Bases de données – Le modèle relationnel

I) Historique et objectifs des Systèmes de Gestion de Bases de Données

Une **base de données** est un « conteneur » stockant des données telles que des chiffres, des dates ou des chaînes de caractères, pouvant être retraités par des moyens informatiques pour produire une information ; par exemple, des chiffres et des noms assemblés et triés pour former un annuaire. Les retraitements sont typiquement une combinaison d'opérations de recherches, de choix, de tri, de regroupement, et de concaténation.

Un **Système de Gestion de Base de Données (SGBD en français et DBMS pour DataBase Management System en anglais)** est une suite de programmes qui manipule la structure de la base de données et dirige l'accès aux données qui y sont stockées. Une base de données est composée d'une collection de fichiers ; on y accède par le SGBD qui reçoit des demandes de manipulation du contenu et effectue les opérations nécessaires sur les fichiers. Il cache la complexité des opérations et offre une vue synthétique sur le contenu. Le SGBD permet à plusieurs usagers de manipuler simultanément le contenu, et peut offrir différentes vues sur un même ensemble de données.

La base de données est une notion des années 1960, mais le modèle relationnel date des années 1970 et les SGBD des années 1980. Les bases de données relationnelles sont basées sur la théorie des ensembles avec l'utilisation des opérateurs de l'algèbre relationnel (l'union, la différence, produit cartésien, la projection, la sélection, le quotient et la jointure).

Les objectifs d'un SGBD sont de plusieurs ordres :

a) Indépendance

- Indépendance physique
  L'utilisateur ne doit pas être affecté par un changement de support, de méthode d'accès
  (par exemple le système de fichier) ;
- Indépendance logique
  L'utilisateur ne doit pas être affecté par une modification de l'organisation logique des données
  (ajout ou suppression d'un attribut, changement des liens entre les éléments) ;
- Indépendance de stratégie d'accès
  L'utilisateur ne doit pas être affecté par l'ajout ou la modification de nouveaux chemins d'accès aux données.

b) Sécurité

- Intégrité (non corruption des données, cohérence lors d'une reprise sur panne, respect des contraintes du monde réel modélisé) ;
- Confidentialité (limitation de l'accès selon profil utilisateur).

Le SGBD permet généralement la même chose qu'un système d'exploitation (protection de certaines données en écriture, authentification par mot de passe ou via le système d'authentification de l'OS)

c) Disponibilité

- Gestion de la concurrence des accès
  (chaque utilisateur doit pouvoir accéder aux données sans être gêné par les autres. Un utilisateur en train de lire les données ne doit pas être bloqué par un autre utilisateur qui lit ou écrit simultanément) ;
- Optimisation des accès
  (la demande formulée par l'utilisateur, dans le langage compris par le SGBD, doit être traitée de la façon la plus efficace et certaines données fortement sollicitées du SGBD peuvent être transférées sur des supports plus performants, comme les supports SSD) ;

Comment atteindre ces objectifs ?

- Avec un langage de description de données ;
- Avec un langage d'interrogation et de manipulation des données ;
- Par la définition de contraintes d'intégrité ;

- Par la définition de clause de confidentialité ;
- En prévoyant des mécanismes de reprise sur panne.

Historique des différents types de bases de données :

1) 1950 - Base de données fichier

   Saisie et traitements se font en fonction des fichiers, avec des programmes différents.
   Les accès concurrents sont mal gérés.

2) 1966 - Base de données hiérarchique

   Les données sont stockées dans des fichiers.
   Les relations entre les données sont représentées sous la forme d'un arbre et il y a duplication de certaines données, qui sont donc redondantes.
   La clef permettant d'identifier les données peut devenir importante.
   L'interrogation est fastidieuse et il est difficile de faire évoluer l'organisation des données.
   La base de registre de windows suit ce modèle.

3) 1969 - Base de données réseau

   Basé sur des graphes et développé pour surmonter certaines limitations du modèle hiérarchique.

4) 1970 - Base de données relationnelle

   Basées sur les travaux d'Edgar F. Codd, chercheur chez IBM.
   Elles utilisent un modèle mathématique simple.
   En théorie : des ensembles sont manipulés par des opérateurs de sélection, de complément, de jointure…
   En pratique : manipulation de tables, contenant des données typées, sans redondance.
   Le SGBDR, Système de Gestion de Base de Données Relationnel, est un SGBD manipulant des données structurées suivant le modèle relationnel.
   Utilisation du langage SQL (Structured Query Language), qui est un langage d'interrogation des données relationnelles.

