Collège Lionel-Groulx

Département d'informatique

420 –KED-LG, conception de bases de données

Introduction, Conception du modèle E/R (oracle) et Normalisation

Table des matières

Bref historique

Les données représentent une ressource organisationnelle et informationnelle cruciale que l'entreprise se doit de maîtriser. La majorité des organisations ne saurait réussir sans connaître les données exactes de leur entreprise et de leur environnement externe Pendant très longtemps, les systèmes d'information des entreprises structuraient leurs données sous forme de fichiers, chaque application utilisait donc son propre fichier. Beaucoup de problèmes étaient connus de cette façon de gérer les données d'une entreprise : la redondance d'information, la dépendance des données des traitements, et le manque d'intégration de données en font partie.

Devant le volume d'information grandissant et la complexité des traitements, le stockage des données dans de simples fichiers n'est plus la solution pour le stockage du patrimoine informationnel de l'entreprise. Un moyen de palier aux problèmes de la gestion des fichiers est le SGBD. (Systèmes de Gestion de Bases de Données).

Vers la fin des années 60 et le début des années 70, sont apparus les premiers SGBDs hiérarchiques et réseaux. Vers le milieu des années 70, nous avons vu naître les SGBDs relationnels, qui utilisent le modèle relationnel de Edgar Frank « Ted » Codd pour la modélisation des données. Aujourd'hui les SGBD relationnels sont présents dans la majorité des entreprises.

Concepts de base

1. Définition d'une base de données

Une base de données est un ensemble de données modélisant les objets d'une partie du monde réel et se servant de support à une application informatique.

Un SGBD (système de gestion de base de données) peut-être perçu comme un ensemble de logiciels système permettant aux utilisateurs d'insérer, de modifier et de rechercher efficacement des données spécifiques dans une grande masse d'information partagée par de multiples utilisateurs.

2. Objectifs des SGBD. :

- Non redondance des données : permet de réduire le risque d'incohérence lors des mises à jour, de réduire les mises à jour et les saisies.
- Partage des données : ce qui permet de partager les données d'une base de données entre différentes applications et différents usagers.
- Cohérence des données : ce qui permet d'assurer que les règles auxquelles sont soumises les données sont contrôlées surtout lors de la modification des données.
- Sécurité des données : ce qui permet de contrôler les accès non autorisés ou mal intentionnés. Il existe des mécanismes adéquat pour autoriser, contrôler ou d'enlever des droits à n'importe quel usager à tout ensemble de données de la base de données.
- Indépendance physique : assurer l'indépendance des structures de stockage au structure des données du monde réel.
- Indépendance logique : possibilité de modifier un schéma externe sans modifier le conceptuel. Cette indépendance assure à chaque groupe les données comme il le souhaite à travers son schéma externe appelé encore une VUE.

3. Types de SGBD

Relationnel

Réseau

Hiérarchique

4. Modélisation des données

Un modèle de données est un ensemble de concepts utilisés pour décrire la structure d'une base de données Par structure de base de données, nous entendons, les types de données, les relations, et les contraintes qui définissent le gabarit de la base de données. Tout modèle peut être exprimé à divers niveaux de précision, conceptuel, logique physique et externe. Cette définition sera retenue lorsque nous présenterons la partie « Présentation des approches traditionnelles pour l'intégration des données »

Les techniques de modélisation de données permettent d'une part de comprendre le fonctionnement d'un système te la façon dont les données sont organisées et d'autres part de construire un système d'information qui reflète la réalité.
Il existe plusieurs niveaux de représentation (ou de modélisation des données).

4.1. Niveau conceptuel : le niveau conceptuel ou le schéma conceptuel de la base de données décrit la réalité du système d'information d'une organisation indépendamment du SGBD d'implantation. Le modèle utilisé à ce niveau peut-être le modèle d'Entité/Association (modèle d'Entité/Relation). Ce niveau est de la responsabilité de l'analyste concepteur ou de l'administrateur de la base de données

4.2. Niveau interne (logique et physique) : le niveau interne spécifie comment les données sont représentées ou stockées sur les supports de stockage. Ce niveau est complètement pris en charge par les SGBD. (SGBDR Oracle)

4.3. Niveau externe :ce niveau correspond à la vision de tout ou une partie du schéma conceptuel par un groupe d'utilisateurs concernés par une application ou une partie de l'application. Il s'agit de décrire à l'aide de schémas externes (vues) la façon dont seront perçues les données par les programmes.

