Concepteur Développeur en Informatique

Assurer la persistance des données

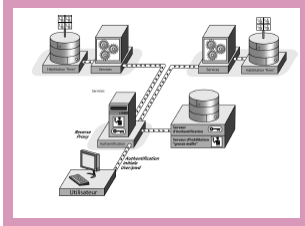
Passage du modèle conceptuel au modèle logique

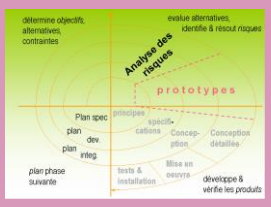
Accueil

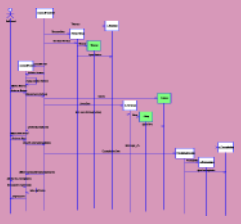
Apprentissage

PAE

Evaluation







Localisation : U02-E01-S03

Sommaire

1 Introduction

Nous allons étudier dans ce document comment effectuer la transformation du Modèle Conceptuel des Données au Modèle Logique des Données.

Le modèle logique sera la représentation du modèle physique qui sera implémenté sur le Système de Gestion de Bases de Données retenu. La modélisation logique ne prend pas en considération les spécificités du langage de définition de données relatif à l'outil informatique.

Toutefois, le modèle logique tient compte du type de SGBD. Il en existe deux types :

- Le modèle logique de données relationnel. Défini par CODD en 1970. De très loin le plus utilisé aujourd'hui compte tenu de la domination des SGBD relationnels.
- Le modèle logique de données navigationnel qui suit les recommandations CODASYL. Ces normes sont décrites sous la norme COBOL CODASYL.

Nous nous intéresserons uniquement au modèle logique de données relationnel. Les bases de données conformes à Codasyl sont aujourd'hui peu usitées en dehors de l'utilisation sur certains systèmes Mainframe. Il est à noter que la plupart aient migrés vers le relationnel.

2 Transformation d'un individu ou entité type

Toute entité type devient une table. Ses propriétés deviennent des attributs de la table (colonne). L'identifiant devient la clé primaire unique de la table.*

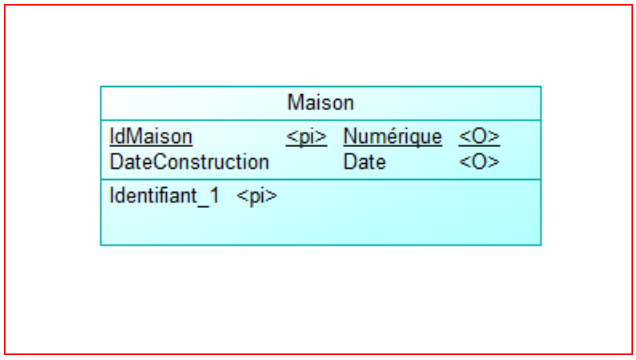
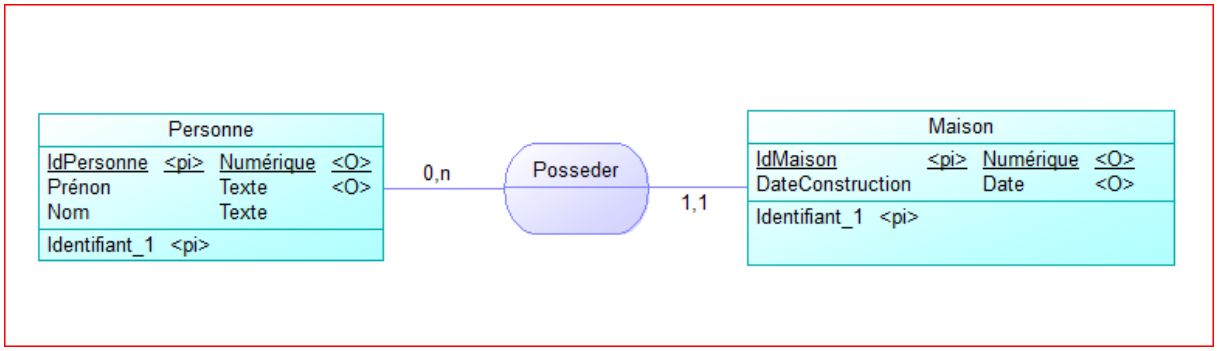


Schéma Relationnel :

MAISON(IdMaison,DateConstruction,Surface)

