ULB UNIVERSITÉ LIBRE DE BRUXELLES

INFO-H303 : Projet Partie 2

Théo Charlier, Albane Keraudren-Riguidel, Keyla Kamara et Hac Le

May 25, 2024

ULB UNIVERSITÉ LIBRE DE BRUXELLES

Contents

1 Introduction

Ce document modélise toutes les demandes de l'entreprise pour l'implémentation de la base de données.

2 Modélisation

2.1 Diagramme

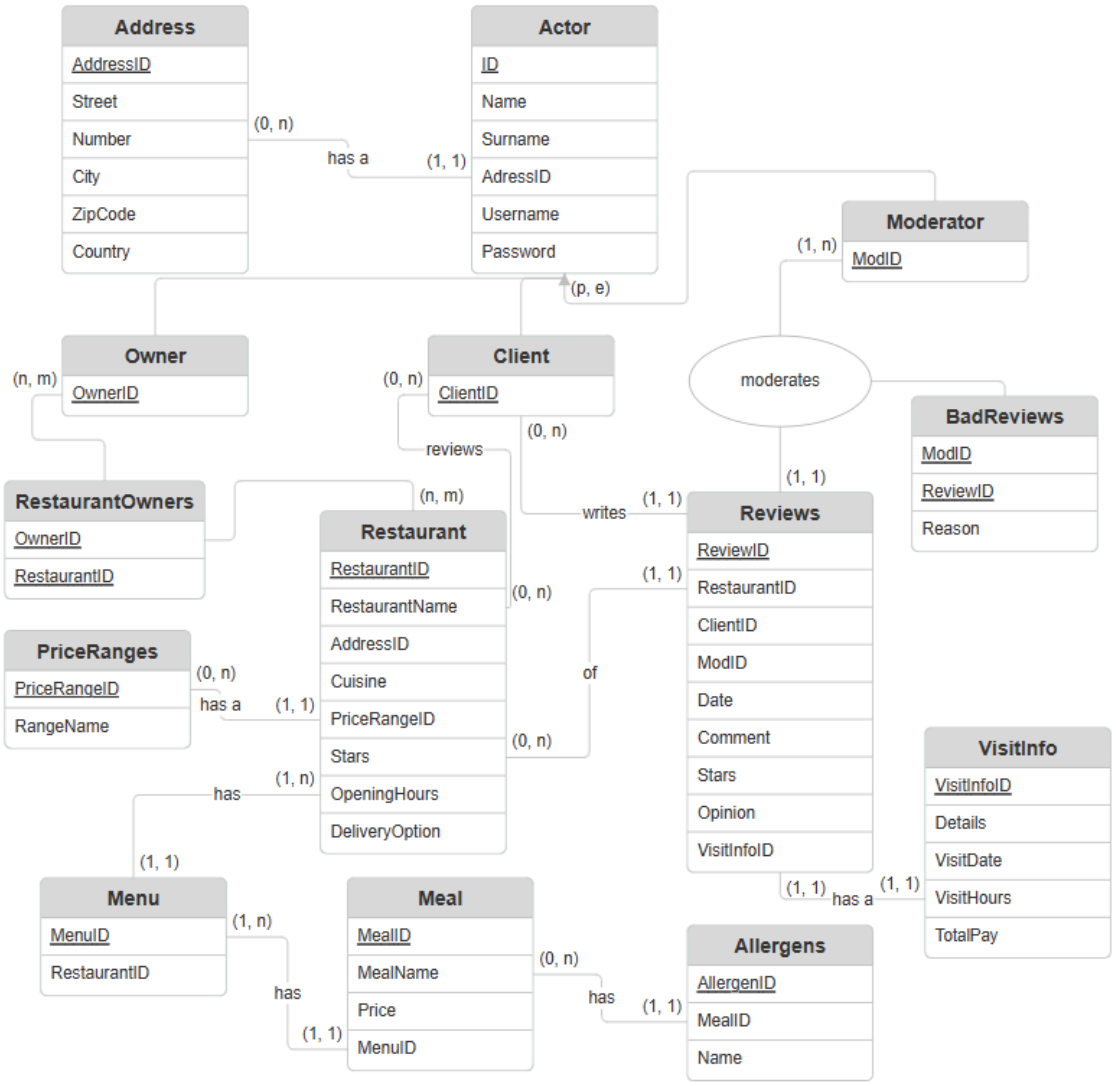

Figure 1: Diagramme entité-association

Ce diagramme représente une version modifiée du diagramme de la phase 1 en prenant en compte les différentes tables de la base de données de notre application.