5. Le modèle Entité/Relation (E/R): Le formalisme MERISE

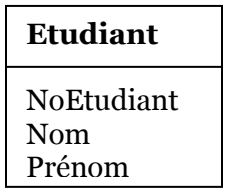
Ce modèle appelé encore le MCD (Modèle Conceptuel de Données) permet de représenter les données d'un système sans se préoccuper du SGBDR (Système de Gestion de Base de Données **Relationnel)** qui sera utilisé ultérieurement. C'est le modèle qui se rapproche le plus du formalisme (modèle) d'oracle

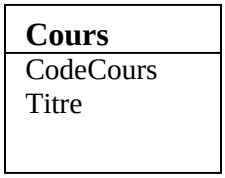
5.1. Rôle du modèle E/A

- Représente le niveau conceptuel d'une base de données
- Permet d'obtenir les caractéristiques logiques d'une base de données
- Utilise 3 (trois) concepts fondamentaux pour modéliser les données : *Entité. Relation et Attributs.*

5.2. Concepts du modèle E/A : définitions

Entité : Collection d'objets de même type concrets ou abstraits ayant une existence propre dans le système. Une entité doit avoir plus d'une occurrence et elle doit être décrite par au moins un attribut, sinon il faudra se questionner sur son existence.
Exemple : Etudiant peut être une entité dans le système de la gestion des étudiants. En effet un étudiant est représenté par plusieurs attributs comme le nom et le prénom. Et nous avons plusieurs occurrences de Etudiant. Deux occurrences de Etudiant sont Lefebvre Martin, Yacoub Saliha
Personnages est également une entité le jeu vidéo Warcraft
Dans le formalisme de MERISE, une entité est représentée par un rectangle.





Occurrence d'un attribut : l'occurrence d'un attribut représente la valeur de cet attribut. Une occurrence pour l'attribut Prénom peut être Yanick

Occurrence d'une entité : une occurrence d'une entité est une instance de cette entité.

Exemple
Soit l'entité Etudiant définie précédemment. Les attributs de l'entité sont :

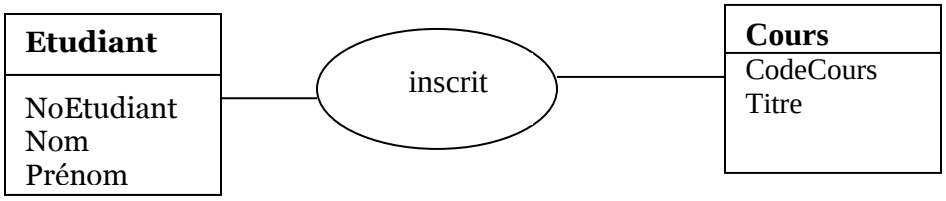
NoEtudiant
Nom
Prénom

Une occurrence de l'entité peut-être 23462007 Courtey Yanick

Relation : (association) : Les entités elles mêmes sont porteuses de peu d'information, mais si celles-ci sont associées à d'autres entitésalors l'information devient plus pertinente et plus conséquente.
En effet l'entité Etudiant est porteuse de deux informations Nom et Prénom, nous sommes renseignés sur l'identité de l'étudiant. Par contre si l'entité Etudiant est associée à l'entité Cours, alors nous pouvons être renseignés sur les cours que suit un Étudiant.

Une relation représente une association entre deux ou plusieurs entités traduisant un fait ou un événement dont le système voudra garder trace. Le nom choisi pour une relation est un verbe d'action permettant de décrire la relation.

