Joël Hamilcaro, Jie Tu

16 décembre 2018

Rapport de projet

Rio ne répond plus

Table des matières

Introduction

Dans le cadre de ce projet, notre binôme devait mettre au point une base de données concernant les jeux olympiques qui s'étaient déroulés en 2016 à Rio. L'objectif étant ici de mettre en pratique, de manière concrète, les enseignements de BD3, à travers la modélisation et la création d'une base de données. L'étape de modélisation a totalement été réalisée en commun puis nous nous sommes répartis les tâches pour les étapes suivantes.

Modélisation

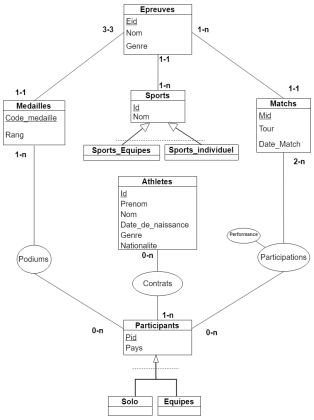
1. – Modèle entité-relation

1.1. – Les sports et épreuves

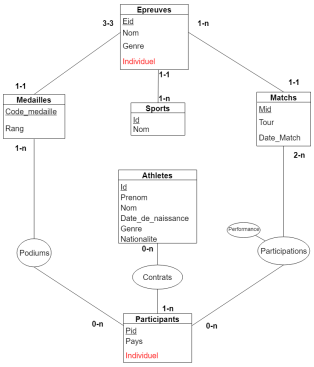
Au regard de ce qui nous avait été demandé, il fallait distingué les sports individuels des sports en équipes. Il nous a donc paru naturel de créer une entité "sports" ayant deux spécialisations : "sports d'équipes" et "sports individuels". Cette spécialisation est totale et non disjointe (par exemple, la natation peut se pratiquer en individuel ou équipe). Les épreuves sont associées à un unique sport, mais un sport peut avoir plusieurs épreuves.

1.2. – Les athlètes, équipes, participant, médailles

Au vue des requêtes demandées, nous avions besoin d'ajouter des attributs genre et date de naissance à l'entité athlètes. De plus, les contraintes liées à la



Modèle ER avec spécialisations
(Voir Figure III.1)



Modèle ER restructuré
(Voir Figure III.2)

gestion des médailles [1] nous ont poussé à ne pas associer directement l'athlète à une médaille. Mais nous associons chaque médaille à ce qu'on appelle un "participant". L'entité participant a pour attributs un pays et un identifiant (PID). Un participant représente soit une équipe soit un participant solo (spécialisations totales et disjointes). L'athlète peut avoir un contrat qui lui donne un PID. C'est sous son PID qu'un athlete participe à une épreuve. Les athlètes qui ont le même PID forment une équipe. C'est toujours le participant (déterminé par un PID) qui gagne une médaille (et non l'athlete directement) Intérêt : Si une équipe gagne, cela correspond à un seul PID (donc une seule médaille pour le pays), mais on peut en déduire les athlètes qui ont gagné cette médaille en regardant les contrats des athlètes. On remarquera que l'entité médaille ne correspond pas à de réelles médailles (en tant qu'objet) mais elle réprésente plutôt à un titre obtenu à une épreuve donnée. Ainsi, les athlètes qui gagnent en équipe sont associé à la même médaille, idem pour les athlètes qui sont à égalité.

1.3. – Matchs, participations

Enfin, les matchs doivent avoir pour attributs une date, l'id de l'épreuve sportive, et le tour correspondant (e.g. "Finale", "Demi-finale"... etc). Cependant, nous n'avons pas mis les gagnants et les scores directement dans matchs. En effet, en prenant en compte toutes les nuances possible, il peut y avoir plusieurs gagnants selon les sports et le système de points diffère selon le sport (meilleur temps, points marqués, ...). Nous avons donc utiliser une association "Participations" qui associe un participant à un matchs. L'association "Participations" a pour attribut une performance qui peut correspondre à un temps, un nombre de points, etc... C'est à partir des "Participations" de chaque matchs qu'on peut en déduire, au cas par cas, les gagnants selon la performance (pour la handball le plus grand nombre de points, pour l'athlétisme le temps le plus court, ... etc). A cette étape, on obtient la Figure III.1.

