Université De Montpellier - Faculté Des Sciences
Année Universitaire 2020 - 2021
L3 informatique
Projet HLIN511

Rapport de projet :

Tournoi sportif

Création d'une plateforme web pour
organiser des événements sportifs

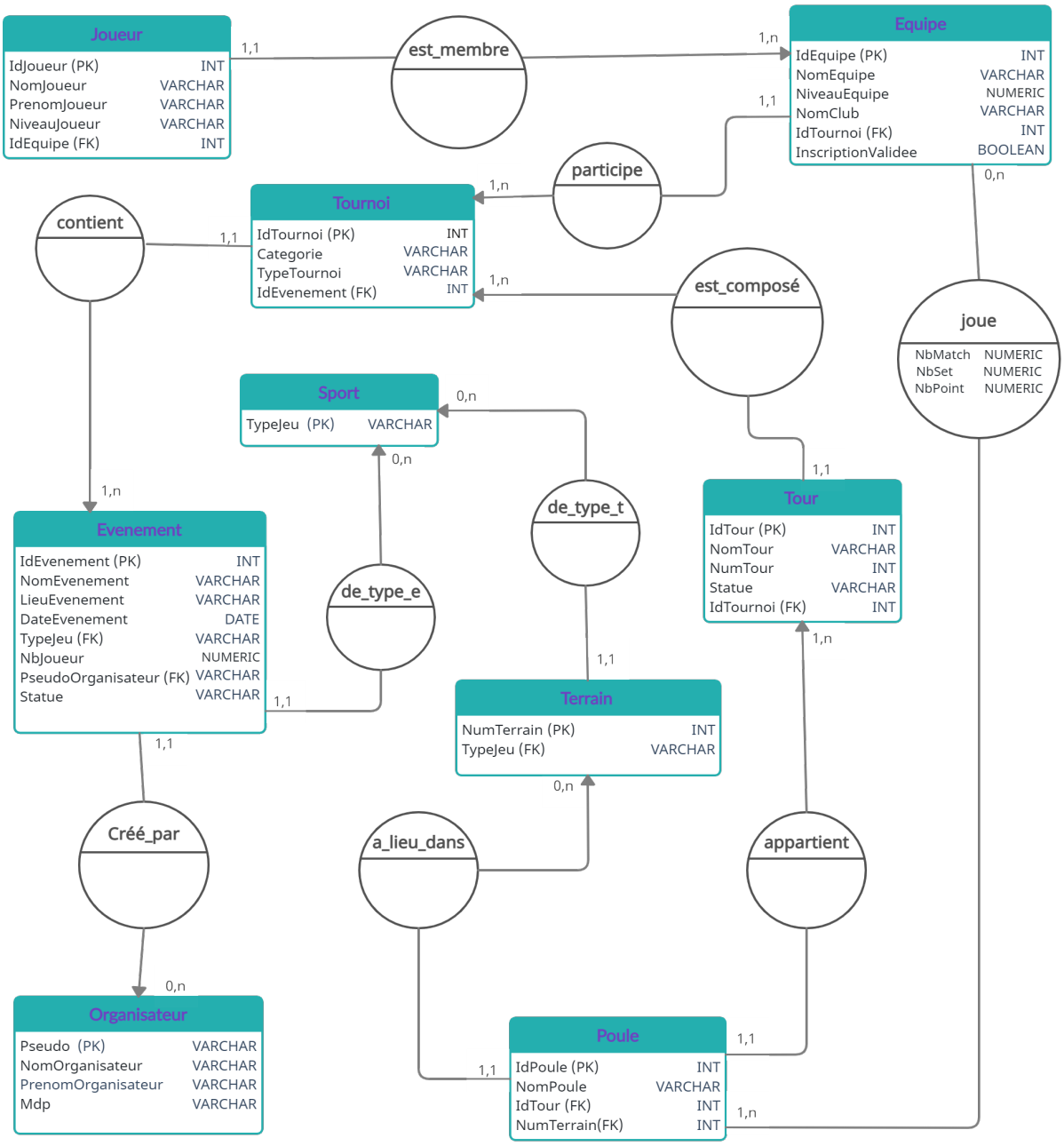
Groupe H :

Marwan MASHRA 21811785
Anh CAO 21713580

Sommaire

Modèle entité-association

Schéma



Explications

Entités

Organisateur : C'est la personne qui a le droit de créer et d'éditer les événements. Il est identifié par son pseudo et possède un nom, un prénom et un mot de passe.