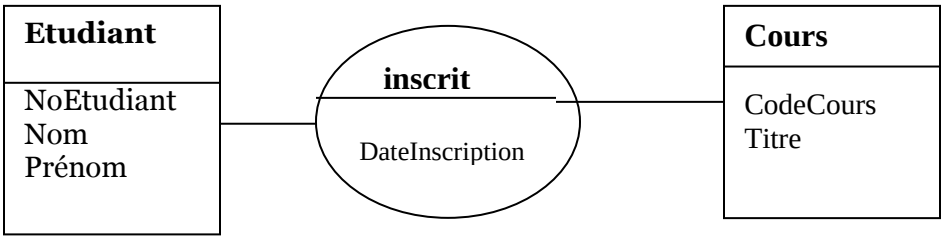
Dans le formalisme de Merise, une relation est représentée par un cercle ou un ovale.



Remarque :

Dans le formalisme de Merise les relations peuvent contenir des attributs.

Dans l'exemple précédent, on peut inclure la date d'inscription à un cours dans la relation (plus loin, lors de la normalisation, nous expliquerons ce choix)



Attributs (propriétés) : Un attribut est la plus petite information que l'on conserve dans le système.. un attribut permet d'identifier, de décrire, ou de quantifier une entité ou une relation. Un attribut est obligatoirement rattaché à une relation ou une entité et doit avoir une signification UNIQUE dans le modèle conceptuel et ne peut être rattaché qu'a un seul concept (entité ou relation)

La valeur d'un attribut est une donnée dans le système.

5.3. Types d'attributs Il existe deux types d'attributs :

a. **Attributs descripteurs :** l'attribut descripteur d'une entité sert à qualifier , à quantifier ou encore à décrire l'état d'une entité de sorte que plusieurs occurrences de la même entité peuvent se répéter

Exemple : NomEtudiant est un attribut descripteur pour l'entité Etuduiant.
Plusieurs occurrences de l'entité Etudiant peuvent avoir le même nom.
Pour la même raison l'attribut **ville** est un attribut descripteur pour l'entité Clients.

b. Attributs identificateur ou identifiant : il sert à identifier les occurrences d'une entité de manière unique

Exemple :
L'attribut Code_Permanent est un attribut identificateur pour l'entité Etudiant.
Chaque occurrence de l'entité Etudiant est identifiée de manière unique par le Code_Permanent. Il n'existe pas d eux étudiants ayant le même code permanent.
Une entité ou une relation doit toujours avoir un identifiant.

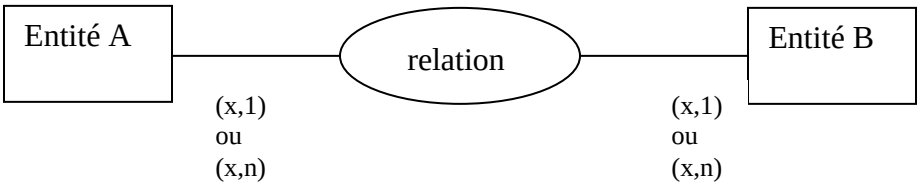
Généralement l'identifiant est le premier attribut d'une entité ou d'une relation et il est souligné

5.4. Types de relation :

Il existe plusieurs types de relation

- Les relations binaires simples : ce sont des relations qui lient deux entités pas plus.
- Les relations n-aires, plus de deux entités participent à la relation
- Les relations réflexives, c'est une relation sur la même entité

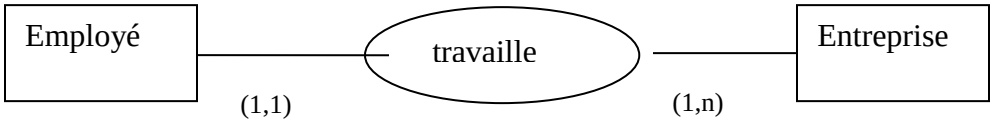
5.5. Notion de cardinalités (multiplicité) : une cardinalité est le nombre de fois minimum et maximum qu'une occurrence d'une entité participe à une relation.