5) 1985 - Base de données objet

   Les données sont représentées sous forme d'objets (une personne possédant des attributs, tels nom, âge, etc., est enregistrée telle quelle, sans transformation. Les BDOO ne se sont pas imposées en remplaçantes des bases de données relationnelles, mais les SGBDR ont intégré des extensions objet).

6) 2000 - Base de données XML

   Basées sur le langage XML (ex : Tamino, Sedna). Elles permettent de stocker les données au format XML sans transformation, contrairement à ce qui se fait dans les SGBD relationnelles. N'ont pas la puissance de calcul des SGBDR, ni leur capacité de stockage.

7) 2009 - Base de données NoSQL (Not Only SQL)

   Nouvelle méthode de gestion des données, basée sur le principe « Not Only SQL ».
   La plupart des bases de données NoSQL sont constituées par des ensembles distribués de couples clefs-valeurs.
   Il existe de nombreux projets, souvent utilisés dans un contexte Web, et associés au format JSON (JavaScript Object Notation), comme MongoDB, CouchDB.

A l'exception des bases de données XML et NoSQL, la plupart des autres types de bases de données ont été éclipsés par les SGBDR, qui sont actuellement les plus utilisés.

II) Le modèle relationnel

Un modèle de données est une représentation des concepts que l'on souhaite étudier.
Il doit permettre d'énoncer des propriétés des données en termes logiques afin de réaliser des actions réelles complexes qui pourront se traduire par des opérations élémentaires sur les données.

Modéliser, c'est définir un monde abstrait qui coïncide avec une partie de l'apparence du réel.
Bien modéliser, c'est faire que ce monde abstrait soit structuré, performant, et accessible facilement.

Le modèle relationnel est actuellement le plus utilisé et il a été défini en 1970 par l'informaticien américain Edgar F. Codd.

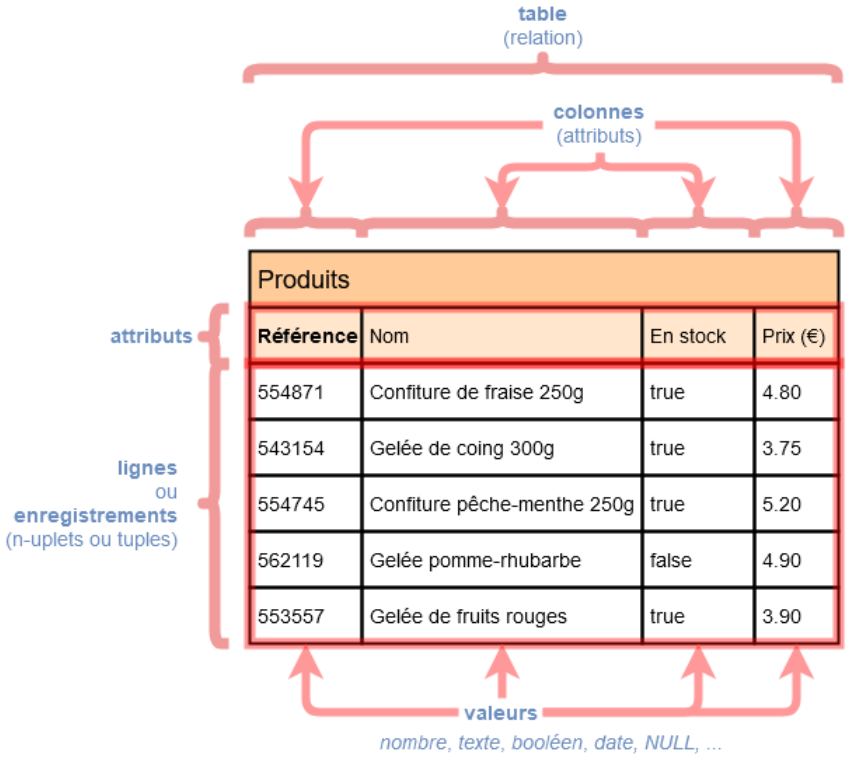
Attention

Il faut bien distinguer la notion de modèle de données de la notion de structure de données :

- Le modèle de données précise quelles entités réelles avec quelles caractéristiques on souhaite manipuler dans le modèle, ainsi que les relations des entités entre elles.
- Une structure de données précise la façon dont les données manipulées seront organisées en machine (un tableau, une liste chaînée, un arbre binaire, etc.).

