POLYTECH SORBONNE

MAIN4

Base de Données

TD 2 : de l'UML au modèle relationnel

(Correction)

Exercice 1 : Vocabulaire

Nous donnons ci dessous les instances de deux relations qui pourraient constituer un fragment d'une base de données bancaire. Indiquez :

- Les attributs de chaque relation

  Correction : *Pour la relation* `COMPTES` *les attributs sont* `num`, `type` *et* `solde`. *Pour la relation* `CLIENTS` *les attributs sont* `nom`, `prenom`, `num-client`. *Pour la relation* `COMPTE-CLIENT`, *les attributs sont* `numClient` *et* `numCompte`.

- Les n-uplets (tuples) de chaque relation

  Correction : *Les n-uplets de chaque relation sont les ensembles des lignes. Pour* `COMPTES` *: { (12345,"courant", 12000), (23456, codevi, 15000), (34567, courant, 5000) }. Idem pour les autres relations.*

- Un domaine adéquat pour chaque attribut

  Correction : `num`, `solde` *et* `numCompte` *peuvent être des entiers (*`Integer`*). Les autres sont des chaines de caractères (*`String`*). Le type du compte a pour domaine {"codevi", "courant", ...}.*

- Le schéma pour chacune des relations

  Correction : `COMPTES(num : Integer, type : String, solde : Integer)`
  `CLIENTS(nom : String, prenom : String, num-client : String)`
  `COMPTE-CLIENT(numClient : String, numCompte : Integer)`

- Le schéma de la base

  Correction : *Le schéma de la base est simplement l'ensemble des trois schémas de relations ci-dessus.*

- Quelles clés primaires pourriez-vous choisir pour ces relations ? Y a-t-il des clés étrangères à prévoir ? Si oui, indiquez-les.

  Correction : *Pour* `COMPTES` `num` *est la clé primaire. Pour* `CLIENTS`, `num-client` *est la clé primaire (ça*

ne peut pas être le couple (`nom,prenom`) à cause des doublons. Pour `COMPTE-CLIENT` la clé primaire est l'ensemble des deux attributs (`numClient`, `numCompte`). Chacun d'eux est de plus une clé étrangère vers les attributs `num-client` de `CLIENTS` et `num` de `COMPTES` respectivement.

COMPTES	num	type	solde
	12345	courant	12000
	23456	codevi	15000
	34567	courant	5000

CLIENTS	nom	prenom	num-client
	Dupont	Marcel	901-222
	Durand	Ginette	805-333
	Durant	Ginette	805-334

COMPTE-CLIENT	num-client	num-compte
	901-222	12345
	805-333	23456
	805-334	34567

Exercice 2

Considérez tous les diagrammes UML du TD1 et transformez chacun de ces diagrammes en un modèle relationnel.

Correction : *Ici, le but est d'appliquer les règles de transformations données dans le cours de manière systématique, "bêtement".*

1. *on commence par créer une relation pour chaque classe.*
2. *les attributs de la classe deviennet les attributs de la relation*
3. *on s'occupe ensuite des associations UML selon leurs cardinalités : * :* donne lieu à une classe intermédiaire (comme `COMPTE-CLIENT` dans l'exo 1), 0 :* ou 1 :* donne rajoute un attribut clé étrangère dans la relation correspondant au *, 1 :1 donne lieu à une fusion des deux relations (avec possiblement renommage). Pour l'héritage, on utilise une clé étrangère pour relier les relations issues des classes filles à la classe mère*
4. *pour la composition UML, on utilise aussi une clé étrangère*
5. *on essaye aussi un maximum d'éviter les redondances.*

Exercice 1, TD 1 `DEPARTEMENT(nom : String, Discipline : String)`
`COURS(id: Integer, numC : Integer, nom : String, Description : String, nom_dep : String)`

Exercice 2, TD 1 `CLIENT(nom : String, nss : String, numPortable : String, id_adresse: Integer);`
`COMPTE(num : String, solde : Real, type : String, nss_client: String);`
`ADRESSE(id : Integer, nom: String, rue: String, codePostal: String, ville: String);`

On créé un identifiant unique pour l'adresse pour ne pas avoir une clé primaire trop complexe.

CLIENT.id_adresse est une clé étrangère faisant référence à ADRESSE.id.
COMPTE.nss_client est une clé étrangère faisant référence à CLIENT.nss.

Exercice 3, TD 1 *Il y a une association 1 :1 entre le client et le portable, on peut donc fusionner la table PORTABLE et la table CLIENT.*

CLIENT(nom : String, nss : String, numPortable : String);
COMPTE(num : String, solde : Real, type : String);

(on devra ensuite préciser en SQL que numPortable doit être UNIQUE).

*La relation COMPTE_CLIENT qui représente l'association 'detient' de cardinalité * : *. Les attributs sont des clés étrangères vers num de COMPTE et nss de CLIENT.*

COMPTE_CLIENT(numCompte : String, nssClient : String);

La relation adresse doit avoir un nouvel attribut, id qui va servir de clé primaire. En effet, si on utilise tous les attributs comme clé, alors dans la table FIXE, lorsqu'on voudra mettre une clé étrangère vers une adresse, il faudra rajouter toutes les colonnes de la table ADRESSE ce qui duplique les données. La relation HABITE dénote l'association UML du même nom.

ADRESSE(id : Integer, num : Integer, rue : String, codePostal : String, ville : String);
HABITE(nssClient:String, idAddr : Integer);
TELEPHONE(num : String)
FIXE(num : String, type : String, idAddr : Integer);

De plus dans FIXE, num est à la fois une clé primaire et une clé étrangère vers le num de TELEPHONE, et idAddr une clé étrangère vers id de ADRESSE. numPortable dans client est une clé étrangère vers TELEPHONE.

Cependant, le bon sens dicte que comme le numéro de portable est passé dans la table CLIENT, conserver deux tables TELEPHONE et FIXE n'a pas d'intérêt, on peut supprimer TELEPHONE. Seul inconvénient de cette approche : rien n'impose maintenant qu'il ne peut pas exister un numéro fixe et un numéro portable qui soient identiques.

Exercice 4, TD1

- *aPourCapitaine est 1 :1, on pourrait donc souhaiter fusionner les deux tables Joueur et Equipe*
- *mais joueDans est * :1, on a donc besoin de garder les deux tables séparées*
- *Par ailleurs, on simplifie en se débarrassant de la classe Personne. Rien n'empêche plus qu'un Joueur et un Fan aient deux identifiants identiques, mais peu importe.*

JOUEUR(id: Integer, nom: String, prenom: String, numPortable: String, nom_equipe: String);
nom_equipe est une clé étrangère vers EQUIPE.nom.
FAN(id: Integer, nom: String, prenom: String);
EQUIPE(nom: String, couleur: String, capitaine_id: String);
capitaine_id est une clé étrangère vers JOUEUR.id.
Attention, rien n'indique ici que le capitaine doit être un joueur de l'équipe.

JOUEUR_IDOLE(joueur_id: Integer, fan_id: Integer);
joueur_id est une clé étrangère vers JOUEUR.id.
fan_id est une clé étrangère vers FAN.id.
EQUIPE_PREFEREE(equipe_nom: String, fan_id: Integer);
equipe_nom est une clé étrangère vers EQUIPE.nom.
fan_id est une clé étrangère vers FAN.id.
