INFO-H-303 : Base de données
Projet : Annuaire d'établissements horeca

Frantzen Christian Küpper Marius

12 mai 2016

Modèle entité-association

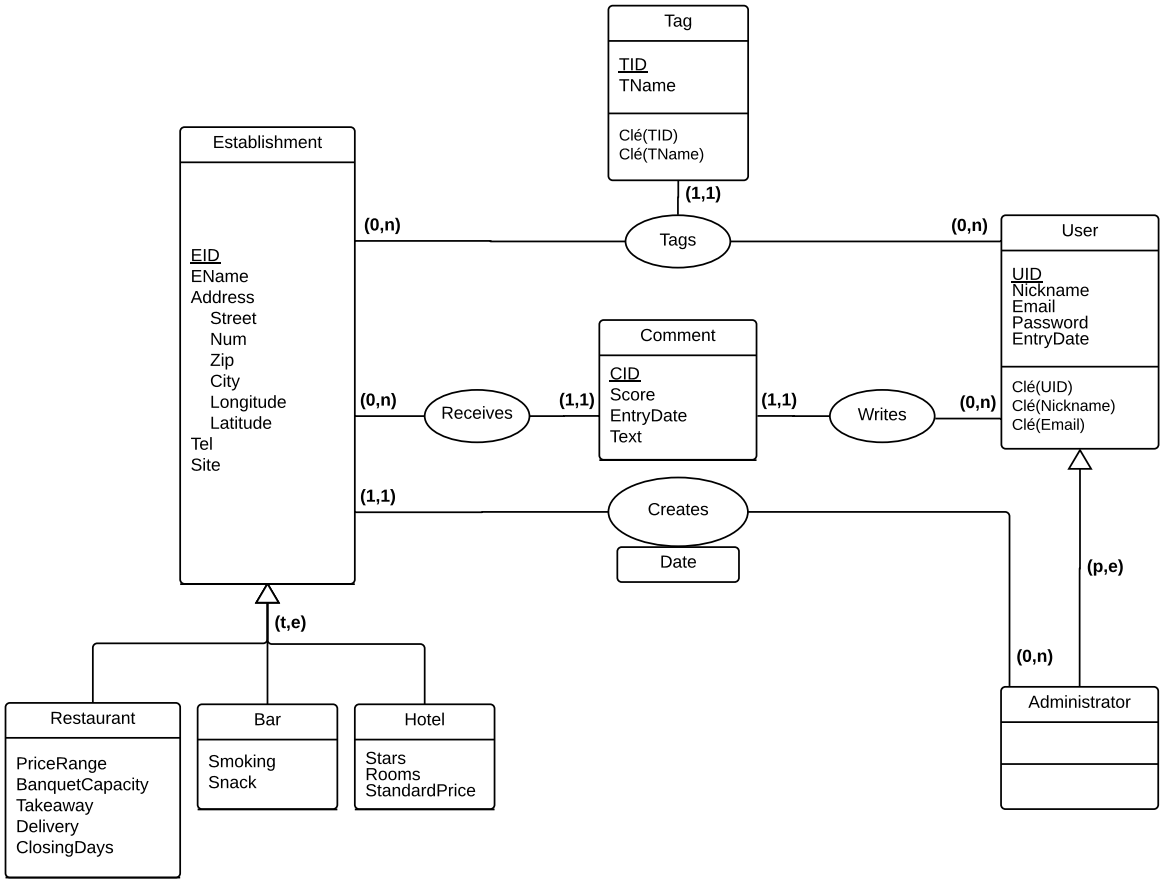

FIGURE 1 – Modèle entité-association

Contraintes d'intégrité

- Un *User* peut commenter plusieurs fois le même *Establishment* à des dates différentes.
- Un *User* ne peut pas apposer le même *Tag* plusieurs fois sur le même *Establishment*.
- L'*EntryDate* d'un *Adminisitrator* doit être strictement supérieure à l'*EntryDate* de l'*Establishment* qu'il a créé.
- L'*EntryDate* d'un *Comment* doit être strictement supérieure à l'*EntryDate* du User qui le fait ainsi que l'*EntryDate* de l'*Establishment* sur lequel il est fait.

Remarques

Si un *Restaurant* ne veut pas organiser de banquet, il spécifie sa *BanquetCapacity* comme étant 0.

Modèle relationnel

Establishment(EID,EName,Street,Num,Zip,City,Longitude,Latitude,Tel,Site,UID,EntryDate)
- UID référence User.UID (représente le createur de l'établissement)

Restaurant(EID, PriceRange, BanquetCapacity, Takeaway, Delivery)
— EID référence Establishment.EID

RestaurantClosingDays(EID, ClosingDay, Hour)
— EID référence Establishment.EID

Bar(EID, Smoking, Snack)
— EID référence Establishment.EID

Hotel(EID, Stars, Rooms, StandardPrice)
— EID référence Establishment.EID

User(UID, Nickname, Email, Password, EntryDate, Admin)
— Nickname est unique et donc également une clé de cette relation
— Email est unique et donc également une clé de cette relation

Comment(CID, UID, EID, EntryDate, Score, Text)
— UID référence User.UID
— EID référence Establishment.EID
— (UID,EID,EntryDate) est unique et donc également une clé de cette relation

Tag(TID, TName)
— TName est unique et donc également une clé de cette relation

EstablishmentTag(TID, EID, UID)
— TID référence Tag.TID
— EID référence Establishment.EID
— UID référence User.UID

Remarques

L'ajout d'une entrée dans *Establishment* implique l'ajout d'une entrée dans soit *Restaurant*, soit *Bar* ou soit *Hotel*. Les deux entrées ont le même *EID*.

