Analyse Projet BDD

Contents

1 Analyse du problème

1.1 Hypothèses

Pour la conception, nous avons fait les choix suivants:

1. Le statut des commandes n'est pas dans l'entité mère COMMANDES mais dans ses sous-entités afin d'éviter d'avoir des statuts non cohérents pour un type de commande donné.
2. Nous avons rajouté le statut "terminée" qui indique qu'une commande a effectivement été effectuée et que l'on peut donner un évaluation
3.
4.

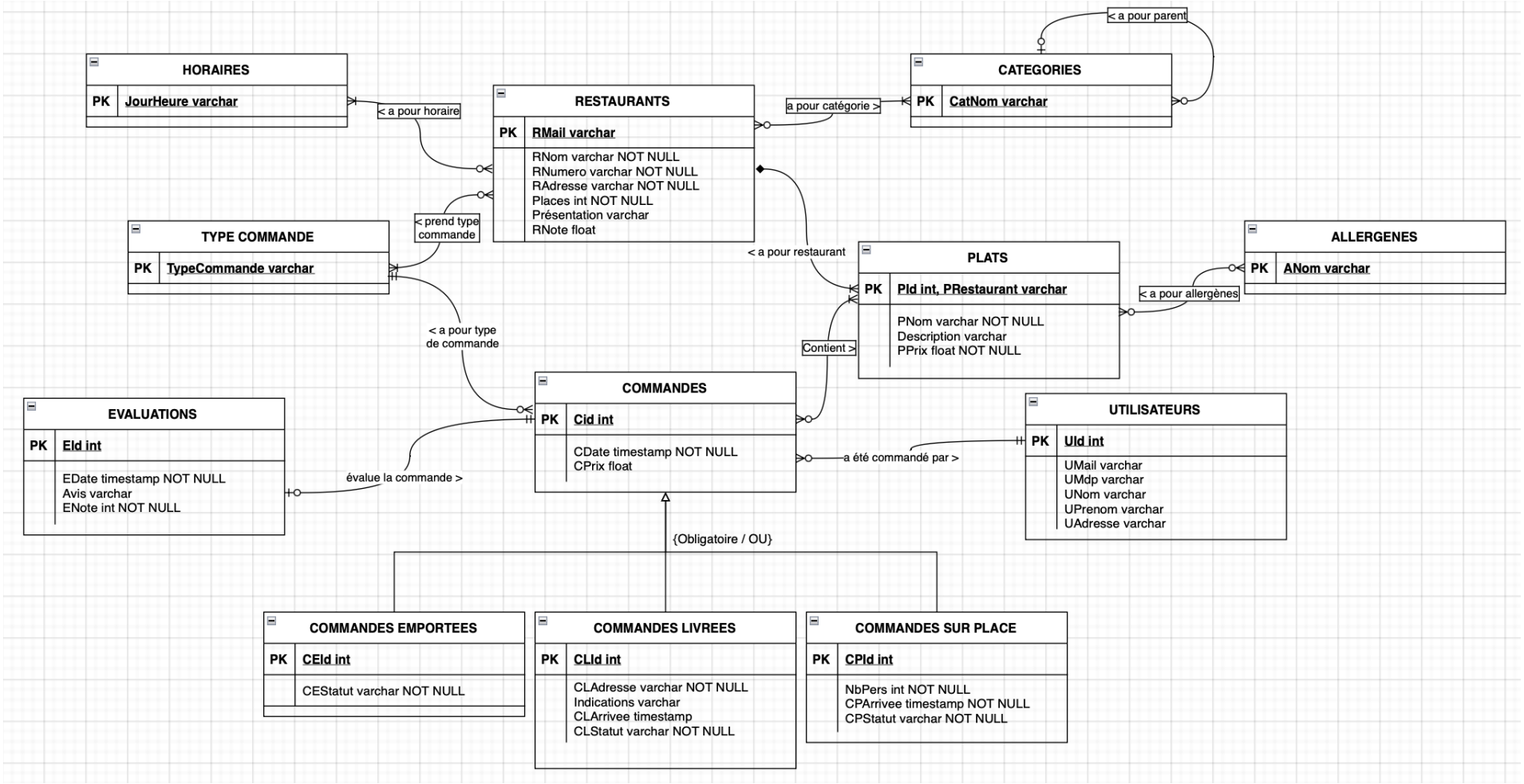
Nous sommes arrivés avec les contraintes suivantes :

