# Avant-Propos

Certaines procédures testent des conditions qui sont déjà vérifiées de part les contraintes de la base de données. Cette étape à pour but de fournir des messages d’erreurs personnalisés.

# Tables SQL

## Table Championnat

Une image contenant capture d’écran

Description générée automatiquement

La table Championnat comprend un Id en GUID généré automatiquement ainsi qu’un lastUpdate en datetime qui est implémenté à la création de tout nouveau championnat.

Aucun de ses champs ne peut être nul.

Année est un champ unique en INT, pour éviter les erreurs d’encodage, il est décidé que l’année doit être supérieur ou égale à 1900 ou un rollback transaction est réalisé

lastUpdate est utilisé pour pouvoir faire un suivi, il n’est pas prévu de modifier un championnat

Trigger :

* FormatAnneesChampionnats : vérifie que l’année est supérieure ou égale à 1900 lors d’un insert ou d’un update.

Procédures stockées :

* BackEnd.Championnats\_Add (int @année) : rajoute un championnat à condition qu’il n’y ait pas de championnat avec la même année et que l’année est supérieure ou égale à 1900.

## Table Intersaisons



La table Intersaison comprend un Id en GUID généré automatiquement ainsi qu’un lastUpdate en datetime qui est implémenté à la création de tout nouvelle intersaison

Elle contient également une date de début et une date de fin en datetime et une clé étrangère vers la table Championnats qui est unique

Aucun de ses champs ne peut être nul.

lastUpdate est utilisé pour pouvoir faire un suivi, il n’est pas prévu de modifier une intersaison

Procédures stockées :

* BackEnd.CreationDivisionChampionnat (datetime @DateDebutChampionnat, uniqueidentifier @ChampionatId) : crée 2 quarters et une intersaison à partir d’un championnat et d’une date de début de championnat. Vérifie l’existence du championnat, que la fin du championnat finisse bien dans la même année que le championnat et qu’il n’existe pas déjà de quarters et d’intersaison pour ce championnat.

## Table Quarters



La table Quarters comprend un Id en GUID généré automatiquement ainsi qu’un lastUpdate en datetime qui est implémenté à la création de tout nouveau quarter

Elle contient également une date de début et une date de fin en datetime et une clé étrangère vers la table Championnats qui ne peut être reprise que 2 fois à l’identique.

Aucun de ses champs ne peut être nul.

lastUpdate est utilisé pour pouvoir faire un suivi, il n’est pas prévu de modifier un quarter

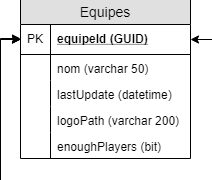
Triggers :

* MaxQuarters : vérifie qu’il n’existe pas déjà 2 quarters pour un championnat.

Procédures stockées :

* BackEnd.CreationDivisionChampionnat (datetime @DateDebutChampionnat, uniqueidentifier @ChampionatId) : crée 2 quarters et une intersaison à partir d’un championnat et d’une date de début de championnat. Vérifie l’existence du championnat, que la fin du championnat finisse bien dans la même année que le championnat et qu’il n’existe pas déjà de quarters et d’intersaison pour ce championnat.

## Table Equipes



La table Equipes comprend un Id en GUID généré automatiquement ainsi qu’un lastUpdate en datetime qui est implémenté à la création de tout nouvelle équipe

Elle contient un nom qui est unique et un suivioPath qui donne l’accès au suivio de l’équipe. Par défaut ce path est noSuivio

Aucun de ses champs ne peut être nul.

lastUpdate est utilisé pour pouvoir faire un suivi, il n’est pas prévu de créer ou de modifier une équipe via nos programmes.

enoughPlayers est utilisé pour savoir si l’équipe a assez de joueurs pour s’inscrire dans un championnat (entre 5 et 10 joueurs). Par défaut la valeur est fausse (0).

## Table EquipesParticipationHistory

Une image contenant capture d’écran

Description générée automatiquement

La table EquipesParticipationHistory comprend les Id en GUID de chaque équipe et du/des championnat(s) auxquels ils ont participé ainsi qu’un lastUpdate en datetime qui est implémenté à la création de toute nouvelle participation.

Une participation est unique par championnat.

Aucun de ses champs ne peut être nul.

lastUpdate est utilisé pour pouvoir faire un suivi, il n’est pas nécessaire de vérifier si on travaille sur des données plus anciennes que celles de la bdd. Une équipe participe ou non à un championnat.

Procédures stockées :

* BackEnd.EquipeParticipation\_Add (uniqueidentifier @equipeId, uniqueidentifier @championnatId) : vérifie que le championnat et l’équipe existe et si la combinaison n’existe pas déjà et sinon crée l’association de participation.