1.4. – Restructuration

Enfin, nous avons remarqué que les participants solo et équipes (respectivement les sports individuel et en équipes) ne possèdaient pas d'attributs supplémentaires par rapport à l'entité "participants" (respectivement "sports"). Par conséquent, nous avons restructuré notre modèle entités-relation de manière à ce que "individuel" deviennent un attribut de participants (qui sera de type booléen). Concernant l'entité "sports", nous aurions pu remplacer ses spécialisations par un attribut qui pourrait prendre trois valeur ("individuel","en équipe", "les deux"). Cependant, nous avons constaté qu'une épreuve d'un sport est soit individuelle, soit en équipe. De plus chaque épreuve est associée à un (et un seul) sport. Ainsi, nous avons

1. on doit pouvoir savoir si ils ont gagné des médailles individuellement ou en équipes, mais les médailles gagnées en équipe ne doivent compter que pour une seule médaille pour le pays. C'est pourquoi

ajouté un attribut "individuel" (qui ne peut prendre que deux valeurs booléennes) à l'entité épreuve plutôt qu'à l'entité sports. Ce seront donc les épreuves d'un sport qui nous permettront d'en déduire si le sport est individuel, en équipe ou les deux. Nous avons ainsi obtenu la modélisation de la Figure III.2

2. – Schéma relationnel

A partir de notre modèle, nous avons établi le schéma relationnel en suivant les principes vus en cours (reports de clés pour les association 1-n, créations de nouvelles tables avec report de clé pour les association n-n ... etc). Nous avons obtenu le schéma relationnel suivant :

Athletes (id,prenom,nom,date_de_naissance,genre,nationalite)
Participants(pid,pays,individuel)
Sports(sid,nom)
Contrats(id#,pid#)
Contrats[id] ∈ Athlete[id] et Contrats[pid] ∈ Participants[pid]
Epreuves(eid,sid#,nom,genre,individuel)
Epreuves[sid] ∈ Sports[sid]
Matchs(mid,eid#,tour,date_match)
Matchs[eid] ∈ Epreuves[eid]
Participations (mid#, pid# ,performance)
Participations[mid] ∈ Matchs[mid]
Medailles(code_medaille,eid#,rang)
Medailles[eid] ∈ Epreuve[eid] et Medailles[rang] ∈ {'or', 'argent', 'bronze'}
Podiums(code_medaille#,pid#)
Podiums[code_medaille] ∈ Medailles[code_medaille] et Podiums[pid] ∈ Participants[pid]

3. – Adéquation aux formes normales

1. 1NF (Tous les attributs sont atomiques) ✓
2. 2NF (Si la clé primaire est constituée de plusieurs attributs, les attributs non-clé ne dépendent pas que d'une partie de la clé) [2] ✓
3. 3NF (Les dépendances fonctionnelles transitives sont séparés en plusieurs tables). Pour chaque table, les attributs non-clés ne dépendent pas d'autres attributs non-clés. [3] ✓

2. Dans la table Participations, la performance dépend à la fois du matchs et du participant.
3. La date du match n'implique ni le tour ni l'id de l'épreuve car des matchs de tour différents ou d'épreuves différentes peuvent avoir lieu le même jour. Pour les autres tables, c'est évident (la date de naissance n'implique pas le prénom ... etc)

4. FNBC (Les attributs qui font partie de la clé ne doivent pas dépendre d'un attribut non-clé). Notre modèle n'est pas en FNBC par rapport à la table Médailles [4]. ✗

4. – Limites

Notre modèle présente des limites. Tout d'abord, la table participations ne permet pas de vérifier si les participants correspondent bien au genre de l'épreuve. En effet, dans notre modèle il est possible qu'un homme participe à une catégorie d'épreuve pour femme et inversement. Il en est de même pour les équipes qui peuvent participer à des épreuve individuelle. Nous aurions pu rajouter des contraintes de vérification mais cela semblait assez complexe. Par exemple, si on voulait vérifier que les genres des participants corresponde bien au genre de l'épreuve, il fallait :

— Vérifier que les genres des athletes ayant un contrat vers le même pid, soit le même.
— Verifier que le match auquel ces athletes participent corresponde au genre de l'épreuve correspondante.

