3
Questionnaire Answer
0:N
deleted
1:1
Id
Offensive Word
word
image
Id
name
date
Product
0:N
0:N
0:N
IsAbout
answer
VariableAnswer
Reviews
ReferingTo
1:1
1:1
Id
text
Review
Id
text
Variable question
0:N

Relational model



Motivations

Questionnaire Answer table, associated to one user and one product, represents the answers to the statistical part of the questionnaire (fixed questions → one column per question).

Variable Question table, associated to one product, represents the marketing questions of the product's questionnaire. Since they vary in number, they can't be column of a table but they're instead tuples of this table.

SQL DDL

```
CREATE TABLE `user` (
`id` int NOT NULL AUTO_INCREMENT,
`username` varchar(45) NOT NULL,
`password` varchar(100) NOT NULL,
`email` varchar(100) NOT NULL,
`role` varchar(10) NOT NULL,
`blocked` tinyint NOT NULL,
 PRIMARY KEY (`id`)
)
```

```
CREATE TABLE `product` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `date` date NOT NULL,
  `image` longblob NOT NULL,
  PRIMARY KEY (`id`),
)
```

SQL DDL

```
CREATE TABLE `questionnarieAnswer` (
`id` int NOT NULL AUTO_INCREMENT,
`userID` int NOT NULL,
`prodID` int NOT NULL,
`answ1` int NOT NULL,
`answ2` char(1) NOT NULL,
`answ3` varchar(10) NOT NULL,
`deleted` tinyint NOT NULL,
PRIMARY KEY (`id`),
CONSTRAINT `userquestionnaire`
FOREIGN KEY (`userID`) REFERENCES
`user` (`id`) ON DELETE CASCADE ON
UPDATE CASCADE,
CONSTRAINT `prodquestionnaire`
FOREIGN KEY (`prodID`) REFERENCES
`product` (`id`) ON DELETE CASCADE
ON UPDATE CASCADE
)
```

```
CREATE TABLE `log` (
  `id` int NOT NULL AUTO_INCREMENT,
  `userID` int NOT NULL,
  `timestamp` timestamp NOT NULL,
  PRIMARY KEY (`id`),
 CONSTRAINT `userlog` FOREIGN KEY
(`userID`) REFERENCES `user` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE
)
```

SQL DDL

```
CREATE TABLE `review` (
`id` int NOT NULL AUTO_INCREMENT,
`prodID` int NOT NULL,
`text` varchar(500) NOT NULL,
 PRIMARY KEY (`id`),
CONSTRAINT `prodreview` FOREIGN KEY (`prodID`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
)
```

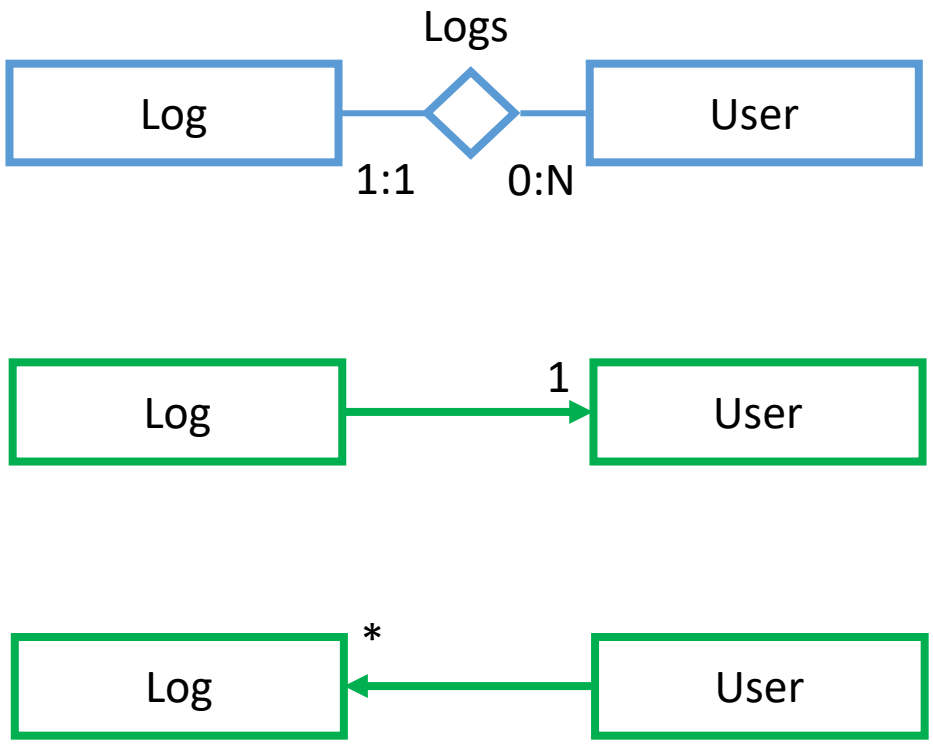
```
CREATE TABLE `variableQuestion` (
  `id` int NOT NULL AUTO_INCREMENT,
  `prodID` int NOT NULL,
  `text` varchar(250) NOT NULL,
  PRIMARY KEY (`id`),
 CONSTRAINT `prodquestion` FOREIGN KEY (`prodID`) REFERENCES `product` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
)
```