2.2 Traduction relationelle

Actor (ID, Name, Surname, AddressID, Username, Password)
Actor.AddressID references Address.AddressID

Address (AddressID, Street, Number, City, ZipCode, Country)

Client (ClientID)
Client.ClientID references Actor.ID

Moderator (ModID)
Moderator.ModID references Actor.ID

Owner (OwnerID)
Owner.OwnerID references Actor.ID

RestaurantOwners (OwnerID, RestaurantID)
RestaurantOwners.OwnerID references Owner.OwnerID
RestaurantOwners.RestaurantID references Restaurant.RestaurantID

Restaurant (RestaurantID, RestaurantName, AddressID, Cuisine, PriceRangeID, Stars, OpeningHours, DeliveryOption)
Restaurant.AddressID references Address.AddressID
Restaurant.PriceRangeID references PriceRanges.PriceRangeID

Menu (MenuID, RestaurantID)
Menu.RestaurantID references Restaurant.RestaurantID

Meal (MealID, MealName, Price, MenuID)
Meal.MenuID references Menu.MenuID

Allergens (AllergenID, Name, MealID)
Allergens.MealID references Meal.MealID

PriceRanges (PriceRangeID, RangeName)

Reviews (ReviewID, RestaurantID, ClientID, ModID, VisitInfoID, Date, Comment, Stars, Opinion)
Reviews.ID references Client.ClientID
Reviews.ModID references Moderator.ModID
Reviews.RestaurantID references Restaurant.RestaurantID
Reviews.VisitInfoID references VisitInfo.VisitInfoID

VisitInfo (VisitInfoID, Details, VisitDate, VisitHour, TotalPay)

BadReviews (ModID, ReviewID, Reason)
BadReviews.ModID references Moderator.ModID
BadReviews.ReviewID references Reviews.ReviewID

2.3 Hypothèses et justifications avec modifications lors de la phase 2

- – **phase 1:** *Un Client n'a pas besoin d'un attribut ClientID additionnel vu que ID est son primary key, contrairement à Moderator qui a un ModID*
  – **modification:** Client, Owner et Moderator ont tous leur propre ID (ClientID, OwnerID, ModeratorID) hérité de l'ID de Actor.
- Un Owner ne peut pas écrire/modérer des Reviews pour son propre Restaurant
- Un Restaurant peut avoir plusieurs Menus
- 2 Actors peuvent avoir le même attribut AddressID

3 Requêtes

Règles appliquées:

- D'après le théorème de Codd: "Une requête est exprimable en calcul relationnel si et seulement si elle peut être exprimée par une expression de l'algèbre relationnelle."
- Le tri et la limite du nombre de résultats ne sont pas supportés par l'algèbre linéaire et le calcul relationnel.
- Nous n'utilisons pas la fonction agrégative COUNT() pour l'algèbre linéaire et le calcul relationnel.

3.1 Les restaurants ayant un avis moyen de 3 ou plus

SQL:

```
SELECT Restaurants.RestaurantName
FROM Restaurants
WHERE Restaurants.Stars >= 3;
```

Algèbre relationnelle:

$$\pi_{\text{RestaurantName}}(\sigma_{\text{Stars} \geq 3}(\text{Restaurants}))$$

Calcul tuple:

$$\{\text{name} \mid \text{Restaurants}(\text{RestaurantName} : \text{name}, \text{Stars} : \geq 3)\}$$

Figure 2: Requête 1

3.2 Le restaurant avec le plat le plus cher

SQL:

```
SELECT Restaurants.RestaurantName, Meals.Price
FROM Restaurants
JOIN Menus ON Restaurants.RestaurantID = Menus.RestaurantID
JOIN Meals ON Menus.MenuID = Meals.MenuID
WHERE Meals.Price = (SELECT MAX(Meals.Price) FROM Meals);
```

