

Analyse Projet BDD

Contents

1	Analyse du problème	2
1.1	Hypothèses	2
1.2	Contraintes	3
1.3	Diagramme Entités/Associations	4
2	Modèle Relationnel	5
3	Analyse des fonctionnalités	5
4	Bilan du projet	5
5	Mode d'emploi	5

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 une évaluation

Nous sommes arrivés avec les contraintes suivantes :

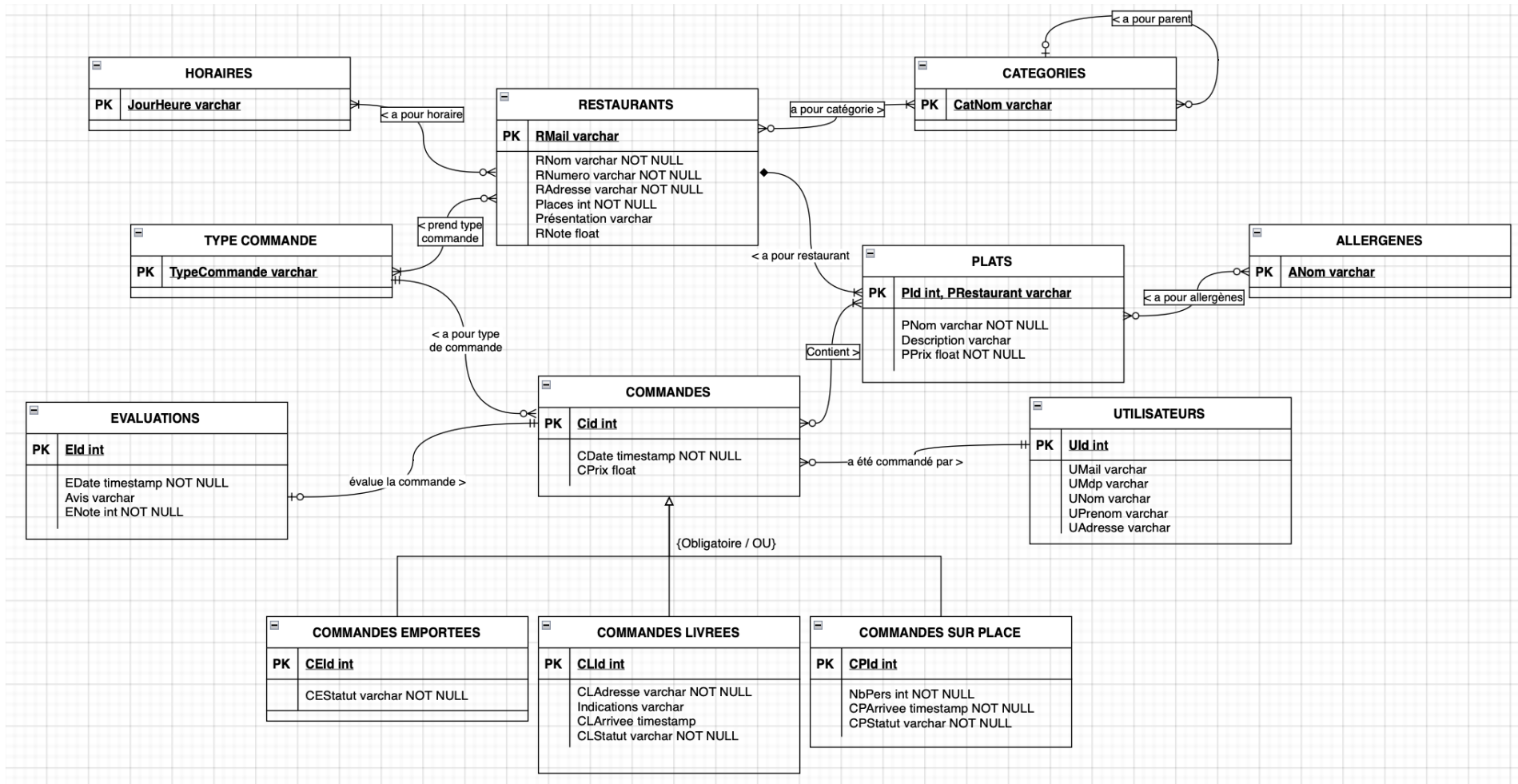
1.2 Contraintes

3

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

1.3 Diagramme Entités/Associations

On obtient ensuite le diagramme Entité-Relation suivant :



2 Modèle Relationnel

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 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 convainquent pas 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 soit rentrer dans sa session en indiquant son email et mot de passe, soit créer un nouveau compte en introduisant toutes les données nécessaires. Si un utilisateur souhaite demander un effacement de ses données, le demander à l'application dans l'option correspondante. Le compte efface alors toutes les données personnelles, mais existe encore dans l'application même s'il reste non accessible. Ceci permet de garder les évaluations et l'historique des commandes sans que celles-ci soient attribuées à une personne.

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.