Contraintes de domaine

— $User.Admin \in \{True, False\}$ (Seul les *User* avec *User.Admin* = True ont les droits de créer, supprimer ou modifier des *Establishment*).
— Pour les entrées dans *Comment*, *Comment.EntryDate* > *Establishment.EntryDate* et *Comment.EntryDate* > *User.EntryDate*
— $Comment.Score \in \{0, 1, 2, 3, 4, 5\}$
— $Hotel.Stars \in \{0, 1, 2, 3, 4, 5\}$
— $Hotel.Rooms \in \mathbb{N}_0$
— $Hotel.StandardPrice \in \mathbb{R}_0^+$
— $Restaurant.BanquetCapacity \in \mathbb{N}$
— $Restaurant.PriceRange \in \mathbb{R}_0^+$
— $Restaurant.Takeaway \in \{True, False\}$
— $Restaurant.Delivery \in \{True, False\}$
— $RestaurantClosingDays.Hour \in \{"AM", "PM"\}$ (matin / après-midi)
— $RestaurantClosingDays.ClosingDay \in \{0, 1, 2, 3, 4, 5, 6\}$
— $Bar.Smoking \in \{True, False\}$

— $Bar.Snack \in \{True, False\}$

Requêtes

R1

Tous les utilisateurs qui apprécient au moins 3 établissements que l'utilisateur "Brenda" apprécie.

Calcul relationnel tuple

$$\{u|User(u) \wedge \exists e_1 \exists e_2 \exists e_3 (Establishment(e_1) \wedge Establishment(e_2) \wedge Establishment(e_3) \wedge e_1 \neq e_2 \wedge e_1 \neq e_3 \wedge e_2 \neq e_3 \wedge \exists u_2 \exists c_1 \exists c_2 (User(u_2) \wedge Comment(c_1) \wedge Comment(c_2) \wedge u_2.uid \neq u.uid \wedge u_2.nickname = "Brenda" \wedge \forall i \in \{1,2,3\}(c_1.uid = e_i \wedge c_2.uid = e_i \wedge c_1.uid = u.uid \wedge c_2.uid = u_2.uid \wedge c_1.score \geq 4 \wedge c_2.score \geq 4)))\}$$

Algèbre relationnelle

R2

Tous les établissements qu'apprécie au moins un utilisateur qui apprécie tous les établissements que "Brenda" apprécie.

Calcul relationnel tuple

$$\{e|Establishment(e) \wedge \exists u_1 \exists c_1 (User(u_1) \wedge Comment(c_1) \wedge c_1.eid = e.eid \wedge c_1.uid = u_1.uid \wedge u_1.nickname \neq "Brenda" \wedge c_1.score \geq 4 \wedge \exists u_2 \exists c_2 \exists c_3 \exists e_2 (User(u_2) \wedge Comment(c_2) \wedge Comment(c_3) \wedge Establishment(e_2) \wedge u_2.nickname = "Brenda" \wedge c_2.eid = u_2.uid \wedge c_2.eid = e_2.eid \wedge c_3.eid = e_2.eid \wedge c_3.uid = u_1.uid \wedge c_2.uid = u_2.uid \wedge c_2.score \geq 4 \wedge c_3.score \geq 4))\}$$

Algèbre relationnelle

R3

Tous les établissements pour lesquels il y a au plus un commentaire.

Calcul relationnel tuple

$$\{e|Establishment(e) \wedge \exists! c_1 (Comment(c_1) \wedge c_1.eid = e.eid) \vee \nexists c_2 (Comment(c_2) \wedge c_2.eid = e.eid)\}$$

Algèbre relationnelle

R4

La liste des administrateurs n'ayant pas commenté tous les établissements qu'ils ont crées.

Calcul relationnel tuple

$$\{u|User(u) \wedge u.admin = true \wedge \exists e (Establishment(e) \wedge e.uid = u.uid \wedge \nexists c (Comment(c) \wedge c.eid = e.eid \wedge c.uid = u.uid))\}$$

Algèbre relationnelle

Hypothèses

- Deux *User*s ne peuvent pas avoir le même *Nickname* : L'interface doit permettre de consulter la fiche de chaque *User*. Un *User* qui cherche le profil de quelqu'un ne peut pas distinguer deux *User*s avec le même *Nickname* puisque le celui-ci est la seule donnée visible (pas d'image de profil, etc ...).
- Un *Admin* ne peut pas modifier des *Comment*s ni des *Tag*s : On pourrait lui donner les droits de gestion pour vérifier qu'il n'y ait pas d'abus, mais pour le moment un *Admin* ne peut que créer, modifier et enlever des *Establishment*s.
- Pour les requêtes R1 et R2 on exclut l'utilisateur "Brenda" comme étant un résultat qu'on considère lors de la recherche des résultats.