Le **modèle de données relationnel** représente une base de données comme un ensemble de **relations** (**tables** à deux dimensions) qui contiennent :

- des **attributs**, qui peuvent servir de **clefs**.
  Un attribut est un identifiant (un nom) décrivant une information stockée dans une base. Un attribut correspond à une **colonne** si la relation à laquelle il appartient est représentée sous forme de table ;
- un ensemble de **n-uplets** ou **tuples** (les **lignes** – ou **enregistrements** – de la table) dont les **entrées** (des **valeurs** qui peuvent être de type numérique, chaîne de caractère, booléen, NULL, etc.) appartiennent à un **domaine** ;



Produits

Référence	Nom	En stock	Prix (€)
554871	Confiture de fraise 250g	true	4.80
543154	Gelée de coing 300g	true	3.75
554745	Confiture pêche-menthe 250g	true	5.20
562119	Gelée pomme-rhubarbe	false	4.90
553557	Gelée de fruits rouges	true	3.90

Dans le modèle relationnel, le n-uplet de valeurs scalaires (suites de chiffres ou de lettres, par opposition aux objets structurés et types complexes) représente un objet modélisé (on parle **d'entité**).

On appelle **ordre** (ou **degré**) le nombre d'attributs d'une relation. Dans l'exemple ci-dessus, la relation Produits est d'ordre 4.

Dans une même base de données, différentes relations peuvent avoir des attributs de même nom. Pour désigner l'attribut *attr1* d'une relation *rel1*, on le notera alors *rel1.attr1* (par exemple *Produits.Référence*).

Chaque relation se conforme à un **schéma**. Ce dernier est une description qui indique pour chaque composante des n-uplets de la relation, c'est-à-dire pour chaque attribut, leur nom et leur domaine (par exemple l'attribut Nom est de type String).

La modélisation relationnelle des données et les contraintes d'intégrité

Principes généraux :

La modélisation des données se décompose en plusieurs étapes :

1. déterminer les entités (objets, actions, personnes, etc.) que l'on souhaite manipuler ;
2. modéliser les ensembles d'entités comme des relations en donnant leur schéma, avec en particulier le bon domaine pour chaque attribut ;
3. définir les contraintes de la base de données, c'est-à-dire l'ensemble des propriétés logiques que les données doivent vérifier à tout moment.

1) Les contraintes d'intégrité

Une contrainte d'intégrité est une propriété logique préservée à tout instant par la base de données et qui garantit la cohérence des données. Une contrainte d'intégrité est un invariant de la base de données.

On distingue quatre grandes catégories de contraintes d'intégrité.

a) Contrainte de domaine

Le **domaine** d'un attribut est un ensemble, fini ou infini, de **valeurs admissibles**.
Il est renseigné au moment de la création de la relation. Le SGBD s'assure par la suite qu'un élément ajouté à une relation respecte bien le domaine de l'attribut correspondant.
Sans considération pour les aspects techniques propres au SGBD, le domaine d'un attribut pourra être de l'un des types génériques suivants :

- **String** pour une chaîne de caractères
- **Float** ou **Int** pour un nombre
- **Date** pour une date
- **DateTime** pour une date avec l'heure

Il est parfois difficile de décider comment représenter une entité réelle par un attribut. Cela dépend de la finalité du modèle. Ce sera le cas par exemple pour une propriété complexe comme l'adresse postale, qui sera séparée en plusieurs attributs (par exemple Adresse, CodePostal et Commune).

A faire :

- Définir le domaine de l'attribut CodePostal ;
- Définir le domaine des attributs de la relation ci-dessus.

L'utilisation seule du domaine ne permet pas de garantir l'absence de données incohérentes dans la base.
Par exemple, si on considère l'âge d'une personne, il faudra vérifier en amont la validité de la valeur affectée à l'attribut.

b) Contrainte d'entité

La contrainte d'entité permet de s'assurer que chaque élément d'une relation est unique. Il est en effet nécessaire d'**identifier une entité de façon non ambigüe**.
Considérons par exemple une relation *Usager* où l'on aurait stocké uniquement le nom et le prénom d'une personne. Il est possible que deux personnes aient le même nom et prénom. Ces deux attributs ne permettent donc pas d'identifier un usager de façon certaine.