3 Transformation d'une relation binaire (0,n)-(1,1) ou (1,n)-(1,1)

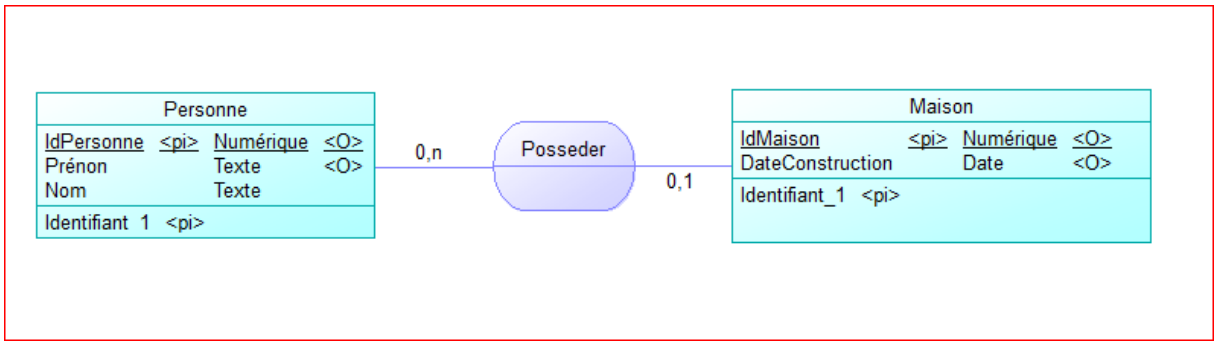


On duplique la clé issue de l'identifiant de l'individu porteur de la cardinalité (0,n) ou (1,n) dans la table issue de l'individu à cardinalité (1,1).Celle-ci devient alors une clé étrangère.

PERSONNE(IdPersonne,Prénom,Nom)

MAISON(IdMaison,DateConstruction,IdPersonne) avec Maison.IdPersonne doit être référencé comme Personne.IdPersonne (Clé étrangère)

4 Transformation d'une relation binaire (0,n)-(0,1) ou (1,n)-(0,1)



Deux solutions :

- Création d'une table avec comme clé primaire l'identifiant de l'individu porteur de la cardinalité (0,1). Définition de contraintes référentielles sur les deux attributs identifiants issus des entités associées.

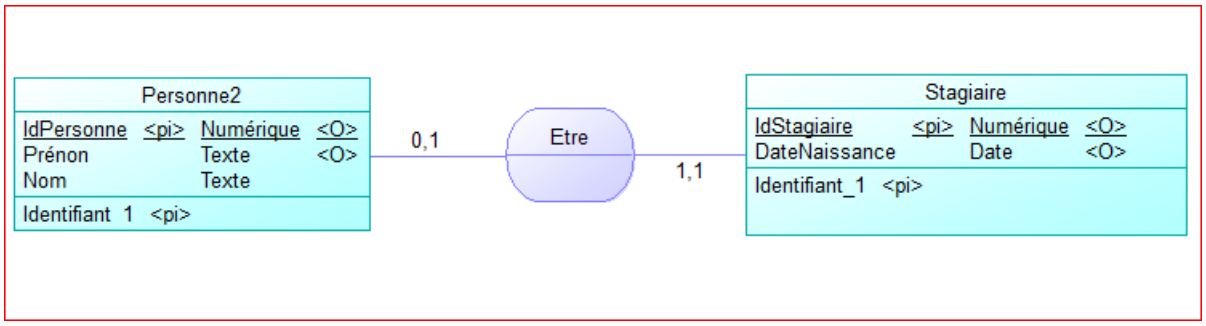
POSSEDER(IdMaison,IdPersonne) avec Posseder.IdMaison doit être référencé comme Maison. IdMaison (Clé étrangère) et avec Posseder.IdPersonne doit être référencé comme Personne.IdPersonne (Clé étrangère)

- Duplication de l'identifiant de l'individu porteur de la cardinalité (*,n) dans la table issue de l'individu porteur de la cardinalité (0,1). Il devient alors une clé étrangère. Il faut pour cette deuxième solution que le SGBD cible accepte des valeurs nulles pour la clé étrangère.

