DataBases 2 2020-2021

Gamified Marketing

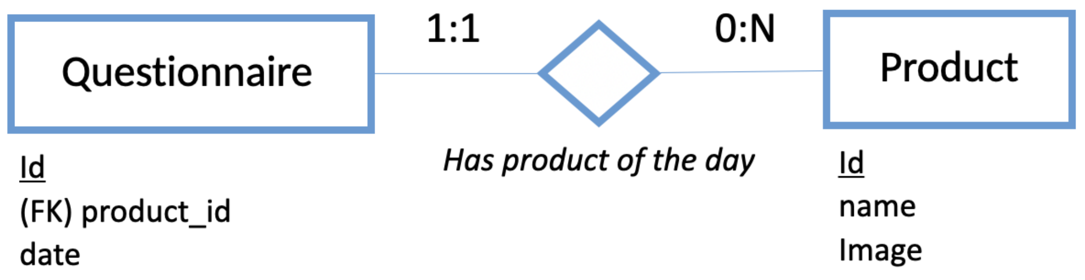
Francesco Attorre

Entity-Relationship Model – Design decisions

- What is "*Product of the day*"
- Composition of a Questionnaire
- No Leaderboard table
- Others

What is "*Product of the day*"

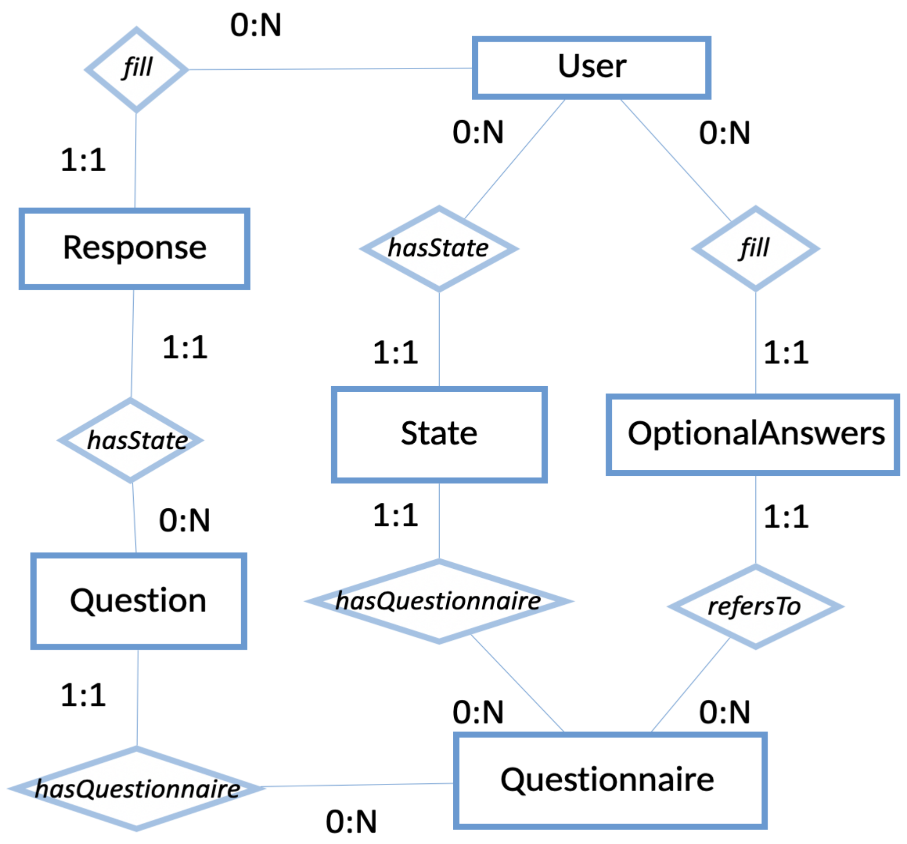
A **Product** becomes "*product of the day*" when a **Questionnaire** related to it is created



Constraint date as unique. It is allowed:

+ A questionnaire for each day and no more (if a questionnaire exist the related product is the product of the day)
+ More questionnaire related to the same product (in this case they are product of the day in different days)