Ce qui induit des vérifications d'attributs mettant en jeu plusieurs tables (Athletes[genre,id] - Contrats[id,pid] - Participants[pid]- Participations[pid,mid] - Matchs[mid,eid] - Epreuves[eid,genre]). Cependant le fait de pouvoir créer des équipes mixte était bien volontaire car, même si ce cas n'est pas présent dans notre base de données, on s'imagine qu'on puisse avoir des équipes mixtes pour des épreuves mixtes (auquel cas le genre de l'épreuve serait initialisé à NULL). Enfin, nous aurions pu modifier la table Médaille afin qu'elle soit en adéquation avec la forme normale de Boyce-Codd en supprimant l'identifiant des médailles (code_medaille) (chaque médaille serait ainsi identifiée par l'id de l'épreuve correspondante et par le rang de la médaille) Mais cela rendrait les requêtes plus lourdes.

4. on a :
code_medaille $\Longrightarrow$ eid
code_medaille $\Longrightarrow$ rang
rang & eid $\Longrightarrow$ code_medaille

Création et interactions avec la base de données

1. – Remplissage des tables

Pour remplir les tables que nous avions créé grâce à notre schéma relationnel, nous avons récupéré des données du site Lequipe.fr. Nous avons tout d'abord rempli à la main des tableaux au format CSV correspondant au tables sports et epreuves de notre base de données. Pour le reste, nous avons codé un programme en JAVA qui nous a aidé à remplir les tables. Par exemple, nous avons uniquement rempli le nom, le prénom et le genre des athlètes dans un fichier athletes_init.csv. Le programme JAVA a quand à lui généré automatiquement un id et une date de naissance (aléatoire) pour chaque athlète. Suivant le même principe, nous avons rempli un fichier csv contenant les équipes et les contrats (uniquement athlètes - équipes). Le programme JAVA a ensuite rempli le fichier participants.csv (équipes et solos) et contrats.csv (en rajoutant les contrats solos). Une fois que nous avions les PID de chaque participants, nous avons rempli les tableaux matchs et participations à la main. L'intégralité du code JAVA et des fichiers CSV que nous avons remplis sont dans le dossier ("pack"). Nous avons dû faire face à plusieurs difficultés. Une fois les données créées on ne pouvait plus changer les ID des différents participants, athlètes ... etc). Il fallait particulièrement faire attention à la manière dont nous utilisions notre programme pour ne pas rendre les données incohérentes suite à certaines modifications.

Au final, nous avons ajouté toutes les catégories de sports, au moins deux types d'épreuves par sport (une épreuve féminine et une épreuve masculine), voire quatre si le sport est à la fois individuel et en équipe (comme le tennis par exemple). Nous avons ajouté les podiums de chaque épreuve présente dans notre base de données. Concernant les matchs et les participations (temps, points marqués, ... etc), nous nous sommes surtout focalisés sur le remplissage des données qui portaient sur les requêtes demandées (Natation, Handball, Athlétisme). Nous nous sommes repartis les tâches pour remplir les tables de manière indépendante

2. – Requêtes

Nos requêtes sont dans le fichier projet.sql mais nous avons fournis, en annexe, l'ensemble du code et les résultats des requêtes

2.1. – Requêtes demandées

Nous avons effectué toutes les requêtes qui étaient demandées, sauf la requête 1 de difficulté 3. Nous nous sommes reparti la moitié des requêtes entre nous (une personne pour les requêtes paire, l'autre pour les requêtes impaires). Voici les difficultés que nous avons rencontrer, comment nous les avons surmontées, et autres remarques concernant cette partie :