Evenement : Un événement est identifié par un id attribué. Il possède un nom, un lieu, une date, un type de jeu (sport), un nombre de joueur, le pseudo de son créateur et un statut. Le statut indique si l'événement n'a pas encore commencé, avait déjà commencé ou avait déjà terminé.

Tournoi : Un tournoi est identifié par un id attribué. Il possède une catégorie (initial, féminin, jeune...etc) et un type (principal ou consultante).

Equipe : Un équipe est identifié par un id attribué. Il possède un nom, un niveau, un nom de club et un attribut indiquant l'état de l'inscription de l'équipe. Cette entité représente une inscription d'équipe dans un tournoi. Le nom d'équipe est donc unique seulement dans ce tournoi. Deux équipes inscrits dans deux tournoi différents peuvent avoir le même nom.

Joueur : Un joueur est identifié par un id attribué. Il possède un nom, un prénom et un niveau.

Tour : Un tour est identifié par un id attribué. Il possède un nom, un numéro (pour conserver l'ordre) et un statut.

Poule : Une poule est identifiée par un id attribué et possède un nom.

Terrain : Un terrain est identifiée par un numéro attribué. Cette entité aurait pu être remplacée par un attribut directement intégré dans l'entité poule. Toutefois, avoir une entité terrain permet de spécifier le type de sport de chaque terrain ainsi que de chercher tous les terrains disponibles pour un sport choisi.

Sport : Un sport est identifiée par son nom. Cette entité aurait pu être remplacée par un attribut directement intégré dans l'entité événement. Toutefois, avoir une entité sport permet d'accéder directement aux sports qui sont reconnus et donc d'éviter d'avoir des problèmes comme (football, Football, FootBall...etc).

Associations

Créé_par : Une association binaire entre Organisateur et Evenement. Un organisateur peut créer aucun ou plusieurs événements, et un événement est créé par un seul organisateur.

Contient : Une association binaire entre Evenement et Tournoi. Un Evenement contient au moins un tournoi, et un tournoi appartient à un seul événement.

Participe : Une association binaire entre Equipe et Tournoi. Un équipe peut participer dans exactement un tournoi, et un tournoi a au moins un équipe participant.

Est_membre : Une association binaire entre Equipe et Joueur. Un équipe doit avoir au moins un joueur, et un joueur est membre d'un seul équipe.

Est_composé : Une association binaire entre Tournoi et Tour. Un tournoi est composé d'au moins un tour, et un tour appartient à un seul tournoi.

Appartient : Une association binaire entre Tour et Poule. Un tour contient au moins une poule, et une poule appartient à un seul tour.

Joue : Une association binaire entre Equipe et Poule. Un équipe joue dans aucune ou plusieurs poules, et une poule contient au moins un équipe.

A_lieu_dans : Une association binaire entre Poule et Terrain. Une poule a lieu dans un terrain, et un terrain peut être utilisée par aucune ou plusieurs poule mais pas simultanément.

de_type_t : Une association binaire entre Terrain et Sport. Un terrain a un seul type de sport, et un sport peut avoir aucun ou plusieurs terrains.

de_type_e : Une association binaire entre Evenement et Sport. Un événement a un seul type de sport, et un sport peut avoir aucun ou plusieurs événements.

Modèle relationnelle

Organisateur (Pseudo, NomOrganisateur, PrenomOrganisateur, Mdp).

Evenement (IdEvenement, NomEvenement, LieuEvenement, DateEvenement, *TypeJeu*, NbJoueur, *PseudoOrganisateur*,Statue).

Tournoi (IdTournoi, Categorie, TypeTournoi, *IdEvenement*).

Equipe (IdEquipe, NomEquipe, NiveauEquipe, NomClub, *IdTournoi*, InscriptionValidee).

Joueur (IdJoueur, NomJoueur, PrenomJoueur, NiveauJoueur, *IdEquipe*).

Tour (IdTour, NomTour, NumTour, Statue, *IdTournoi*).

Poule (IdPoule, NomPoule, *IdTour*, *NumTerrain*).