La valeur de gauche (**l'optionalité)** indique si les occurrences des l'entité sont obligées de participer à la relation. Les valeurs possibles sont : 0 pour optionnel et 1 pour obligatoire.

La valeur de droite le degré ou la **multiplicité**) indique le nombre maximum de fois qu'une occurrence de l'entité participe à la relation. Les valeurs possibles sont 1 pour une seule participation et N pour plusieurs participation.

Exemple



Un employé travaille pour 1 et 1 treprise et une entreprise t plusieurs employés.

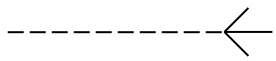
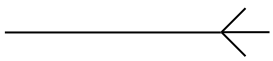
Le modèle Entités associations : Formalisme d'oracle.

Dans le formalisme d'oracle on met en évidence les entités avec leurs propriétés et les identifiant en respectant le formalisme suivant :

Caractéristiques

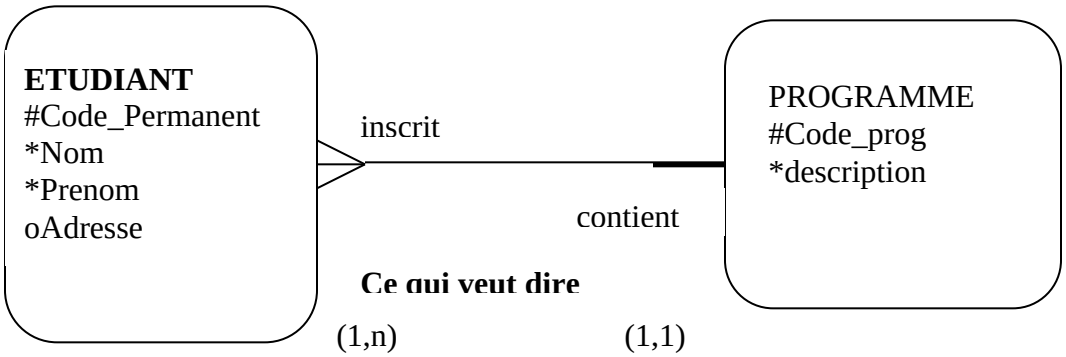
- ✓ **Les relations ne sont pas porteuses de propriétés** (contrairement au modèle Merise)
- ✓ Les relations ont des noms indiquant leurs rôles dans le système.
- ✓ Dans une entité l'identifiant est précédé du symbole **#**
- ✓ Dans une entité les attributs obligatoires sont précédé du symbole *
- ✓ Dans une entité, les attributs optionnels sont précédés du symbole **o**
- ✓ Les cardinalités sont représentées par des lignes avec les terminaisons possibles suivantes :

Représentation des cardinalités

	Optionnelle	signification	Obligatoire	signification
Unique		(0,1)		(1,1)
Multiple		0,n)		(1,n)

Pour une entité dans le modèle, la ligne prés de l'entité indique **l'optionalité** et la ligne au bout indique la **multiplicité**.