- Pour la requête 1.3 nous ne savions pas s'il fallait considérer l'âge actuel ou l'âge au moment des jeux. Nous avons choisi de considérer l'âge actuel (Mais dans la deuxième requête supplémentaire (celle de notre choix) on utilise l'âge au moment des jeux)
- Nous avons eu des difficulté pour la requêtes 1.6 mais nous avons fini par réussir en s'aidant de la documentation de PosgreSQL (nous avons eu beaucoup de problèmes avant de réussir à caster la chaîne de caratères en type Time)
- Pour la requête 1.6, nous ne pouvions pas vérifier si notre requête était correcte. En effet, aucun sport n'avait plus de médaille que la France en Handball. De fait, le tableau renvoyé était toujours vide. Finalement nous avons pu vérifié en affichant, pour chaque sport, les pays qui ont eu plus de médaille que la France (cette requête est laissée en commentaire dans notre code)
- Nous n'avons pas réussi faire la requête 3.1, cependant, elle devrait, en théorie, être implémentable.

2.2. – Requêtes de notre choix

Les trois requêtes supplémentaires que nous avons choisi d'implémenter sont les suivantes :

1. Le prénom et le nombre d'occurrence des athlètes distincts de la même équipe qui ont le même prénom (en précisant le prenom, l'équipe (pays et sport) pour lequel ils concourrent) dans l'odre décroissant du nombre d'occurrence.
2. L'athlète le plus jeune et le plus agée de chaque épreuve, en précisant l'âge (au jour près) du plus jeune au moment de son premier match et du plus agé au moment de son dernier match.
3. Pour chaque épreuve, afficher le nombre de matchs par épreuves en fonction du jour et de l'épreuve.

Nous avons choisi ces requêtes parce que, même si elles ne sont pas forcément pertinente d'un point de vue sportif, ce sont des questions courantes que n'importe quel utilisateurs peut se poser (le plus jeune et le plus agé de chaque épreuve, les personnes de la même équipe qui ont le même prénom ...). La troisième requête permet quand à elle d'illustrer qu'à partir de notre base de donnée, on peut afficher une ébauche de calendrier) en selectionnant les diverses informations présente dans les tables, nous voulions montrer que c'était possible sans modifier notre base de donnée initiale (mais la requête est syntaxiquement très lourde).

2.3. – Requête impossible

Une requête qui serait impossible (ou très difficile) à écrire dans notre modèle serait une requête qui demande, pour chaque épreuve, le record de la performance (pour une épreuve d'athlétisme on affiche le meilleur temps, pour le handball le plus grand nombre de points, pour le golf le plus petit nombre de points ... etc) Cette requête serait impossible car il faudrait connaître à l'avance le mode de victoire à chaque épreuve sachant que la performance est une chaîne de caractères dans notre table "participations" (il faudrait donc analyser au cas par cas). Une manière de modifier cela serait d'ajouter un attribut "Mode de victoire" à la table épreuve. Ainsi, on pourrait utiliser un CASE en fonction du mode de victoire (le plus petit temps si le mode de victoire c'est "meilleur_temps", le plus haut score si le mode de victoire c'est "le_plus_de_pts" ... etc).

3. – Indice prédictif de Paris 2024

Pour cette partie du projet, nous avons pris nos informations sur les jeux olympiques de paris 2024 sur le site https ://paris2024.org/fr Nous avons créé une table sports2024 dont les attributs ayant des attributs en plus que la table sports : l'attribut "Lieu" et "Nombre". (Nous avons également ajouté le tuple "cérémonie" dans cette table (il ne correspond pas à un sport mais à la cérémonie d'ouverture et de fermeture)) Les évènements de même type seront dans le même lieu (par exemple le Judo et la Lutte) et le nombre correspond au nombre de participants estimé pour ce sport. Ensuite nous avons créé une table "competitions2024" ayant comme premier attribut le sid de chaque sport les 17 autres attributs correspondent à 17 jours compétitions. Il sont de type entier et leur valeur correspond au nombre d'événements pour le sid (sport) considéré.