SQL DDL

```
CREATE TABLE `variableAnswer` (

`answerIDINCREMENT,

`variableQuestionID` int NOT NULL,

`answer` varchar(500) NOT NULL,

PRIMARY KEY (`answerID`, `
variableQuestionID`),

CONSTRAINT `answers` FOREIGN KEY
(`answerID`) REFERENCES
`questionnarieAnswer` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `question` int NOT NULL
AUTO_` FOREIGN KEY (
variableQuestionID`) REFERENCES
`variableQuestion` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE

)
```

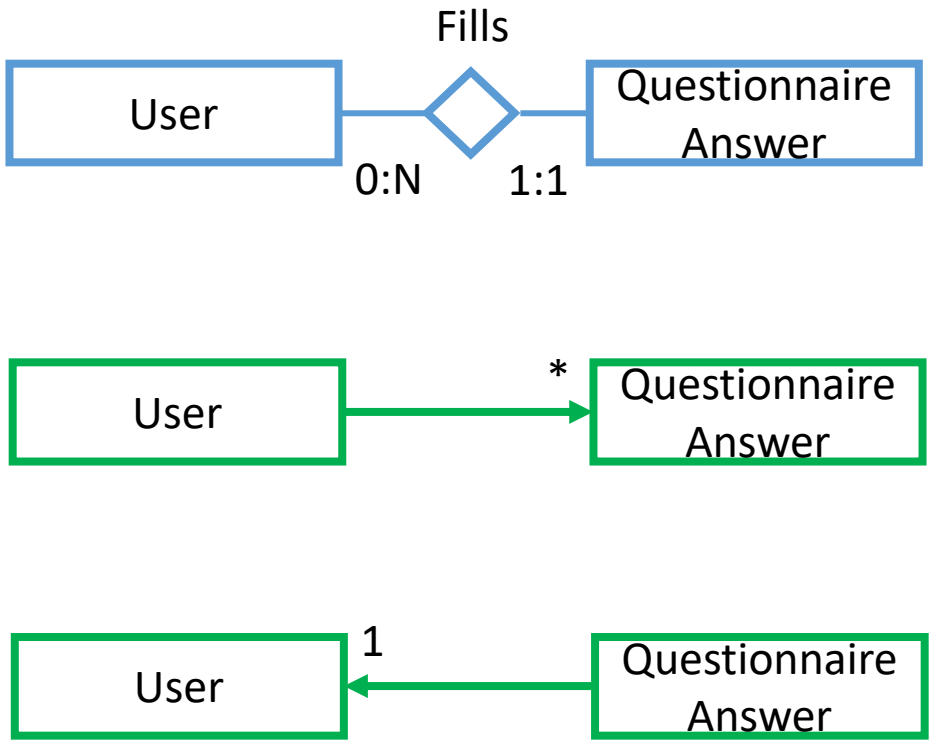
```
CREATE TABLE `offensive_word` (
  `id` int NOT NULL AUTO_INCREMENT,
  `word` varchar(50) NOT NULL,
 PRIMARY KEY (`id`))
```