Exemple 1 et explication



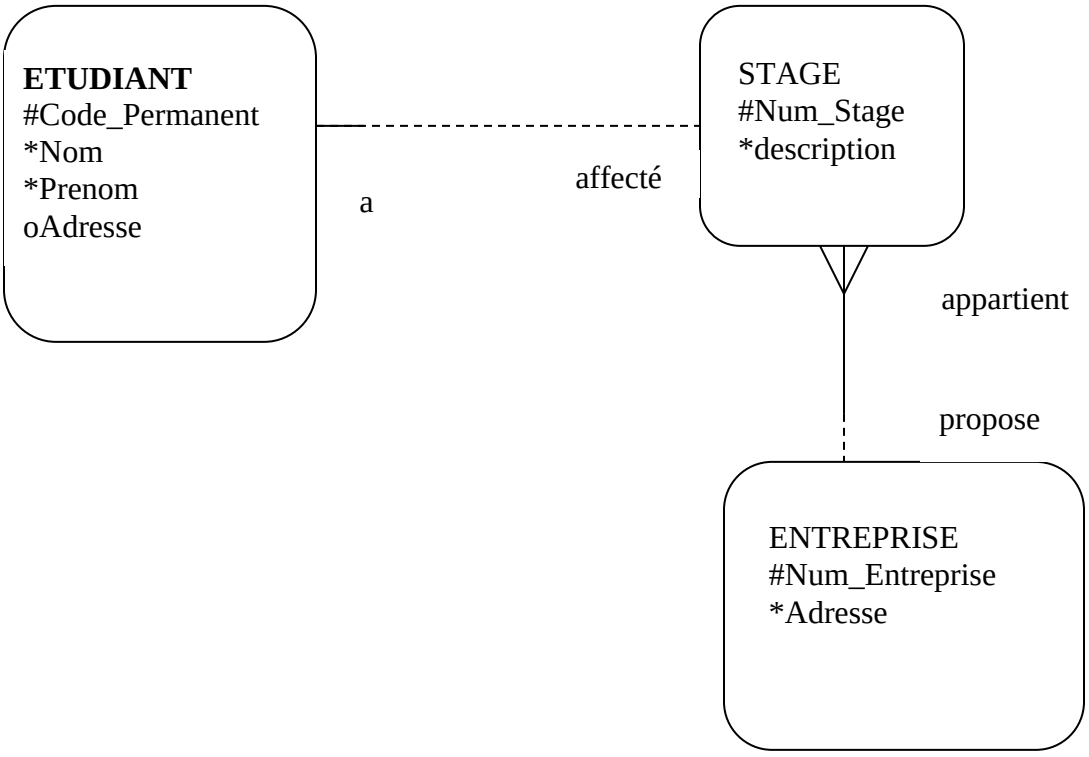
Pour la relation *inscrit*

La ligne étant continue du côté de l'entité Etudiant, donc il s'agit d'une obligation. Comme il n' y a pad de patte d'oie du côté de l'entité Programme, alors il s'agit de l'unicité. On peut lire alors la relation :Un étudiant est inscrit (le nom de la relation) à un seul programme.

Pour la relation *contient*

Du côté de l'entité Programme, la ligne et continue, il s'agit d'une obligation. Du côté de l'entité Etudiant, la terminaison de la relation est une patte d'oie, donc il s'agit de la multiplicité. On lira la relation de la façon suivante. Un programme contient 1 à plusieurs étudiants.

Exemple 2

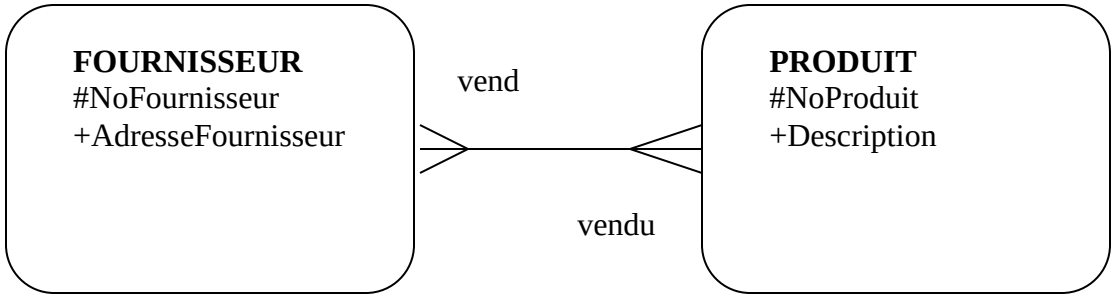


Pour la relation *a*, on peut la lire de la façon suivante : un étudiant a 0 ou 1 stage.

La relation *affecté* se lie de la même manière. En effet un stage est affecté à 0 ou 1 étudiant.

Une entreprise propose 0 ou plusieurs stages, c'est ainsi que nous pouvons lire la relation *propose*. Par contre, un stage appartient 1 et une seule entreprise.