Pour calculer le nombre de volontaire on utilise le critère suivant. Pour les jeux olympiques de Tokyo 2020, le nombre de volontaire sera d'environ 80000. En principe, il est demandé à chaque volontaire de travailler 5 jours consécutifs et 8h par jour, une fois les jeux lancés. Donc les 80000 volontaire correspondent à un besoin réel de 80000*5 (un volontaire compte pour 5). En reprenant cela pour Paris 2024, si on considère que le nombre d'épreuve est de 400. Par épreuve on prendra

une moyenne de 40 matchs [1], on prendra un nombre moyen de 50 participants par matchs. Ce qui nous donne par participants, un besoin en volontaire de 80000*5/(400*40)/50 = 0.5. Ainsi notre coefficient sera de 0.5 volontaire par participants. C'est ce coefficient qui est utilisé dans la requête qui calcule le nombre de volontaire nécessaire par jour et par lieu.

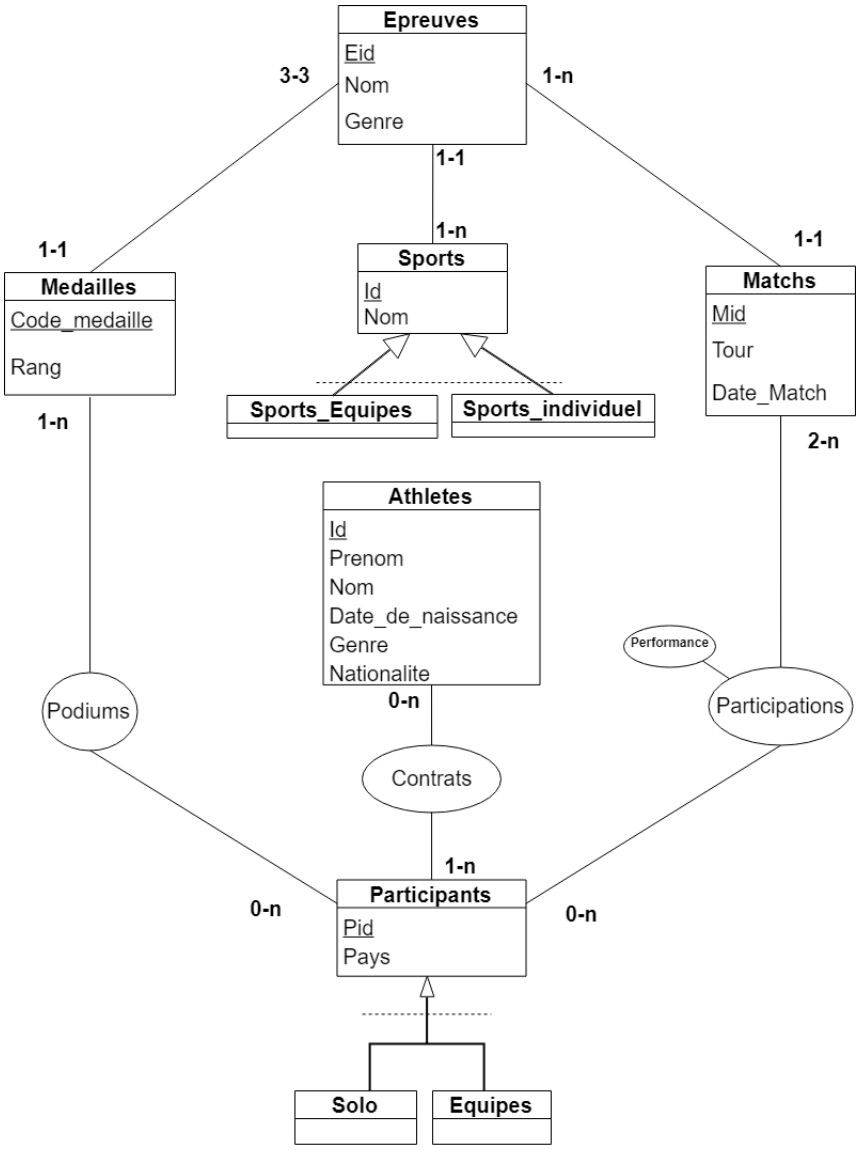


FIGURE III.1 – Modèle entité-relation avant restructuration, laissant apparaître les spécialisations