MAISON(IdMaison,DateConstruction,IdPersonne) avec Maison.IdPersonne doit être référencé comme Personne.IdPersonne (Clé étrangère)

5 Transformation d'une relation binaire (0,1)-(1,1)

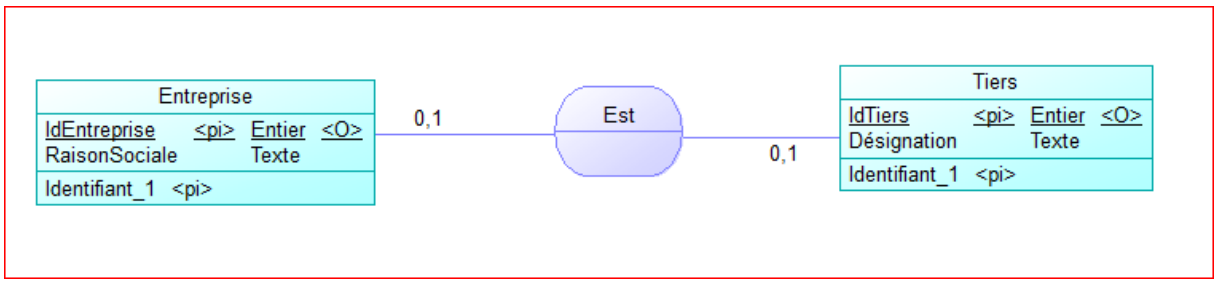
La clé issue de l'identifiant de l'entité porteuse de la cardinalité (0,1) est dupliquée dans la table issue de l'entité porteuse de la cardinalité (1,1) et est précisée clé étrangère.



STAGIAIRE(IdStagiaire,DateNaissance,IdPersonne) avec Stagiaire.IDPersonne est référencé comme Personne.IdPersonne (clé étrangère)

6 Transformation d'une relation binaire (0,1)-(0,1)

C'est une particularité des cas précédemment traités. 2 types de solutions sont possibles qui peuvent donner lieu à quatre solutions au final.



1er type de solution :

Créer une table avec comme attributs les identifiants des individus en relation et comme clé primaire l'identifiant d'une des deux tables.

Les deux attributs sont des clés étrangères.

Exemple de schémas relationnels :

Table ENTREPRISE(**IDEntreprise**, RaisonSociale,…)

Table TIERS (**IdTiers,**,…)

Table EST(**Idtiers**, IDEntreprise) ou table CORRESPOND(IdTiers, **IDEntreprise**)

2ème type de solution :

Duplication de la clé d'une des deux tables issues des entités types dans l'autre table.

Table ENTREPRISE (**IDEntreprise**,RaisonSociale,…,IDtiers)

Table TIERS (**IdTiers**,…)

Ou

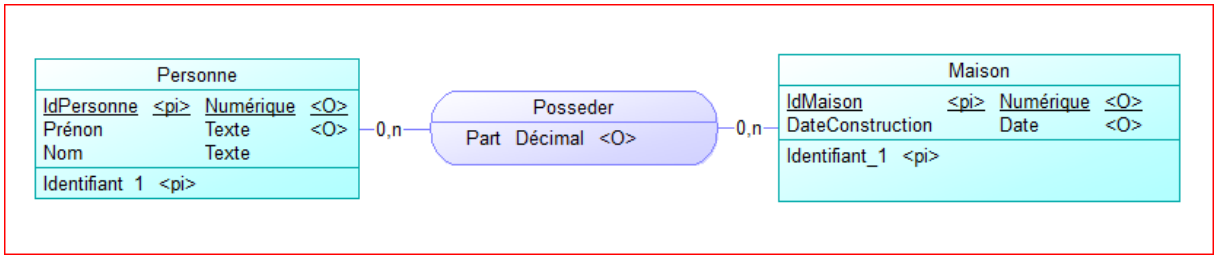
Table ENTREPRISE (**IDEntreprise**,RaisonSociale,…)

Table TIERS (**IdTiers**,…,IDEntreprise)

La première solution est préférable car plus rigoureuse.

