

Projet: A propos de la Gestion des Jeux Olympiques

Duy Minh LE, 12315392

1. Conception UML et Normalisation

Question 1

On rappelle le schéma relationnel fourni:

LesEpreuves (numEp, nomEp, formeEp, categorieEp, nbSportifsEp, dateEp, nomDi)

$\text{/* } <no, n, f, c, nb, da, di> \in \text{LesEpreuves} \iff \text{no est le numéro d'épreuve du nom n, forme (individuelle, par équipe ou par couple), catégorie c (féminin, masculin ou mixte), un nombre de sportifs nb et une date d. L'épreuve fait partie de la discipline di */}$

LesSportifsEQ (numSp, nomSp, prenomSp, pays, categorieSp, dateNaisSp, numEq)

$\text{/* } <no, n, p, pa, d, c, ne> \in \text{LesSportifs} \iff \text{no est le numéro de sportif, avec un nom n, un prénom p, un pays pa une date de naissance d et une catégorie c (feminin ou masculin). Il est inscrit dans l'équipe ne. */}$

On peut en suite en déduire les dépendances fonctionnelles

- **LesEpreuves** (numEp, nomEp, formeEp, categorieEp, nbSportifsEp, dateEp, nomDi)

numEp → nomEp, formeEP, categorieEP, nbSportifsEp, dateEp, nomDi

On voit bien que **nomEp** détermine tous les autres attributs, donc une clé. C'est en fait, la seule clé, on détermine bien que cette relation est de forme **BCNF**.

- **LesSportifsEQ** (numSp, nomSp, prenomSp, pays, categorieSp, dateNaisSp, numEq)

numSp → nomSp, prenomSp, pays, categorieSp, dateNaisSp

numEq → pays

Ici on a **numEq** qui ne détermine que pays, donc via la règle d'augmentation, (**numSp**, **numEq**) est une clé. Vu que tous les attributs sauf numEq sont dépendants à numSp, cette relation est de forme **1NF**.

Ici le numéro d'équipe numEq est dans la clé, bien qu'il y aie des sportifs sans équipe. On va ultimativement attaquer ce problème en transformant le schéma

On transforme donc la relation **LesSportifsEQ** en la coupant par 2:

- **LesSportifs** (numSp, nomSp, prenomSp, pays, categorieSp, dateNaisSp)

numSp → nomSp, prenomSp, pays, categorieSp, dateNaisSp

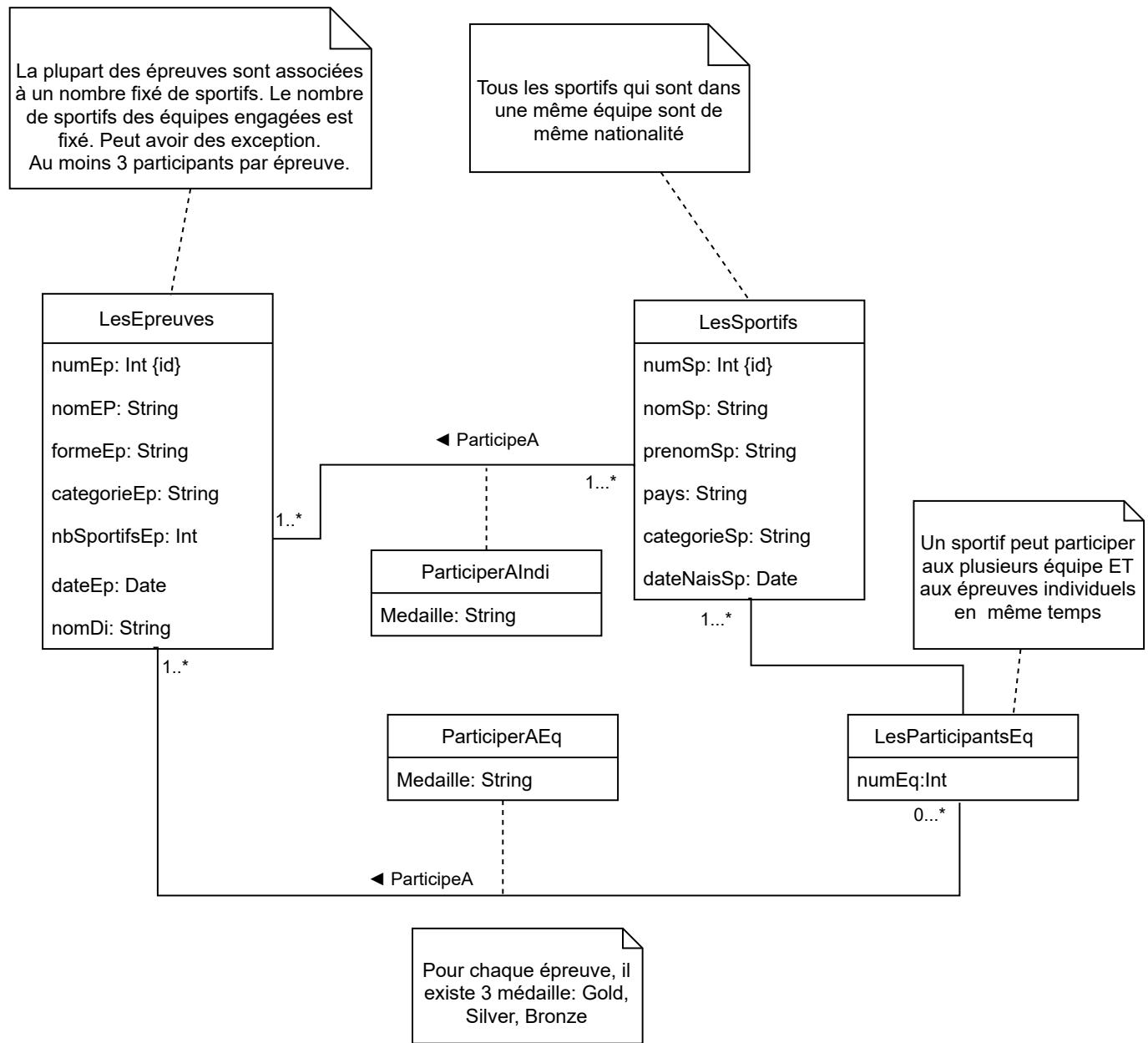
numSp est facilement la clé dans ce cas, et cette relation est de forme **BCNF**.