Pour éviter cette situation, il faut s'assurer, lors de la modélisation, que pour chaque entité d'une relation, un attribut où un ensemble d'attribut permet d'identifier cette identité de manière unique. On appelle un tel attribut la **clé primaire de la relation**.

A faire :

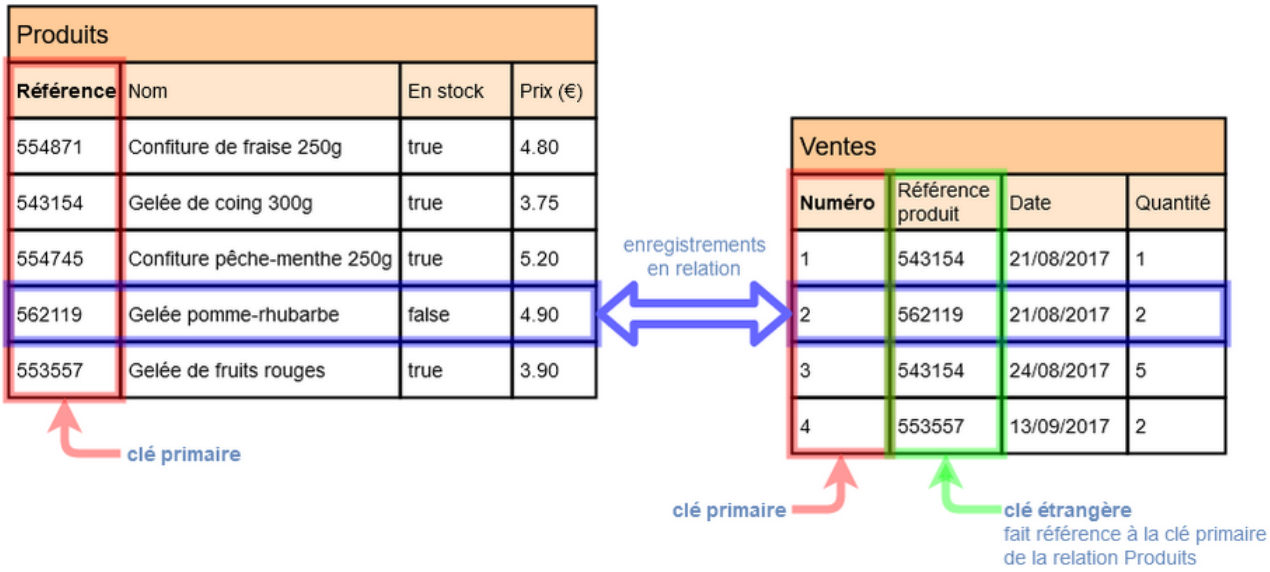
Identifier la clé primaire de la relation *Produits* ci-dessus.

Dans certains cas, la clé primaire d'une relation peut être composée de plusieurs attributs.
Pour une relation donnée, une **clé** est un groupe d'attributs permettant d'identifier une unique entité de la relation.
Pour qu'un groupe d'attributs forme une **clé**, il faut être sûr que deux entités n'auront jamais des valeurs identiques pour ces attributs : on parle d'***unicité*** de la clé.
On parle de **clé candidate** si le groupe d'attributs de la clé est ***minimal***, c'est-à-dire que si on retire un seul des attributs de ce groupe, l'unicité n'est plus vérifiée.
Toute relation doit avoir au moins une clé candidate (et peut en avoir plusieurs).

Le choix d'une clé primaire est important et le SGBD garantira son unicité.

c) Contrainte de référence

Les clés primaires ne permettent pas seulement de distinguer les entités d'une relation de manière unique. Elles sont aussi utilisées pour servir de référence à l'entité dans une autre relation.
Les relations d'une base de données seront ainsi reliées par certaines des valeurs qu'elles contiennent.



Dans cet exemple, l'attribut *Référence produit* de la relation *Ventes* est une **clé étrangère**.
Cela signifie que la valeur de cet attribut dans la relation *Ventes* doit être l'une des valeurs existantes pour cet attribut dans la relation *Produits*. Cette contrainte permet de garantir que la relation *Ventes* ne mentionne que des produits connus de la base de données :

- Elle permet d'éviter de rajouter à la relation *Ventes* des références fictives, inconnues dans la relation Produits ;
- Elle permet aussi d'éviter de retirer une entité de la relation *Produits* si cette dernière apparaît dans une entité de la relation *Ventes*.