## Table Joueurs

Une image contenant capture d’écran

Description générée automatiquement

La table Joueurs comprend un Id en GUID généré automatiquement ainsi qu’un lastUpdate en datetime qui est implémenté à la création de tout nouveau joueur.

Un joueur possède également un nom et un prénom en varchar (50).

Aucun de ses champs ne peut être nul.

lastUpdate est utilisé pour pouvoir faire un suivi, il n’est pas prévu de créer ou de modifier un joueur

## Table Matchs

Une image contenant capture d’écran

Description générée automatiquement

La table Championnat comprend un Id en GUID généré automatiquement ainsi qu’un lastUpdate en datetime, des points pour les 2 équipes initialisés à 0 en INT et un bit pour savoir si le match a été joué (0 par défaut) qui sont implémentés à la création de tout nouveau match.

Un match possède également une date de rencontre en datetime (qui doit être compris dans un quarter) et 2 clés étrangères qui sont 2 équipes distinctes. De plus la combinaison équipe 1 – équipe 2 doit être unique lors d’un championnat.

Aucun de ses champs ne peut être nul.

lastUpdate est là pour faire un suivi mais également s’assurer que lors de toute modification d’un match des données plus récentes n’existent pas suite à un délai d’encodage.

Triggers :

* Matchs\_WithinChampionnat : vérifie que les matchs créés ou modifiés ont bien une date qui correspond à un quarter existant.
* Matchs\_SameTeams : vérifie que les 2 équipes d’un match ne soient pas les mêmes
* Matchs\_Doublons : vérifie qu’un match n’a pas déjà été prévu entre ses 2 équipes, dans le même sens et dans le même championnat.
* Matchs\_UpdatePlusVieux : vérifie qu’une mise à jour des données n’a pas été réalisée après l’encodage.

Procédures stockées :

* BackEnd.Matchs\_Add (datetime @matchDate, uniqueidentifier @equipe1Id, uniqueidentifier @equipe2Id) : rajoute un match en vérifiant les mêmes conditions que les 3 premiers triggers sur la table.
* BackEnd et MatchManagement.Matchs\_Update (@matchId uniqueidentifier, @matchDate datetime, @equipe1Id uniqueidentifier, @equipe2Id uniqueidentifier, @equipe1Points int, @equipe2Points int,@lastUpdate datetime) mets à jour un match en vérifiant les 4 conditions.

## Table CartonsRougesHistory

Une image contenant capture d’écran

Description générée automatiquement

La table CartonsRougesHistory comprend un Id en GUID généré automatiquement ainsi qu’un lastUpdate en datetime qui est implémenté à la création de tout nouveau carton rouge.

Elle possède 2 clés étrangères : joueurId et matchId. NombreSuspensionRestante et minuteRecu sont des int.

minuteRecu doit être inclus entre 0 et 110 (sans prolongations un match dure 90 minutes)

Aucun de ses champs ne peut être nul.

Un joueur ne peut recevoir qu’un carton rouge par match.

lastUpdate est là pour faire un suivi mais également s’assurer que lors de toute modification d’une carte rouge des données plus récentes n’existent pas suite à un délai d’encodage.

Triggers :

Procédures stockées :

BackEnd et MatchManagement.CartonsRougesHistory\_Add (@joueurId uniqueidentifier, @matchId uniqueidentifier, @nombreSuspensionsRestantes INT, @minuteRecue INT) : ajoute un carton rouge à un joueur à condition qu’il n’en ait pas déjà reçu un dans le même match

# Listes erreurs

|  |  |  |
| --- | --- | --- |
| Numéro | Message | État |
| 50001 | Les années doivent être supérieures ou égales à 1900 | 1 |
| 50002 | Ce championnat existe déjà | 2 |
| 50003 | Ce championnat n’existe pas | 2 |
| 50004 | La date de début du championnat est trop tardive | 3 |
| 50005 | il existe déjà une intersaison pour ce championnat | 2 |
| 50006 | il existe déjà un quarter pour ce championnat | 2 |
| 50007 | Il ne peut avoir que 2 quarters par championnat | 3 |
| 50008 | Cette équipe n’existe pas | 2 |
| 50009 | L'équipe est déjà inscrite à ce championnat | 3 |
| 50010 | La date de match ne tombe pas pendant un quarter existant | 4 |
| 50011 | les 2 équipes inscrites sont les mêmes | 4 |
| 50012 | un match similaire a déjà été prévu pour ce championnat | 4 |
| 50013 | Une mise à jour plus récente a été effectuée | 5 |
| 50014 | Un carton rouge a déjà été donné à ce joueur pendant ce match ! | 4 |

# Schémas

MatchManagement

BackEnd