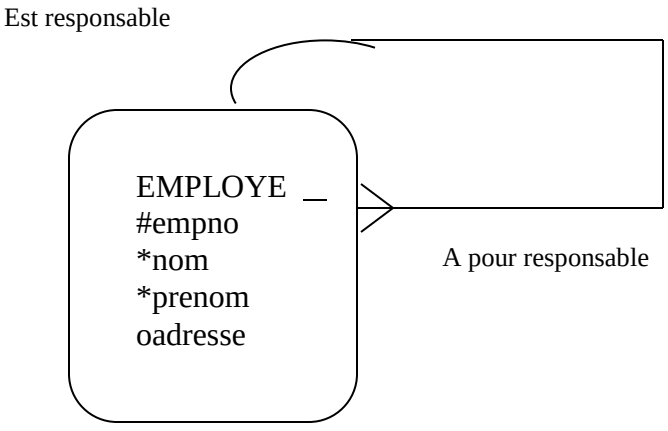
Exemple 3



Attention : dans le formalisme d'oracle, les cardinalités signifient la même chose que dans le modèle MERISE. MAIS elles sont placées de l'autre côté de l'entité.

Remarque : Toutes les relations décrites plus haut sont des relations binaires simples.
Sauf cas contraire (si le système l'exige), il est fortement recommandé de modéliser notre base de données en utilisant des relations binaires

Exemple d'une relation réflexive :
Il existe des systèmes qui exige qu'une relation soit relié à la même entité. par exemple dans une entreprise si nous voulons représenter les employés avec leur responsable immédiat il est claire que nous avons besoin d'une relation sur l'entité EMPLOYE avec l'entité e EMPLOYE. La représentation sera la suivante



Autres concepts

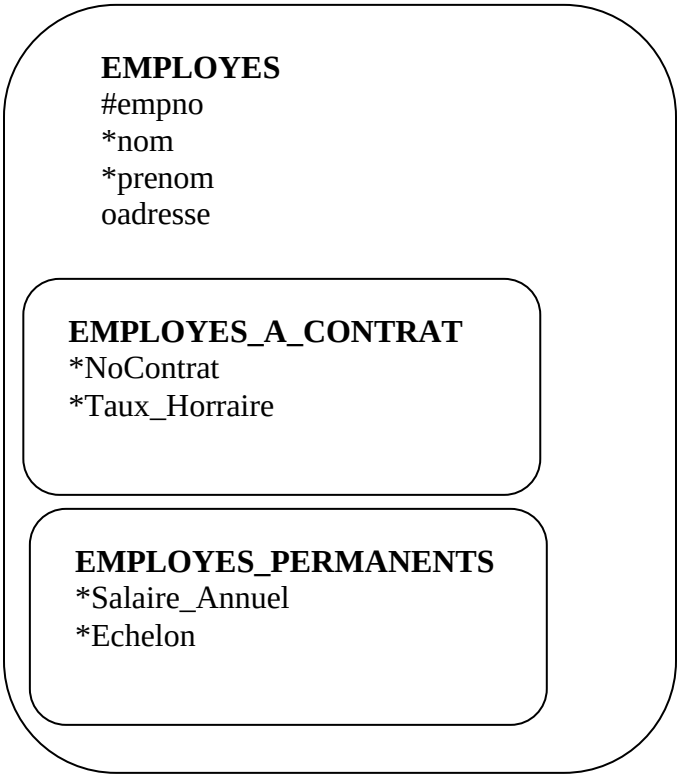
Généralisation d'entité.

La généralisation d'entité consiste à regrouper sous la même entité plusieurs entités ayant les mêmes caractéristiques.(un peu comme le concept d'héritage en POO.).

On appelle entité *surtype,* l'entité qui englobe plusieurs entités.

On appelle entité *sous-type* une entité à l'intérieur de l'entité surtype.