Le SGBD se chargera d'effectuer ces vérifications lors des opérations sur la base de données.

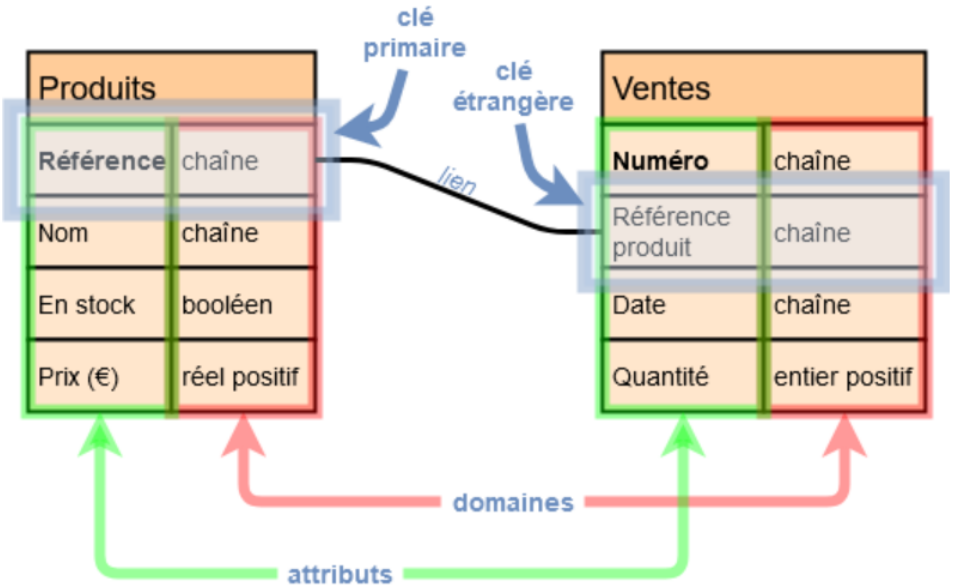
Clé primaire : c'est l'une des clés candidates de la relation, choisie pour être utilisée comme clé étrangère dans une autre relation ;
Clé étrangère : attribut d'une relation dont les valeurs sont des références à une clé primaire d'une autre relation.

d) Contraintes utilisateurs

Les contraintes utilisateurs, parfois appelées contraintes métier, sont toutes les contraintes d'une relation qu'on ne peut exprimer par les trois précédentes. Elles restreignent encore plus les valeurs d'un ou plusieurs attributs d'une relation et sont guidées par la nature des données que l'on souhaite stocker dans la base.
On citera par exemple l'âge d'une personne, le format d'une adresse mail, etc.

2) Le schéma relationnel

Les tables constituent la structure logique du modèle.
Un **schéma de relation** précise le nom de la relation ainsi que la liste des attributs avec leurs domaines.
On peut représenter un schéma de relation sous une forme graphique, par exemple à l'aide d'un diagramme :



On peut aussi le noter plus simplement sous la forme du nom de la relation suivi de la liste de ses attributs.

- **Produits** (Référence, Nom, En stock, Prix)
- **Ventes** (Numéro, #Référence produit, Date, Quantité)

Dans cette notation, les **clés primaires** apparaissent soulignées, les *clés étrangères* précédées (ou suivies) d'un #.
On peut également préciser les *domaines* des différents attributs :

- **Produits** (Référence : TEXT, Nom : TEXT, En stock : BOOL, Prix : FLOAT)
- **Ventes** (Numéro : INT, #Référence produit : TEXT, Date : DATE, Quantité : INT)

Modèle Conceptuel de Données (MCD) et cardinalité

La conception de la structure d'une base de données, si elle est un peu complexe à appréhender, peut nécessiter en amont l'utilisation d'outils de modélisation conceptuels entités-associations (**Modèle Conceptuel des Données** ou **MCD** de la méthode MERISE, diagramme de classes du langage UML).