Relationship “Logs”



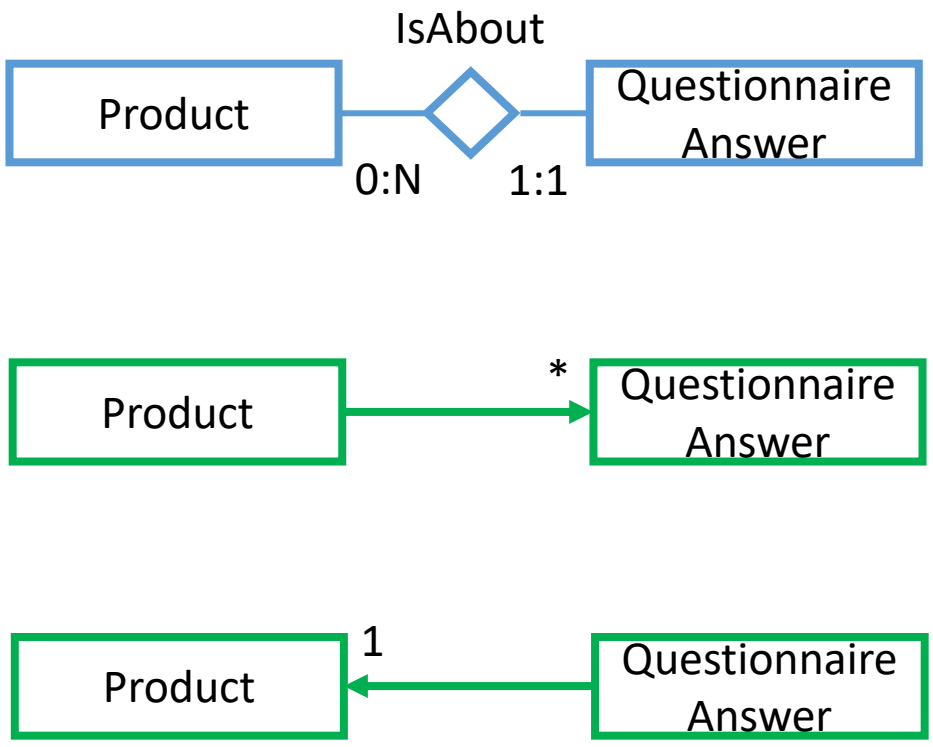
- User → Log @OneToMany
  - Not requested by the specification
- Log → User @ManyToOne
  - Necessary to add to new log info as the user logs in
- Unidirectional 0:N
  - Do not map the @ToMany side of the relationship as a collection data member and use (named) JPQL queries to retrieve the correlated objects when needed

Relationship “Fills”