Composition of a Questionnaire 1



1. A questionnaire is composed by its declaration in Questionnaire and a varaible set of Questions.

2. When a questionnaire is filled by the User then will be created :

    a. one Response for each *mandatory* section question

    b. one OptionalAnswers

    c. one State

Composition of a Questionnaire 2

Response →

Text response of a Question

Response(id, *user*, *question*, text)

OptionalAnswers →

Multiple answers or fixed input answers
Assuming that will not change in future

OptionalAnswers(id, *user*, *question*, age, sex, expertise)

State →

Keep track of User activity on a specific Questionnaire
If State tuple does not exist for a specific questionnaire it means that user has not submitted nor cancelled a submission

State(*user*, *questionnaire*, submitted, cancelled)

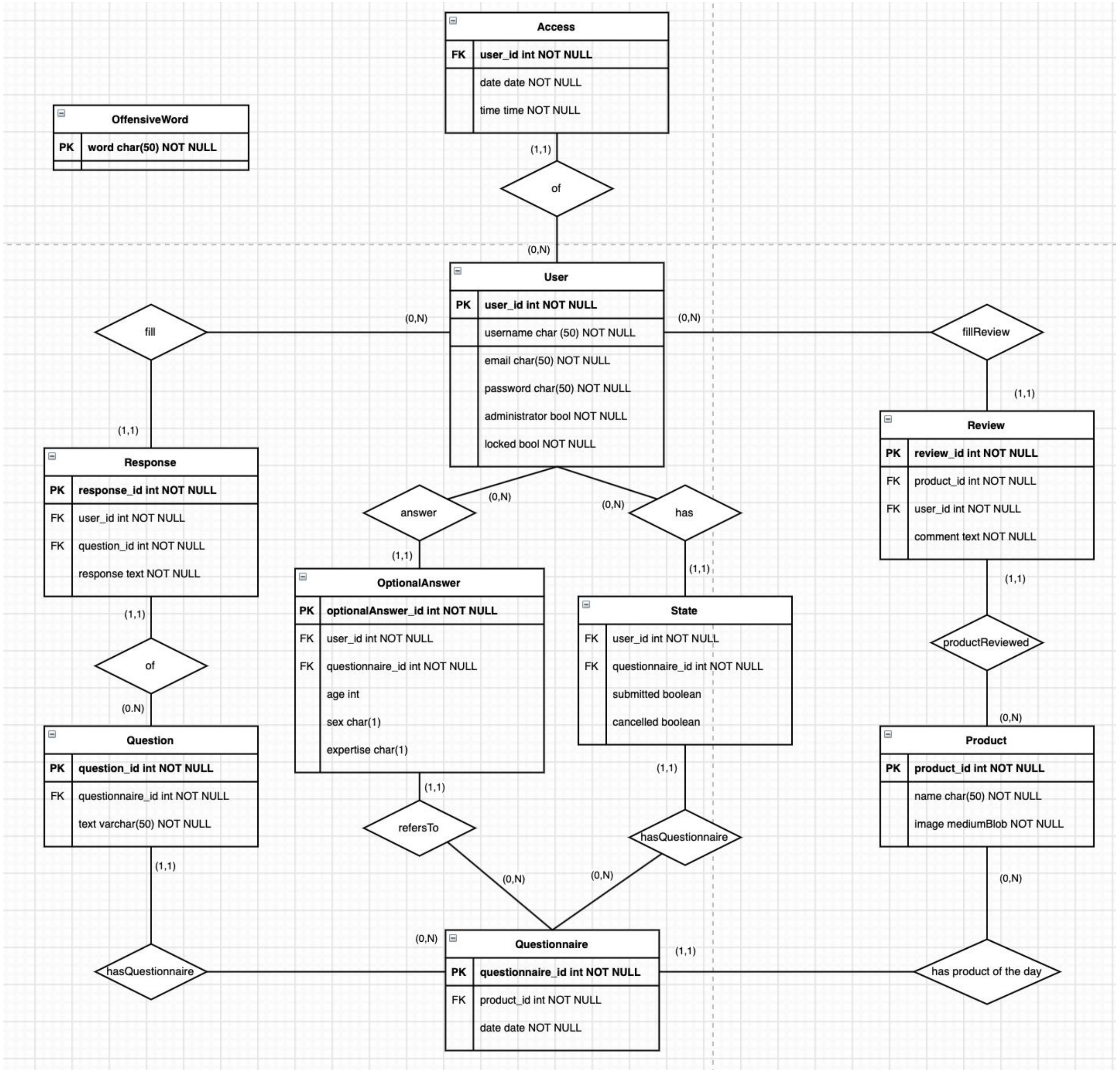
No Leaderboard table

Leaderboard is computed only when requested, by applying the gamification rule and points assignation