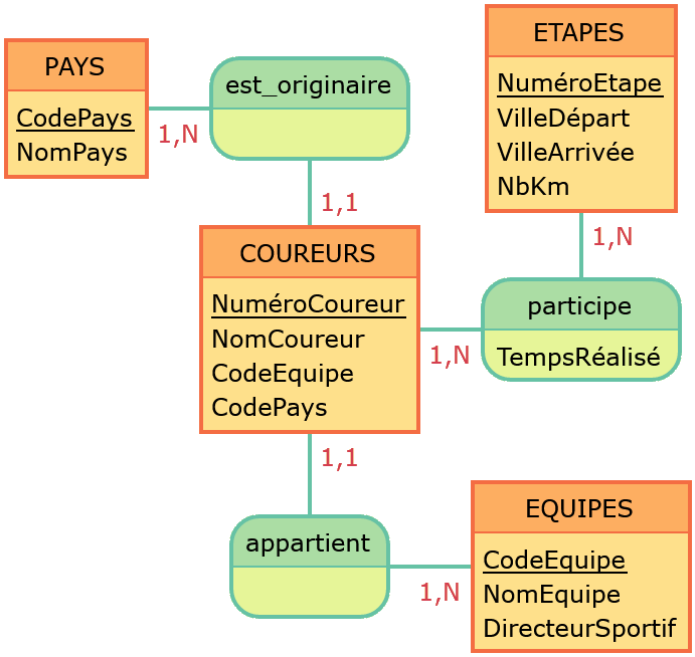
La méthode Merise est une méthode d'analyse, de conception et de réalisation de systèmes d'informations. Elle permet de cadrer un projet informatique en facilitant notamment les échanges entre utilisateurs finaux et informaticiens. Créée dans les années 70 sur commande de l'État français et destinée aux gros projets informatiques de l'époque, la méthode a perduré jusqu'à aujourd'hui. Son utilisation très répandue en Europe.

Dans la méthodologie Merise destinée à créer des bases de données, il y a des outils dédiés aux traitements et aux données. Le **MCD** (**Modèle Conceptuel des Données**) est un des outils majeurs concernant les données. Il s'agit d'une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux à l'aide de diagrammes codifiés.

Eléments d'un diagramme du **Modèle Conceptuel des Données (MCD)** :

- Les entités (1 rectangle = 1 objet) ;
- Les propriétés (la liste des données de l'entité) ;
- Les relations qui expliquent et précisent comment les entités sont reliées entre elles (les ovales avec leurs « pattes » qui se rattachent aux entités). Sous le nom de la relation, on indique la liste des données éventuelles « portées » par la relation ;
- Les cardinalités (les petits chiffres associés aux « pattes »).

Exemple de modélisation MCD des étapes du Tour de France cycliste :



La cardinalité d'un lien entre une entité et une relation précise le minimum et le maximum de fois qu'un individu de l'entité peut être concerné par la relation. Dans le diagramme MCD, on précise les cardinalités minimale et maximale, séparés par une virgule :

- Cardinalité maximale N (vs 1) : signifie une cardinalité positive sans limite supérieure.
- Cardinalité minimale de 1 (vs 0) : les individus de l'entité ont besoin de l'association pour exister.

Une cardinalité se lit de l'entité vers l'association : combien de fois un élément de l'entité peut être concerné par l'association ?
Dans la modélisation ci-dessus, un coureur appartient à une équipe et une seule. On a la cardinalité 1,1 pour le lien entre l'entité COUREURS et la relation *appartient*. A une équipe, appartiennent au minimum 1 coureur et au maximum N coureurs (limite non fixée). La cardinalité du lien entre l'entité EQUIPES et la relation *appartient* est notée $1,N$.

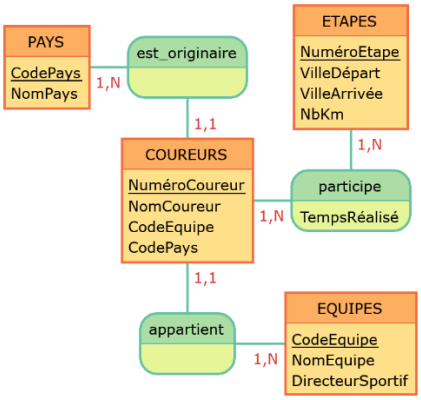
Dans la méthodologie Merise, le **MPD** (**Modèle Physique des Données**) fait suite au MCD. Il s'agit de préparer l'implémentation dans un SGBDR. Cette étape permet de **construire la structure finale de la base de données** avec les différents liens entre les éléments qui la composent. Lors de cette étape, on change aussi de vocabulaire :

