PROJET 3 : MODÈLE LOGIQUE DE DONNÉES

Classes :

Client(#id : int, nom : str, telephone : str, adresse : str) avec {telephone key, (telephone, adresse) NOT NULL}

Compte(#date_creation : date, statut : Statut, solde : decimal) avec {statut, solde NOT NULL}

Appartenir(#compte : => Compte.date_creation, #client => Client.id)

CompteCourant(#compte => Compte.date_creation, date_decouvert : date, decouvert_autorise : decimal) avec {decouvert_autorise NOT NULL}

CompteRevolving(#compte => Compte.date_creation, montant_min : decimal, taux_interet_journalier : decimal) avec {(montant_min, taux_interet_journalier) NOT NULL}

CompteEpargne(#compte => Compte.date_creation)

MinMaxMois (#année : int, #mois : Mois, min : int, max : int, #compte => Compte.date_creation) avec {(min, max) NOT NULL}

Opération(#Client=>Client.id, #Compte=>Compte.date_creation, montant : int, #date : date, état : Etat, type_operation : TypeOpération, TypeChèque : DepotEmission)
avec {(Client, montant, état, TypeOpération) NOT NULL}

★ **Justification des héritages :**

Pour la classe Opération, on effectue un héritage par classe mère car on est dans le cas où la classe mère est abstraite et où les classes filles ne sont pas référencées par d'autres classes.

Pour la classe Compte, on effectue un héritage par référence. En effet, en raison des relations qui nécessitent un référencement des classes mères et des classes filles, il est impossible d'envisager un héritage par classe mère ou par classe fille sans perdre d'informations.

Types :

DepotEmission : enum{depot, emission}
TypeOpération : enum{Guichet, Chèque, CB, Virement}
Statut : enum{ouvert, fermé, bloqué}
Mois : enum{'janvier', 'février', 'mars', 'avril', 'mai', 'juin', 'juillet', 'août', 'septembre', 'octobre', 'novembre' et 'décembre'}

Etat : enum{‘traité’, ’non traité’, ’en cours’}

Contraintes :

- ★ **Opération :**

NOT (TypeChèque != NULL AND TypeOpération IN {Guichet, CB , Virement})
NOT (Compte.Statut = fermé)
NOT (Compte.Statut = bloqué AND TypeOpération IN {CB, Chèque})
NOT (montant < 0 AND TypeChèque = {dépôt})
NOT (montant > 0 AND TypeChèque = {émission})
NOT (montant > 0 AND TypeOpération IN {Virement, CB})

- ★ **Comptes :**

Compte épargne :
solde > 300 :
Projection(Jointure(Restriction(Compte, solde >= 300),CompteEpargne,Compte.date_creation = CompteEpargne.compte), Compte.date_creation) = Projection(CompteEpargne, compte)

{seules des opérations au guichet et virements peuvent faits avec un compte épargne}

Compte révolving :
solde < 0
Projection(Jointure(Restriction(Compte, solde < 0),CompteRevolving,Compte.date_creation = CompteRevolving.compte), Compte.date_creation) = Projection(CompteRevolving, compte)

Classe mère abstraite :

Projection(Compte, id) = Union(Projection(CompteEpargne, compte), (Union(Projection(CompteRevolving, compte), Projection(CompteCourant, compte)))

Héritage exclusif :

Intersection(Projection(Comptecourant, compte), Intersection(Projection(CompteRevolving, compte), Projection(CompteEpargne, compte)))={}

Pour la composition :

Projection(CompteCourant, compte) = Projection(MinMaxMois, compte)

- ★ **Appartenir :**

Tous les clients ont au moins un client et un client possède au moins un compte :

Projection(Restriction(Appartenir, compte)) = Projection(Compte, date_creation)

Projection(Restriction(Appartenir, client)) = Projection(Client, id)