NGuyen Minh 11/06/2017

Schlegel Antoine

**NF18 : Relationnel**

A noter : les ID de restaurant, commande seront implémenté en SERIAL. En effet, pour le php, ce type va permettre l’auto-incrémentation lorsque l’utilisateur va rentrer une donnée.

Restaurant(#ID: integer, Adresse: varchar, Ville: varchar, pays: varchar, Nom: varchar) avec Adresse, Ville, Pays, Nom NOT NULL

**Héritage par mère pour le groupe (Employé, Cuisinier, Serveur, Manager):**

La classe mère est dans une association, donc on ne peut pas utiliser un héritage par classe fille. Nous avons le choix entre l’héritage par référence et l’héritage par classe mère. Comme l’héritage n’est pas complet, nous avons préféré utiliser un héritage par référence.

Employe(#ID: integer, Nom: varchar, Prenom: varchar, DateNaissance: Date, Anciennete: integer, IDRestaurant =>Restaurant(ID), t:{Cuisinier, Serveur, Manager}, Spécialité: varchar, AccueilRestaurant: Boolean)

*Contrainte:*

Cardinalité 1 côté restaurant

ID NOT NULL

t NOT NULL

If t=Cuisinier, Spécialité NOT NULL;

if t=Serveur, AccueilRestaurant NOT NULL

Commande(#ID: integer, #IDRestaurant=>Restaurant(ID), DateCommande: DateTime)

*Contrainte :*

Cardinalité 1 côté restaurant

IDRestaurant NOT NULL

QuantiteElem (#IDCommande=>Commande(ID), #IDRestaurant=>Commande(IDRestaurant),#IDElement =>Element(IDElement), Quantite : integer)

Carte(#Nom:varchar)

Periode(#IDRestau=>Restaurant(ID), #NomCarte=>Carte(Nom), #DateDebut: Date, DateFin: Date)

*Contrainte:*

Cardinalité 1..\* des 2 cotés

Proj(Restaurant,Adresse, Ville, Pays) ⊆ PROJ (Periode, Adresse, Ville, Restaurant)   
Proj(Carte, Nom) ⊆ PROJ(Periode,NomCarte)

DateFin NOT NULL

DateFin>DateDebut

Les périodes ne se chevauchent pas

Prix(#IDElement => Element(IDelement),#NomCarte=>Carte(Nom), Prix : Numeric)

*Contrainte*:

Cardinalité 1..\* coté Element

PROJ(Element, IDelement) ⊆ PROJ(Prix, IDelement)

**Héritage de Element:**

La classe mère est abstraite, est reliée aux classes Carte et Commande par des associations N:M. L’héritage par les classes filles est proscrit. L’héritage n’est pas complet, il est déconseillé de faire un héritage par la classe mère. Nous avons donc choisi d’utiliser un héritage par référence.

On crée une clé artificiel IDelement.

Element (#IDelement : integer)

Boisson(#ID=>Element(IDelement), NomBoisson : varchar, Volume : varchar) avec (NomBoisson,Volume) clé

Menu(#ID=>Element(IDelement), NomMenu : varchar) avec NomMenu clé

Plat(#ID=>Element(IDelement), NomPlat : varchar, Catégorie : varchar, t :{Plat Principal, Entrée, Dessert}) avec NomPlat clé.

*Contrainte:*

PROJ(Element, IDelement) IN ( PROJ(Menu, ID) UNION PROJ(Plat,ID) UNION PROJ(Boisson,ID))

**Héritage de Plat:**

On a une classe mère abstraite mais reliée à la classe Ingrédients par une association N:M. Plat est aussi relié à la classe Menu par une autre association N:M. L’héritage par les classes filles est donc proscrit. L’héritage est complet, il est donc plus intéressant de faire un héritage par la classe mère.

*Contraintes pour Plat:*

On a un héritage par classe mère, et les classes filles n’ont pas d’attributs

t NOT NULL

Ingredients(#Nom: varchar, Solide: bool)

QuantiteIngredients(#IDPlat=>Plat(ID),#NomIngredients=>Ingredient(nom),Quantite: Float)

*Contrainte:*

Cardinalité 1..\* coté Ingrédients

PROJ(Ingrédients, Nom) ⊆ PROJ(QuantiteIngredients, NomIngredients)

ContenuMenu(#IDMenu=>Element(IDelement), #IDPlat=>Element(IDelement))

*Contrainte*:

Cardinalité 1..\* coté Plat

PROJ(Plat, NomPlat) ⊆ PROJ(ContenuPlat, NomPlat)