- Les **entités** se transforment en **tables** ;
- Les propriétés se transforment en champs (ou attributs) ;
- Les propriétés se trouvant au milieu d'une relation génèrent une nouvelle table ou glissent vers la table adéquate en fonction des cardinalités de la relation ;
- Les identifiants se transforment en clés et se retrouvent soulignés. Chaque table dispose d'au minimum 1 clé dite primaire ;
- Les relations et les cardinalités se transforment en champs parfois soulignés : il s'agit de créer des « clés étrangères » reliées à une « clé primaire » dans une autre table.

Vient ensuite le **MLD** ou **Modèle Logique des Données**, qui est simplement la représentation textuelle du MPD. Le MLD correspondant au MCD des étapes du Tour de France réalisé ci-avant sera ainsi le suivant :

EQUIPES(CodeEquipe, NomEquipe, DirecteurSportif)
PAYS(CodePays, NomPays)
COUREURS(NuméroCoureur, NomCoureur, #CodeEquipe, #CodePays)
ETAPES(NuméroEtape, VilleDépart, VilleArrivée, NbKm)
participe(#NuméroCoureur, #NuméroEtape, TempsRéalisé)

La relation *participe* du MCD, qui contient comme donnée TempsRéalisé a été transformée en table lors du passage du MCD au MPD.



Activité 1

Soit une base de données dont le schéma est le suivant :

Employe(NumEmp, Nom, Prénom, Adresse, Téléphone, Qualification)
Service(NomService, #Responsable, Téléphone)
Projet (NomProjet, DateDeb, DateFin, #NumEmp)

Réécrire ce schéma sous forme de diagramme.
Un employé peut-il avoir plusieurs qualifications ?
Un employé peut-il faire plusieurs projets en même temps ?
Une personne peut-elle être responsable de plusieurs services ?
Un service peut-il avoir plusieurs responsables ?

Activité 2 :

Identifier une clé primaire candidate possible dans le schéma relationnel Usager(NOM,PRENOM,TEL).

Activité 3 :

Soit une base de données dont le schéma est le suivant :

Pilote (PLNUM, PLNOM, PLPRENOM, VILLE, SALAIRE)
Avion (AVNUM, AVNOM, CAPACITE, LOCALISATION)
Vol (VOLNUM, #PLNUM, #AVNUM, VILLEDEP, VILLEARR, HEUREDEP, HEUREARR)

Réécrire ce schéma sous forme de diagramme.

Activité 4 :

Proposer une modélisation relationnelle d'un bulletin scolaire. Celle-ci doit permettre :

- d'identifier un élève, à l'aide d'un numéro étudiant alphanumérique unique ;
- d'identifier une matière, à l'aide d'un identifiant unique ;
- d'attribuer une note à un élève dans une matière.

Activité 5 :

Proposer une modélisation relationnelle pour la situation suivante.
Vous êtes cinéphile et vous souhaitez construire une base de données contenant des informations sur vos films préférés : année de sortie, titre, genre, nom du réalisateur, etc. Vous voulez également associer à cette base des informations sur les acteurs principaux.

Activité 6 :

On considère deux relations *R(a : INT, b : INT, c : INT)* et *S(#a : INT, e : INT)* où l'attribut a de S est une clé étrangère faisant référence à a de R.
Dire si les affirmations suivantes sont vraies ou fausses, en justifiant :

1. Les a de R sont tous deux à deux distincts

2. Les b de R sont tous deux à deux distincts
3. Les a de S sont tous deux à deux distincts
4. Les e de R sont tous deux à deux distincts
5. S peut être vide alors que R est non vide
6. R peut être vide alors que S est non vide

Activité 7 :

Modéliser des informations sur les départements français.
Pour chaque département, on doit pouvoir stocker son nom, son code, sa préfecture ainsi que la liste de tous les départements voisins. Proposer une contrainte utilisateur permettant d'éviter la redondance d'informations dans la liste des voisins.

Activité 8 :

Proposer une modélisation d'un réseau de bus.
La modélisation doit permettre de générer, pour chaque arrêt de bus, une fiche horaire avec tous les horaires de passage de toutes les lignes de bus qui desservent l'arrêt. Dans un premier temps, on cherchera les informations pertinentes pour la modélisation.