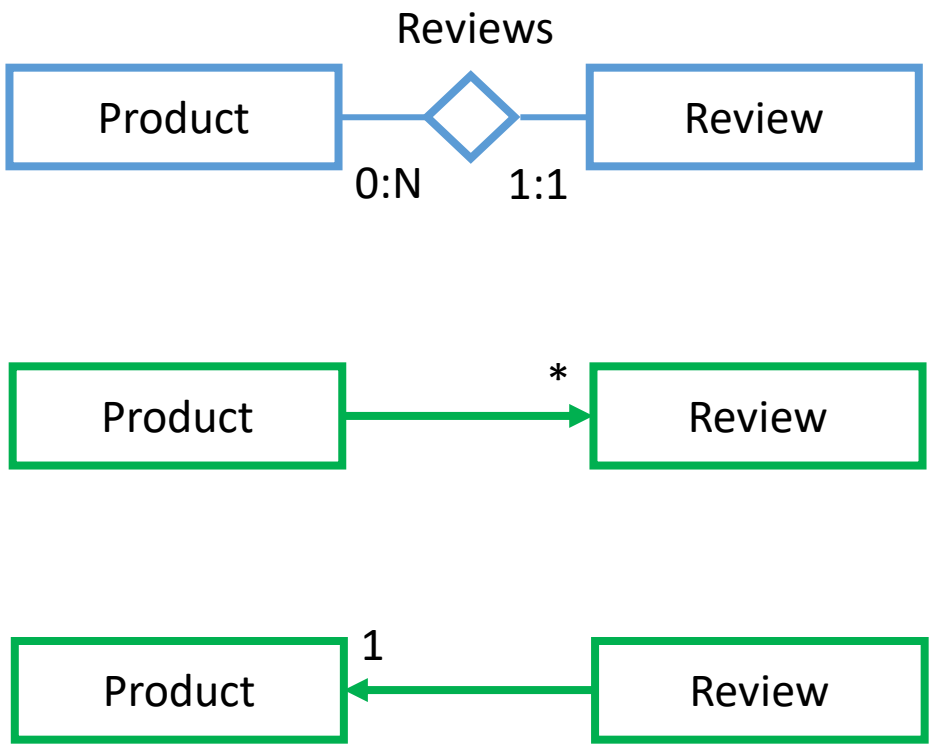
- User → Questionnaire Answer @OneToMany
  - Necessary to retrieve the user questionnaire to check if he has already filled the questionnaire of the day
- Questionnaire Answer → User @ManyToOne
  - Not requested by the specification
- Unidirectional 1:N
  - Mapping the relationship as if it were bidirectional and use only the needed side

Relationship “IsAbout”



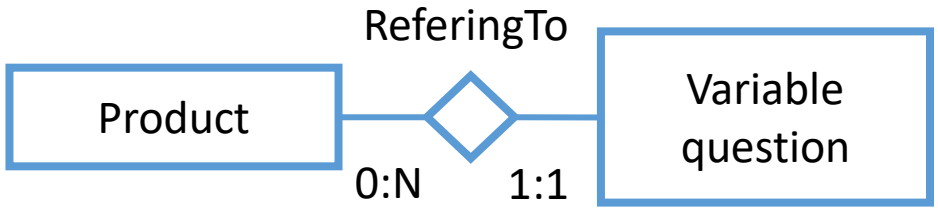
- Product → Questionnaire Answer @OneToMany
  - Not required
- Questionnaire Answers → Product @ManyToOne
  - To retrieve the Questionnaire Answer’s product
- Unidirectional 0:N
  - Do not map the @ToMany side of the relationship as a collection data member and use (named) JPQL queries to retrieve the correlated objects when needed

Relationship “Reviews”



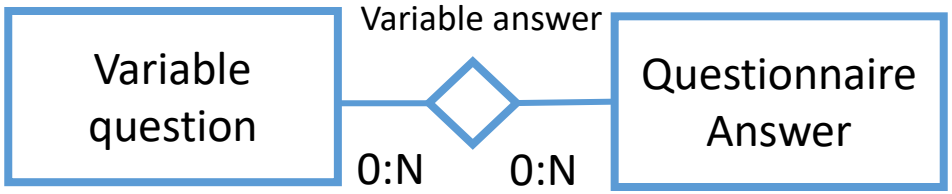
- Product → Review @OneToMany
  - Necessary to retrieve the reviews of a product for the home page
- Review → Product @ManyToOne
  - Not required
- Unidirectional 0:N
  - Mapping the relationship as if it were bidirectional and use only the needed side

Relationship “ReferingTo”



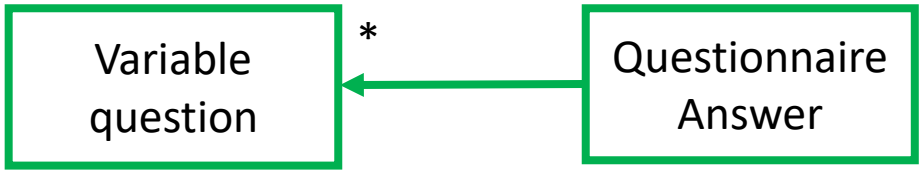
- Product → Variable question @OneToMany
  - Necessary to retrieve the variable question a product when accessing a questionnaire
- Variable question → Product @ManyToOne
  - Necessary to set product of the variable question