Algèbre relationnelle:

$$\pi_{\text{RestaurantName, Price}}(\sigma_{(\text{Meals.Price}=(\sigma_{(\max(\text{Meals.Price}))(\text{Meals})}} (\text{Restaurants} \bowtie \text{Menus} \bowtie \text{Meals}))$$

Calcul tuple:

$$\{r.\text{RestaurantName}, \text{Meals.Price} \mid \text{Restaurants}(r) \bigwedge \exists m \in \text{Meals} (\exists n \in \text{Menus} (r.\text{RestaurantID} = n.\text{RestaurantID} \bigwedge n.\text{MenuID} = m.\text{MenuID} \bigwedge m.\text{Price} = (\forall m2 \in \text{Meals} (m2.\text{Price} \leq m.\text{Price})))))\}$$

Figure 3: Requête 2

3.3 Les 10 clients ayant consommé le plus de plats mexicains

```
SELECT Clients.Name
FROM Clients
JOIN Reviews ON Reviews.ClientID = Clients.ClientID
JOIN Restaurants ON Reviews.RestaurantID =
Restaurants.RestaurantID
JOIN Menus ON Restaurants.RestaurantID =
Menus.RestaurantID
JOIN Meals ON Menus.MenuID = Meals.MenuID
WHERE Restaurants.Cuisine = “mexicain”
GROUP BY Clients.ClientID
ORDER BY count(distinct Meals.MealID) desc
LIMIT 10;
```

Figure 4: Requête 3

3.4 Le restaurant non-asiatique proposant le plus de plats qui sont généralement proposés dans des restaurants asiatiques

```
SELECT r1.RestaurantName, Meals.MealName
FROM Restaurants r1
JOIN Menus ON r1.RestaurantID = Menus.RestaurantID
JOIN Meals ON Menus.MenuID = Meals.MenuID
JOIN Restaurants r2 ON r2.Cuisine = “asiatique”
WHERE not r1.Cuisine = “asiatique”
GROUP BY r1.RestaurantName, Meals.MealName
ORDER BY count(distinct Meals.MealID) desc
LIMIT 1;
```

Figure 5: Requête 4

3.5 Le code postal de la ville dans laquelle les restaurants sont les moins bien notés en moyenne

SQL:

```
SELECT Addresses.ZipCode, Restaurants.Stars
FROM Addresses
JOIN Restaurants ON Restaurants.AddressID = Addresses.AddressID
GROUP BY Addresses.ZipCode, Restaurants.Stars
ORDER BY AVG(DISTINCT Restaurants.Stars) ASC
LIMIT 1;
```

Algèbre relationnelle:

$$\pi_{\text{Addresses.ZipCode, Restaurants.Stars}}(\sigma_{\text{Restaurants.Stars=min(Restaurants.Stars)}} (\text{Addresses} \bowtie \text{Restaurants}))$$

Calcul tuple:

$$\{a.\text{ZipCode}, r.\text{Stars} \mid \text{Addresses}(a) \bigwedge \exists r \in \text{Restaurants}\ (r.\text{AddressID} = a.\text{AddressID} \bigwedge r.\text{Price} = (\ \forall r2 \in \text{Restaurants}\ (r.\text{Stars} \leq r2.\text{Stars})))\}$$

Figure 6: Requête 5

3.6 Pour chaque tranche de score moyen de restaurant, le type de nourriture le plus représenté

SQL:

```
SELECT CASE
        WHEN AVG(Stars)  >= 0 AND AVG(Stars)  <= 1 THEN “1/5”
        WHEN AVG(Stars)  >= 1 AND AVG(Stars)  <= 2 THEN “2/5”
        WHEN AVG(Stars)  >= 2 AND AVG(Stars)  <= 3 THEN “3/5”
        WHEN AVG(Stars)  >= 3 AND AVG(Stars)  <= 4 THEN “4/5”
        WHEN AVG(Stars)  >= 4 AND AVG(Stars)  <= 5 THEN “5/5”
        END
        AND Restaurants.Cuisine
FROM Restaurants
GROUP BY Cuisine
ORDER BY Rating_Category ASC;
```

Figure 7: Requête 6