Joue (*IdPoule*, *IdEquipe*, NbMatch, NbSet, NbPoint).

Terrain (NumTerrain, *TypeJeu*).

Sport (TypeJeu).

Programmation procédurale

Triggers

Equipe_Inscription_Annulee

Ce trigger est déclenché après chaque modification d'un événement. Si l'événement a déjà commencé, il va supprimer tous les équipes qui n'ont pas encore validé leur inscription.

```
DELIMITER $$
CREATE TRIGGER Equipe_Inscription_Annulee
    AFTER UPDATE ON Evenement
    FOR EACH ROW
BEGIN
    IF NEW.Statue = 'encours' THEN
        delete from Equipe where InscriptionValidee=false and IdTournoi in
            (select IdTournoi from Tournoi where IdEvenement=NEW.IdEvenement);
    END IF;
END $$
DELIMITER ;
```

Liberer_Terrain

Ce trigger est déclenché après chaque modification d'un tour. Si le tour a déjà terminé, il va libérer tous les terrains qui étaient utilisés par les poules de ce tour.

```
DELIMITER $$
CREATE TRIGGER Liberer_Terrain
    AFTER UPDATE ON Tour
    FOR EACH ROW
BEGIN
    IF NEW.Statue = 'termine' THEN
        update Poule set NumTerrain=null where IdTour=NEW.IdTour;
    END IF;
END $$
DELIMITER ;
```

Fonctions

Get_Classement

Cette fonction prend en paramètre un id d'équipe et renvoie son classement dans le tournoi en tenant compte de tous les matchs, les sets et les points qu'il a gagné depuis le début du tournoi. Elle renvoie NULL si les matchs de ce tournoi n'ont pas encore commencé.

```
DELIMITER $$
CREATE FUNCTION Get_Classement (idE INT)
RETURNS NUMERIC(3,0)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE cpt NUMERIC(3,0) DEFAULT 1;
    DECLARE var_idE INT;
    DECLARE IdEquipe_cursor CURSOR FOR SELECT E.IdEquipe FROM Tournoi Tn JOIN Tour Tr
        ON Tr.IdTournoi=Tn.IdTournoi JOIN Poule P ON P.IdTour=Tr.IdTour JOIN Joue J ON
        J.IdPoule=P.IdPoule JOIN Equipe E ON E.IdEquipe=J.IdEquipe WHERE E.IdTournoi=
        (SELECT IdTournoi FROM Equipe E2 WHERE E2.IdEquipe=idE) and E.InscriptionValidee=true
        GROUP BY E.IdEquipe ORDER BY sum(NbMatch) DESC,sum(NbSet) DESC,sum(NbPoint) DESC;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
    OPEN IdEquipe_cursor;
    equipe_loop: LOOP
        FETCH IdEquipe_cursor INTO var_idE;
        IF finished=1 AND cpt=1 THEN
            SET cpt=NULL;
            LEAVE equipe_loop;
        ELSEIF finished=1 OR var_idE=idE THEN
            LEAVE equipe_loop;
        END IF;
        SET cpt=cpt+1;
    END LOOP;
    CLOSE IdEquipe_cursor;
    RETURN (cpt);
END$$
DELIMITER ;
```

Procédures

Remove_Old_Event

Cette procédure supprime les événements qui n'ont toujours pas commencé alors que leur date a déjà dépassé la date d'aujourd'hui.

```
DELIMITER $$
CREATE PROCEDURE Remove_Old_Event ()
MODIFIES SQL DATA
BEGIN
    DELETE FROM Evenement WHERE Statue='bientot' AND DateEvenement<NOW();
END$$
DELIMITER ;
```

Événements

Daily_Remove_Old_Event

Cet événement est programmé pour appeler chaque jour la procédure *Remove_Old_Event*. Cela garantit que tout événement ayant dépassé sa date sans commencer sera automatiquement supprimé de la base de données.

```
DELIMITER $$
CREATE EVENT Daily_Remove_Old_Event
ON SCHEDULE EVERY 1 DAY
COMMENT "Suppression quotidienne des anciens événements qui n'ont pas eu lieu"
DO
BEGIN
    CALL Remove_Old_Event();
END$$
DELIMITER ;
```