1.2 Contraintes

DF	C. Valeurs	C. Contextuelles	C. Multiplicité
RMail → RNom, RNumero, RAdresse, Places, Presentation, RNote	RType ∈ {livraison, emporter, place} Places > 0 RNote ∈ [0, 5]	$\sum$ nbPers ≤ Places pour un restaurant et ses commandes associées Ext(CEId) ∩ Ext(CPId) ∩ Ext(CLId) = ∅ Ext(CEId) ∪ Ext(CPId) ∪ Ext(CLId) = Ext(CId) CPArrivee ∈ JourPlage pour un restaurant et ses commandes associées CDate donne JourPlage pour toute commande CDate ≤ EDate EId ⇒ CId.statut = {terminee} CType ∈ TypeCommande pour un CId donné et le RMail associé	RMail ↠ JourPlage RMail ↠ TypeCommande RMail ↠ PId RMail ↠ CatNom (PId, RMail) ↠↠ ANom CId ↠ (PId, RMail) CId → TypeCommande CId → U_Id CId ↛ EId CatNom ↛ CatNom
(PId, Restaurant) → PNom, PDescription, PPrix	PPrix > 0		
U_Id → UMail, UMdp, UNom, UPrenom, UAdresse			
CId → CDate, CPrix	CPrix > 0		
CLId → CLAdresse, Indications, CLArrivee, CLStatut	CLStatut ∈ { attente, validée, en livraison, annuleeC, annuleeR, terminee }		
CEId → CEStatut	CEStatut ∈ { attente, validée, disponible, annuleeC, annuleeR, terminee }		
CPId → NbPers, CPArrivee, CPStatut	CPStatut ∈ { attente, validée, annuleeC, annuleeR, terminee } NbPers > 0		
EId → EDate, Avis, ENote	ENote ∈ [0..5]		
	JourHeure ∈ {LM, LS, MM, MS, MeM, MeS, JM, JS, VM, VS, SM, SS, DM, DS }		

1.3 Diagramme Entités/Associations

On obtient ensuite le diagramme Entité-Relation suivant:



2 Modèle Relationnel

Nous avons appliqué l'algorithme vu en cours pour obtenir le schéma relationnel.

2.1 Traduction des entités simples

Nous avons identifié les entités simples suivantes:

- Restaurant(RMail, RNom, RNum, RAdresse, Places, Présentation, RNote)
- Commandes(Cid, CDate, CPrix, **Uid**, **TypeCommande**)
- Utilisateurs(U_id, UMail, UMdp, UNom, UPrenom, UAdresse)
- Evaluation(Eid, EDate, Avis, ENote, **Cid**)
- Horaires(JourPlage)
- TypesCommande(TypeCommande)
- Categories(CatNom)
- Allergenes(ANom)

2.2 Sous-types d'entités

Pour une question d'espace occupé nous avons décidé de traduire l'héritage des différentes catégories des commandes par référence. Contrairement à l'unification cela nous permet de ne pas avoir à gérer les contraintes avec l'application.

- CommandesEmportees(CEid, CEStatut)
- CommandesLivrees(CLid, CLAdresse, Indications, CLArrivee, CLStatut)
- CommandesSurPlace(CPid, NbPers, CPArrivee, CPStatut)

2.3 Type d'entité faible

- Plats(Pid, PRestaurant, PNom, Description, PPrix)

Contrainte induite par la traduction: Vérifier que tout restaurant a au moins un plat.