Exemple



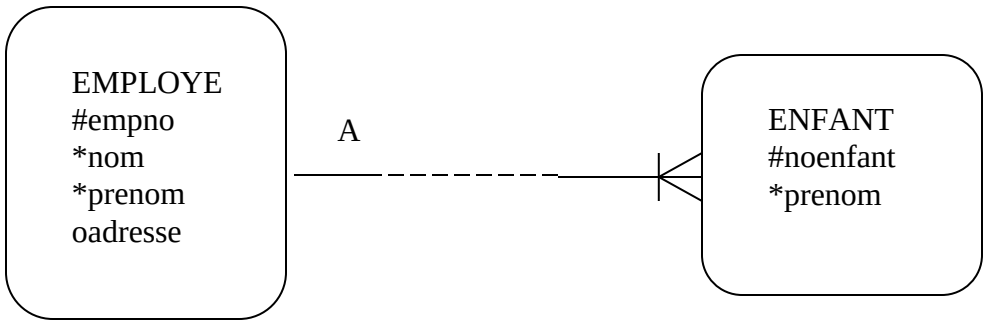
Dans le modèle précédent, l'entité EMPLOYÉS est l'entité *surtype* . Les entités EMPLOYES_PERMANENTS et EMPLOYE_A_CONTRAT sont des entités sous-type. Chacune d'elles possèdent ses propres propriétés et les propriétés de l'entité surtype EMPLOYÉS.

Ce type de modèle fait penser au concept d'héritage dans la POO.

Identifiant Relatif

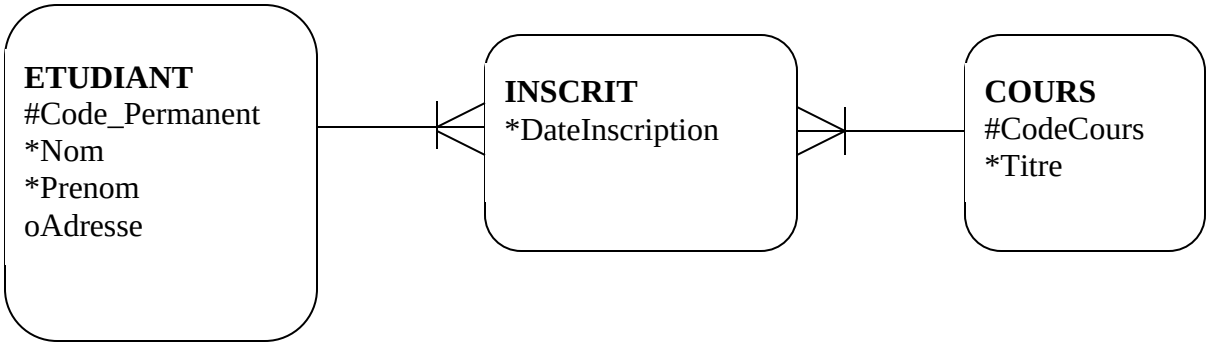
Il existe des entités dont les occurrences ne peuvent être identifiées par l'identifiant de l'entité en question, on dit alors que l'entité a un **identifiant faible**. Pour identifier parfaitement les occurrences, nous devons utiliser l'identifiant d'une autre entité.

Exemple 1



L'identifiant de l'entité ENFANTT est faible. L'entité ENFANT est entièrement dépendante de l'entité EMPLOYE, donc pour identifier l'enfant, nous avons besoin de connaître l'identifiant de l'employé à qui appartient l'enfant.

Exemple 2



Vérification et normalisation du modèle entité/relation

La vérification

On dit qu'un modèle conceptuel (entité/relation) est vérifié s'il répond aux conditions suivantes :

- Touts les attributs non calculés sont présents dans une entité ou une relation dans le cas du modèle de MERISE)
- Aucun attribut n'est redondant
- Toutes les entités ont un identifiant
- Tous les attributs sont élémentaires (non décomposable ou se traitent comme un tout, exemple l'attribut adresse peut-être décomposable)

La normalisation du modèle Entité/Relation

Les règles de normalisation du modèle conceptuel permettent d'assurer :

- La non redondance des données
- L'intégrité des données
- La facilité de mise à jour

Lors de la normalisation d'un modèle de données (peu importe le niveau de modélisation, de conceptualisation ou de représentation) nous devons avoir au mois la 3eme forme normale.