Relationship “VariableAnswer”



@ManyToMany

mapped with @ElementCollection



QuestionnaireAnswer → Variable question

- Needed to create the variable answers associated to each variable question

Components

- Client components
  - CancelAnswer
  - CheckLogin
  - CreateAnswer
  - CreateProduct
  - CreateVariableQuestions
  - DeleteProduct
  - GoToAdmin
  - GoToCreationPage
  - GoToDeletePage
  - GoToGreetingPage
  - GoToHomePage
  - GoToLeaderboardPage
  - GoToLogPage
  - GoToMarketingQuestionnaire
  - GoToProductList
  - GoToQuestionnaireInfo
  - GoToStatistical
  - Logout
  - Register

- Views
  - adminHome.html
  - CreateProduct.html
  - CreateProductGreetings.html
  - CreateVariableQuestions.html
  - delete.html
  - deleteProductGreetings.html
  - greetings.html
  - Home.html
  - HomeNoProduct.html
  - Leaderboard.html
  - LogPage.html
  - Marketing.html
  - productList.html
  - questionnaireInfo.html
  - Statistical.html
  - blocked.html
  - index.html

Components

Business Components

- @Stateless UserService
  - `User checkCredentials(String username, String password)`
  - `List<String> findAllUsernames()`
  - `User registerUser(String username, String password, String email)`
  - `void blockUser(User user)`
  - `Boolean hasAlreadyDoneSurvey(Product product, int userID)`

- @Stateless ProductService
  - `List<Product> findProductsByDate(Date date)`
  - `Product createProduct(String name, String date, byte[] img)`
  - `void deleteProduct(int id)`
  - `List<Product> findPastProducts(int prodId)`
  - `Product findProductById(int prodId)`

Components

Business Components

- @Stateless LogService
  - `void createLog(Timestamp timestamp, User user)`
  - `List<Object[]> findUserLogs()`
- @Stateless VariableQuestionService
  - `void CreateVariableQuestion(String text, int productId)`
- @Stateless OffensiveWordService
  - `boolean checkOffensiveWords(List<String> answers)`
- @Stateless QuestionnaireService
  - `List<QuestionnaireAnswer> findQuestionnaireByProduct(Product product)`
  - `List<QuestionnaireAnswer> findQuestionnaireByProductDeleted(Product product)`
  - `List<Object[]> findLeaderbordByProduct(Product product)`

Components

Business Components

- @Statefull QuestionnaireFillingService
  - `void storeMarketingAnswers(List<String> marketingAnswers)`
  - `void cancelQuestionnaire(User user, Product product)`
  - `void createQuestionnaireAnswer(int answ1, String answ2, String answ3, User user, Product product)`

When a user submits the marketing questionnaire we store the answers in the bean until the questionnaire (marketing + statistical) is submitted or deleted

STATEFULL

The others beans are stateless as there is no need to maintain a statefull connection with a specific user

Triggers

Every time a new questionnaire answer is going to be inserted the trigger modifies the points column with the statistical answers points

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`questionnarieanswer_BEFORE_INSERT`
BEFORE INSERT ON `questionnarieanswer`
FOR EACH ROW
BEGIN
   declare sum integer;
   set sum = 0;

   if new.answ1 != 0 then
                        set sum = sum + 2;
   end if;

   if new.answ2 != 'N' then
                        set sum = sum + 2;
   end if;

   if new.answ3 != 'n/d'  then
                        set sum = sum + 2;
   end if;

   set new.points = sum;
END
```

Triggers

```
CREATE DEFINER=`root`@`localhost` TRIGGER `variableanswer_AFTER_INSERT`
AFTER INSERT ON `variableanswer`
FOR EACH ROW
BEGIN
   update `questionnarieanswer`
   set points = points + 1
   where id = new.answerID;
END
```

Every time a new variable answer is inserted the trigger adds one point to the corresponding questionnaire answer