1. 1 finale + 1 petite-finale + 2 demi-finale + 4 quarts de finale + 8 huitième de finale + 24 matchs éliminatoire

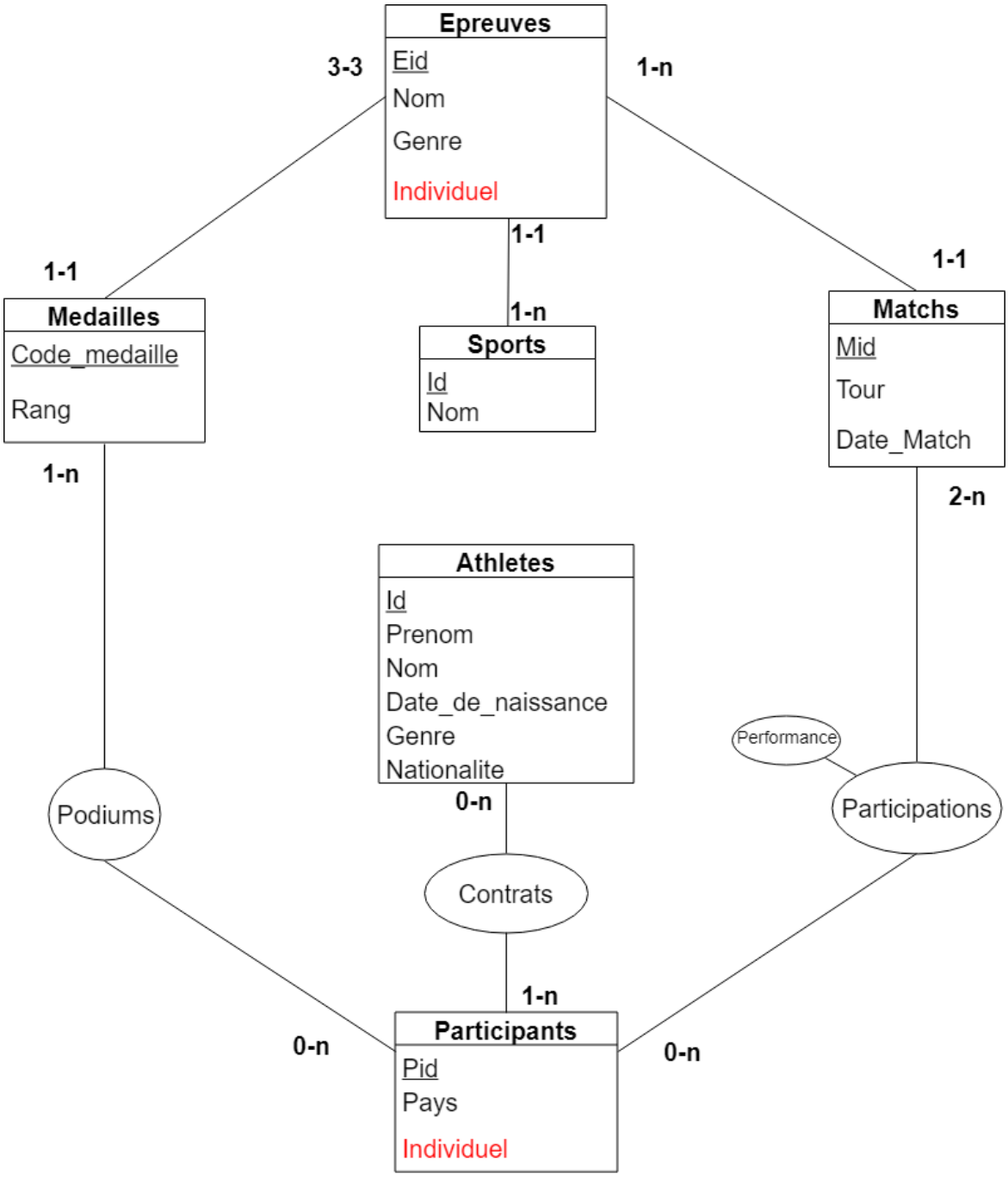


FIGURE III.2 – Modèle Entité-Association restructuré