7 Transformation d'une relation binaire (*,n)-(*,n)

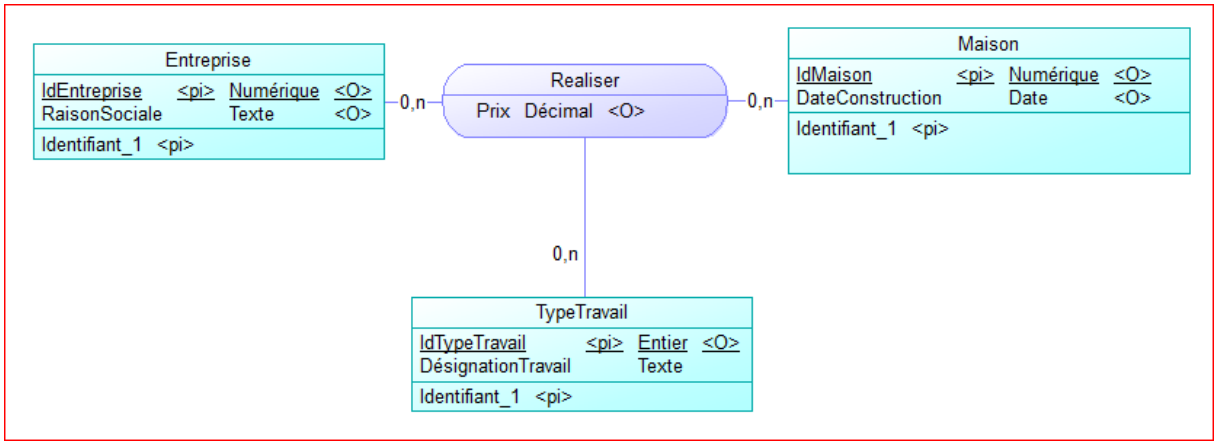
La solution consiste à créer une table dont la clé sera une clé composée des identifiants des deux entités en relation. Les propriétés éventuelles de la relation seront des attributs de la table.



La table POSSEDER aura comme schéma relationnel POSSEDER (**Nom**, **IDMaison**, Part)

8 Transformation d'une relation d'arité supérieure à 2 quelles que soient les cardinalités

La relation donne lieu à la création d'une table qui aura comme clé primaire une clé composée des identifiants des individus sur lesquels porte la relation.



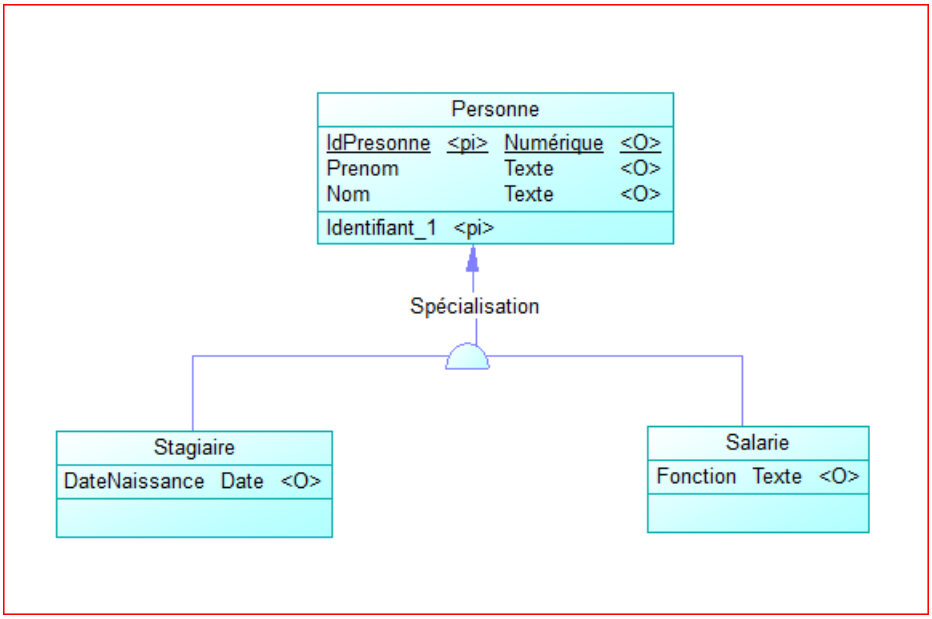
La table REALISER aura comme schéma relationnel :

REALISER (<u>IdEntreprise, IDMaison, IdTypeTravail</u>, Prix)

9 Transformation des sous-types et sur-types

Le schéma relationnel issu de la modélisation des sous-types diffère en fonction des relations qui mobilisent les sous-types et sur-types et les contraintes existantes entre ses derniers.

Soit le modèle conceptuel suivant :



9.1 Une seule table dérivée du sur-type

Lorsque les spécialisations, ici Stagiaire et Salarié, ne sont pas porteuses d'associations spécifiques. Dans ce contexte, les associations porteront sur l'entité Personne.