- **LesParticipants** (numSp, numEQ)

Pas de DF donc **BCNF**.

Question 2

On va maintenant modifier le schéma obtenu à la question précédente pour compléter la BD.



2. Implementation

Question 3

Schema proposé

- **LesEpreuves** (numEp, nomEp, formeEp, categorieEp, nbSportifsEp, dateEp, nomDi)
- **LesSportifs** (numSp, nomSp, prenomSp, pays, categorieSp, dateNaisSp)
- **LesParticipantsEq**(numEq)
- **RepartitionEq**(numSp, numEq)

- **ParticiperAIndi**(numSp, numEp, medaille)
- **ParticiperAEq**(numEq, numEp, medaille)

Domaines

- domaine (dateNaisSp) = date(dateEp) = Date
- domaine (formeEp) = {'individuelle', 'par equipe', 'par couple'}
- domaine (categorieEp) = {'feminin', 'masculin', 'mixte'}
- domaine (categorieSp) = {'feminin', 'masculin'}
- domaine (medaille) = {'Gold', 'Silver', 'Bronze'}
- domaine (nomDi) = domaine (nomEp) = domaine (nomSp) = domaine (prenomSp) = domaine (pays) = chaines de caracteres
- domaine (numSp) = domaine (numEp) = domaine (nbSportifsEp) = entier > 0

Contraintes d'integralite

- RepartitionEq[numSp] ⊆ LesSportifs[numSp]
- RepartitionEq[numEq] ⊆ LesParticipantsEq[numEq]
- ParticiperAIndi[numSp] = LesSportifs[numSP]
- ParticiperAIndi[numEp] ⊆ LesEpreuves[numEp]
- ParticiperAEq[numEq] = LesParticipantsEq[numEq]
- ParticiperAEq[numEp] ⊆ LesEpreuves[numEp]
- ParticiperAEq[numEp] ∩ ParticiperAIndi[numEp] = ∅

Question 4

Etape 2

Je vous laisse regarder le script SQL fourni avec ce rendu.

Commentaire: J'ai aussi ajoute les triggers **ON DELETE** aux certains attributs, j'expliquerai dans la partie des triggers.

Etape 3

De meme, le code python est fourni avec le rendu donc je fais un petit resume des lignes que j'ai modifie/ajoute:

- **main.py**: Ajoute des options 5 et 6 pour reinitialiser les views et les triggers. En fait, dans le script SQL avant de creer un tel view ou trigger, je fais **DROP IF EXISTS**
- **database_functions.py**: Ajoute les fonctions correspondantes aux options 5 et 6 (principalement copier et coller)
- **excel_extractor.py**: change presque tout le fichier sauf la partie des epreuves. En generalment il n'y a rien qui est complique. Pour l'insertion des medailles, en fait, tout d'abord j'insere tous les sportifs et leurs epreuves dans **ParticiperAIndi** et **ParticiperAEq** avec '**null**' comme medaille. Puis je fais **Update** pour rajouter les propres medailles. Ce qui n'est pas tres efficace, je reconnaiss bien, mais l'echelle avec laquelle on travaille n'est pas enorme, donc ca suffit.

Etape 4

Consultez le script `viewDB.sql` fourni.

Pour les triggers, je propose 2 triggers:

- `disqualificationSp`: dès la disqualification d'un sportif, toutes les équipes desquelles il fait partie est aussi disqualifiée. Ceci explique bien pourquoi j'ai mis des `ON DELETE CASCADE` sur les foreign key dans la création des tables: Une fois un sportif est supprimé, on supprime aussi ses inscriptions individuelles, et une fois une équipe est disqualifiée, on supprime ses inscriptions aussi. Ce qui est difficile à régler, est l'amélioration des médailles. Vu qu'on a pas des informations sur les classements exactes des participants, je garde les médailles intactes.
- `disqualificationEp`: encore une fois avec la disqualification. Si une épreuve n'est plus assez de participant, on la supprime.

Utilisation de GIT

Ce rapport et tous les codes sont sur GitHub.