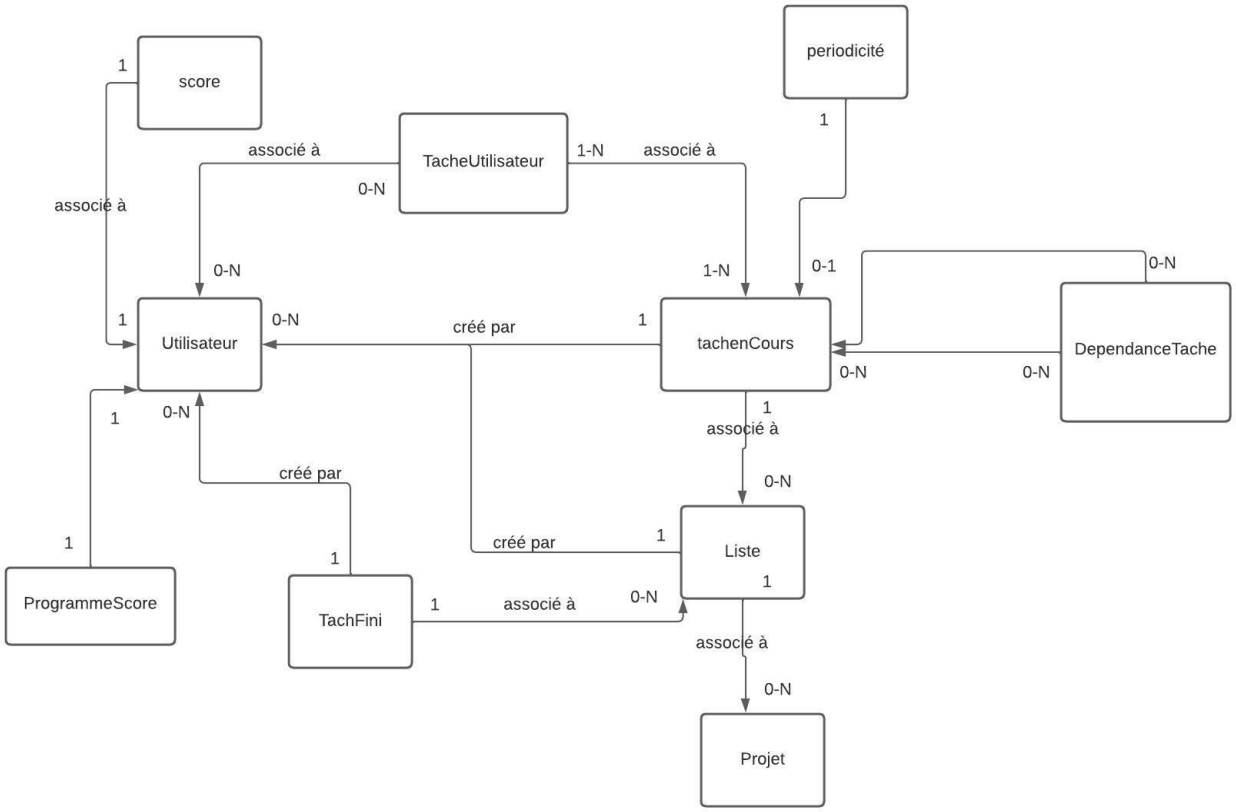
Rapport Todo List project

- Nouveau schema E/S:



- Modèle logique relationnel:

1. Utilisateur(
    idUtilisateur : int PK,
    nom : string,
    prenom : string,
    login : string,
    motDePasse : string,
    dateNaissance : date,
    dateInscription : date
);

2. TACHENCOURS(
    idTache : int PK,
    intitulé : STRING,
    dateEcheance : date,
    lienExterne : STRING,
    categorie : STRING,
    status : int,
    dateAccomplissement : date,
    #idCreateur : int,

```
        #idListe : int
);

3. TACHEFINI(
        idTache : int PK,
        intitulé : STRING,
        dateEcheance : date,
        lienExterne : STRING,
        categorie : STRING,
        status : int,
        dateAccomplissement : date,
        #idCreateur : int,
        #idListe : int
);

4. PERIODICITE(
        #idtache : int PK,
        dateDebut : date,
        dateFin : date
        Periodz : interval
);

5. Projet(
        idProjet : int PK,
        NomProjet : string,
        DescriptionProjet : CLOB
);

6.LISTE(
        idListe : int PK,
        #idProjet : int,
        #idCreateur : int
);

7.PROGRAMMESCORE(
        idProgramme : int PK,
        #idUtilisateur : int,
        scoreToAdd : int,
        scoreToSub : int
);

8. Score(
        #idUtilisateur : int PK,
        score : int,
        niveau : int
)

9. DependanceTache(
        #idTache : int,
        #idTacheDependante : int,
        PRIMARY KEY (idtache,idTacheDependante)
)
```

10. TacheUtilisateur(
#idtache : int,
#idUtilisateur : int,
PRIMARY KEY (idTache,idUtilisateur)
)

- Choix de conception:

Déclencheur :

1- Pour le premier déclencheur on a décidé de le séparer en deux partie :

- trigger déclenché au transfert d'une tâche de la table Tachencours à la table tache fini.
- Trigger déclenché à la mise à jour de l'attribut status d'une tâche dans la table Tachencours.

Où les deux triggers fonctionnent avec un algorithme similaire suivant :

- Si status=0 au transfert; On identifie les utilisateurs associé à la tâche et on déduit de leur score, pour chacun, le nombre précisé dans scoretoSub de leur programme.
- Si status=1 ; On identifie les utilisateurs associé à la tâche et on ajoute à leur score le nombre précisé dans leur programme dans scoretoAdd. On mets à jour la dateCompletion de la table tachencours aussi.

2- Pour le deuxième trigger on a décidé ce qui suit :

Quand le trigger se déclenche, il crée des clones de la tâche référencé par périodicité en utilisant l'intervalle spécifié jusqu'à atteindre la date de fin avec pour chacun un dateEchance Adapté.

Bonus :

- InitUser : un trigger qui fait un double travail :

À la création d'un utilisateur, la première partie crée le login à partir du nom et prénom comme précisé sur le sujet, puis dans la deuxième partie, après la création, crée un enregistrement dans la table score en utilisant l'identifiant de cet utilisateur.

- UpdateLevel : un trigger qui met à jour le niveau des utilisateurs à chaque mise-à-jour sur la table score.

Procédures :

1- nbPoints

Après avoir récupéré les tâches ainsi que les programmes associés dans un curseur, on parcours ce dernier pour fair incrémenter le nombre de points positifs si le status est de 1 (tâche accomplie) ou décrémenter le nombre de points négatifs si la date d'échéance est inférieure à la date du jour et que la tâche reste non accomplie.
NB : une meilleure solution s'est présentée à la fin à l'introduction de la date d'accomplissement de la tâche mais faute de temps, on a décidé de garder la conception initiale.

2- archiveTasks

La procédure transfère les tâches de la table des tâches actives vers la table des tâches à archiver sans aucune modification particulière. La procédure, comme recommandé, est lancée automatiquement chaque semaine et ne cible que les tâches finies ou les tâches non accomplies et dont la date d'échéance dépasse la date du jour.

Travail fait en binome :
Boulhanna Ilyass 22016717
Jbilou Ghait 22014866

Nous deposerons chacun les mêmes fichiers dans nos dépot moodle.