2.4 Types d'associations

2.4.1 Cardinalité 1..1

Les attributs en gras dans les relations de la partie 2.1 correspondent aux traductions des associations avec cardinalité 1..1. Aucune vérification supplémentaire n'est à faire.

2.4.2 Cardinalité 0..1

- CategorieParent(CatNom, CatNomMere)

2.4.3 Cardinalité ?..*

- HorairesRestaurant(RMail, JourPlage)
  Contrainte induite: Vérifier qu'un restaurant a au moins un horaire.
- CategoriesRestaurant(RMail, CatNom)
  Contrainte induite: Vérifier qu'un restaurant a au moins une catégorie
- TypesRestaurant(RMail, TypeCommande)
  Contrainte induite: Vérifier qu'un restaurant a au moins un type de commande.
- PlatsCommande(Cid, Pid, PRestaurant)
  Contrainte induite: Vérifier qu'une commande a au moins un plat.
- AllergenesPlat(Pid, PRestaurant, ANom)

3 Analyse des fonctionnalités

4 Bilan du projet

5 Mode d'emploi

Nous avons développé, comme demandé initialement, un démonstrateur en Java.

Après la panne informatique de l'Ensimag et l'arrêt de l'obligation du démonstrateur, nous avons quand même voulu rendre un démonstrateur fonctionnel, même si toutes les fonctionnalités n'ont pas été introduites. Ainsi, Oracle n'étant pas accessible, nous avons développé la Base de Données en SQLite, en local.

Nous avons alors exporté le démonstrateur dans un `.jar`. Pour démarrer le démonstrateur, il faut aller dans le dossier Interpréteur et lancer la commande `java -jar GrenobleEAT.jar`.

La navigation dans le navigateur est assez intuitive. L'utilisateur peut naviguer de menu en menu en suivant les instructions demandées. Pour sélectionner les options proposées par démonstrateur, l'utilisateur devra introduire soit le chiffre correspondant à l'option souhaitée, soit exactement écrire l'option souhaitée. L'application indiquera quand est-ce qu'il faut introduire l'option en chiffre et quand il faudra le faire en toutes lettres.

Lorsqu'un utilisateur veut réaliser une commande, il peut soit voir la liste des restaurants soit explorer les catégories de l'application.

La liste des restaurants s'affiche dans l'ordre décroissant des évaluations des utilisateurs et dans l'ordre alphabétique des noms.

Pour l'exploration des catégories, tout d'abord l'application recommande à l'utilisateur les 3 dernières catégories commandées. Puis, si les catégories ne conviennent pas à l'utilisateur ou si simplement l'utilisateur manque de commandes réalisées, nous avons choisi de faire un parcours sous forme d'arbre. Au début, l'application propose des catégories générales, et l'utilisateur peut soit sélectionner la catégorie courante, soit continuer à parcourir les sous-catégories. Puis, l'application montre les restaurants spécialisés dans les catégories choisies.

L'utilisateur peut alors choisir les plats qu'il veut dans le restaurant choisi. Une fois qu'il a fini sa sélection, l'utilisateur choisi entre commander en livraison, sur place ou à emporter. L'application demandera alors les informations complémentaires.

Pour s'identifier dans l'application, l'utilisateur doit soit indiquer son email et son mot de passe, soit créer un nouveau compte en introduisant toutes les données nécessaires. Si un utilisateur souhaite demander un effacement de ses données, il est possible de le demander à l'application avec l'option correspondante via la page d'accueil. L'application efface alors toutes les données personnelles, tout en gardant l'identifiant unique de compte (U_id). Ceci permet de garder les évaluations et l'historique des commandes sans que celles-ci soient attribuées à une nouvelle entrée des utilisateurs.

Finalement, l'utilisateur peut aussi évaluer les commandes réalisées dans le passé. Il pourra voir les commandes réalisées et choisir une note de 0 à 5, ainsi que laisser un commentaire optionnel.