It is designed in this way in order to keep simple and avoid a big table that would have contained all the points reached by the user for each questionnaire filled.

Others

- Access(user, date, date) - keep track of log in date of each user
- OffensiveWord(word) - dictionary of words that "bad_words_trigger" uses to check response validity
- Review(id, user, product, comment) - user's review of products


Access
FK user_id int NOT NULL
date date NOT NULL
time time NOT NULL
(1,1)
of
(0,N)
OffensiveWord
PK word char(50) NOT NULL
User
PK user_id int NOT NULL
username char (50) NOT NULL
email char(50) NOT NULL
password char(50) NOT NULL
administrator bool NOT NULL
locked bool NOT NULL
fill
(0,N)
(0,N)
fillReview
(1,1)
(1,1)
Response
PK response_id int NOT NULL
FK user_id int NOT NULL
FK question_id int NOT NULL
response text NOT NULL
Review
PK review_id int NOT NULL
FK product_id int NOT NULL
FK user_id int NOT NULL
comment text NOT NULL
answer
(0,N)
(0,N)
has
(1,1)
(1,1)
OptionalAnswer
PK optionalAnswer_id int NOT NULL
FK user_id int NOT NULL
FK questionnaire_id int NOT NULL
age int
sex char(1)
expertise char(1)
State
FK user_id int NOT NULL
FK questionnaire_id int NOT NULL
submitted boolean
cancelled boolean
(1,1)
of
(0.N)
(1,1)
productReviewed
(0,N)
Question
PK question_id int NOT NULL
FK questionnaire_id int NOT NULL
text varchar(50) NOT NULL
Product
PK product_id int NOT NULL
name char(50) NOT NULL
image mediumBlob NOT NULL
(1,1)
refersTo
(1,1)
hasQuestionnaire
(0,N)
(0,N)
(1,1)
(0,N)
(0,N)
hasQuestionnaire
Questionnaire
PK questionnaire_id int NOT NULL
FK product_id int NOT NULL
date date NOT NULL
(1,1)
has product of the day

Implementation structure description

Structure description :

JPQL oriented implementation, **only fk** relationships are mapped from relational to object oriented style and **others are retreived** by means of queries

Implementation 1

Relevant method of entity's services:

1. ResponseService :

   insertResponses(int writerId, Map<Integer,String> mandatoryResponses, Integer age, char sex,expertise)
   insert *in a unique transaction Response, OptionalAnswer and a State(if not already existent and to be updated)*

2. StateService :

   insertState(int userId, int questionnaireId, boolean submitted, boolean cancelled)
   update *already existent state if user has previously cancelled the questionnaire, else would create a new state*

3. QuestionnaireService :

   deleteQuestionnaire(int questionnaireId)
   since *relationships are not all implemented there is the need to delete constrained tuples before removing questionnaire effectively (this is handmade instead of a cascade.remove setup)*

Implementation 2

Main ***controllers*** :

1. **Response retrieval**
   CancelSubmission – *has a post method that allow an user to cancel the responses inserted up to a moment. No response at all is saved although a State is created to keep track of cancellation event*
   CreateResponse – *save mandatory responses in session, in order to let SubmitResponse to have them availabile*
   SubmitResponse – notify ResponseService to save user's mandatory and optional responses

2. **Questionnaire inspection**
   InspectQuestionnaire – *by means of StateService retrieve users who submitted and users who cancelled a specific questionnaire*
   RetrieveAnswers – *method able to deep in a questionnaire inspection and get all answers of a specific user for a previously specified questionnaire*