Nous pouvons alors définir le schéma relationnel de la table personne ainsi :

PERSONNE(IdPersonne,Prenom,Nom,Categorie,DateNaissance,Fonction)

Nous avons introduit une nouvelle colonne Categorie dont le but est de préciser à quelle spécialisation appartient l'occurrence.

Les attributs DateNaissance et Fonction peuvent ne pas être porteurs de valeurs (NULL).

Chaque type spécialisé sera implémenté sous la forme d'une vue. Nous aurons donc une vue Stagiaire et une vue Salarie.

STAGIAIRE(IdPersonne,Prenom,Nom,DateNaissance) avec PERSONNE.Categorie = « Stagiaire ».

SALARIE(IdPersonne,Prenom,Nom,Fonction) avec PERSONNE.Categorie = « Salarie ».

Le nombre de catégorie varie ensuite en fonction des contraintes entre les individus sous-types.

DISJONCTION + COUVERTURE = PARTITION (+)

Si nous avons une contrainte de partition qui régit les sous-types, à savoir disjonction et couverture qui traduit qu'une personne est soit un salarié soit un stagiaire nous aurons :

Catégorie = {Salarie,Stagiaire} et les contraintes suivantes :

Salarie => Fonction != NULL et DateNaissance = NULL implémentée par Trigger.

Stagiaire =>Fonction =NULL et DateNaissance != NULL implémentée par Trigger.

NON DISJONCTION + COUVERTURE = TOTALITE (T)

Une personne peut être Salarie et/ou stagiaire.

Catégorie = {Salarie,Stagiaire,SalarieStagiaire}

Nous aurons alors les contraintes d'intégrité précédentes plus une contrainte pour prendre en considération la catégorie SalarieStagiaire :

SalarieStagiaire => Fonction != NULL AND et DateNaissance != NULL implémentée par Trigger.

DISJONCTION + NON COUVERTURE = EXCLUSION (X)

Nous aurons Catégorie = {Salarie,Stagiaire,Autre} et la contrainte supplémentaire suivante pour les personnes qui ne sont ni Stagiaire ni Salarie :

Autre => Fonction = NULL AND et DateNaissance = NULL implémentée par Trigger.

NON DISJONCTION + NON COUVERTURE

Nous aurons alors Catégorie = = {Salarie,Stagiaire,SalarieStagiaire,Autre} et les contraintes définies précédemment.

9.2 Une table par sous-type

Lorsque le Sur-type est précisé pour uniquement mettre en facteur des propriétés communes dans la démarche de généralisation.

Le sur-type n'est pas mobilisé dans les associations.

Nous sommes nécessairement ici dans le cas d'une partition. Une personne est soit stagiaire soit salarié.

Une table par type de spécialisation dont les schémas respectifs sont :

STAGIAIRE(IdPersonne,Prenom,Nom,DateNaissance

SALARIE(IdPersonne,Prenom,Nom,Fonction)

Contraintes d'intégrité : l'intersection entre stagiaire et salarié doit être un ensemble vide. Vérification par Trigger :

Salarie NOT IN Stagiaire et Stagiaire NOT IN Salarie.

Nous pouvons créer une vue pour connaître l'ensemble des personnes :

PERSONNES(Salaries[ID,Nom,Prenom] Union Stagiaires[ID,Nom,Prenom])

9.3 Une table par entité

Lorsque le sur-type comme les sous-types sont concrets et porteurs d'associations spécifiques, nous allons créer une table pour le sur-type et une table pour chaque sous-type.

Les données communes sont mises en facteur.

Des contraintes d'intégrité référentielles sont définies entre les tables issues des entités sous-type et l'entité sur-type. Une table pour l'entité sur-type, une table pour chaque sous-type ayant pour identifiant celui du sur-type.

Nous aurons les schémas relationnels suivants :

PERSONNE(**IdPersonne**, Prenom,Nom)

STAGIAIRE(IdStagiaire,DateNaissance) avec IdStagiaire doit être référencé comme Personne.IdPersonne (Clé étrangère)

SALARIE(IdSalarie,Prenom,Nom,Fonction) avec IdSalarie doit être référencé comme Personne.IdPersonne (Clé étrangère)

Ensuite, si vous avez exclusion de participation entre les deux typés spécialisés, vous devrez mettre en œuvre les contraintes suivantes :

Salarie NOT IN Stagiaire et Stagiaire NOT IN Salarie.