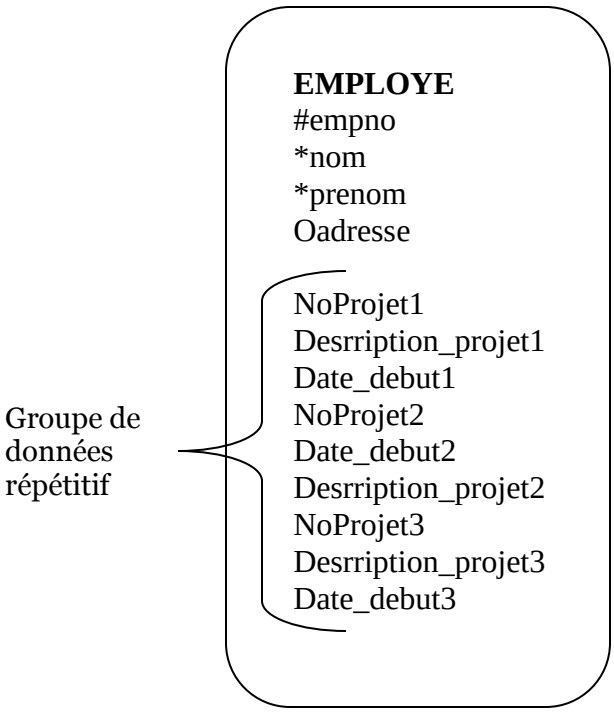
Définition :

Une base de données est normalisée si celle-ci est en 3eme forme normale.

Première forme normale : La clé (énoncé 1)

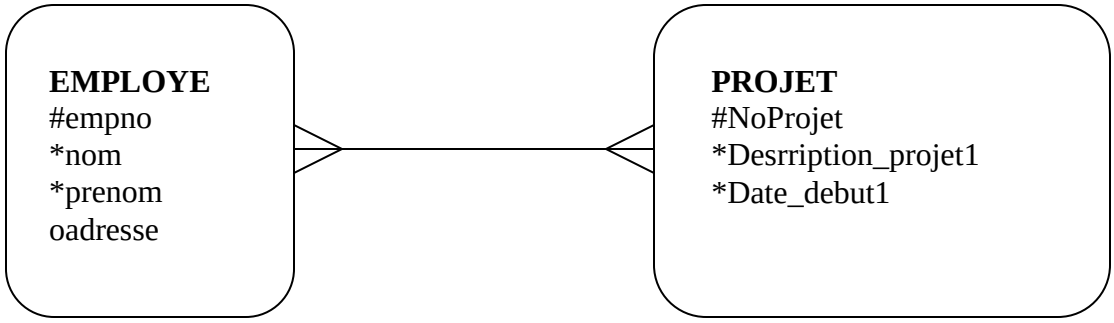
Les entités ne doivent pas contenir des groupes de données répétitives. Si tel est le cas, il faut sortir le groupe de données et créer une autre entité qui va contenir ce groupe de données.

Problèmes avec cette représentation :



- ✓ On ne sait pas combien de projets sont affectés à chaque employé. Ce qui veut dire que lors de la création de la base de données, on ne sait pas combien de champs en rapport avec projet que nous allons créer (0?, 10?, 45? 1000?,..)
- ✓ Problème de redondance si la compagnie qui gère ces employés a affecté 200 employés à un projet alors on retrouvera 200 fois la même description du projet et 200 fois la date début du projet pour le projet en question.

La première forme normale (1FN) va donner la représentation suivante



Les cardinalités sont 1 :N ou 0 :N, dépendamment si l'employé est obligatoirement (obligation don 1) affecté à un projet ou non (si no alors l'optionalité alors 0)
Le schéma de la solution représente l'obligation. (1 :N)

De même, le système déterminera également les cardinalités du côté EMPLOYE (un projet est-il affecté à 0 :1 ou 1 :1 ou 0 :N ou 1 :N) employé. Le schéma indique qu'un projet est affecté 1 :N employés.

Deuxième forme normale **Dépendance totale de la Clé (énoncé 2)**

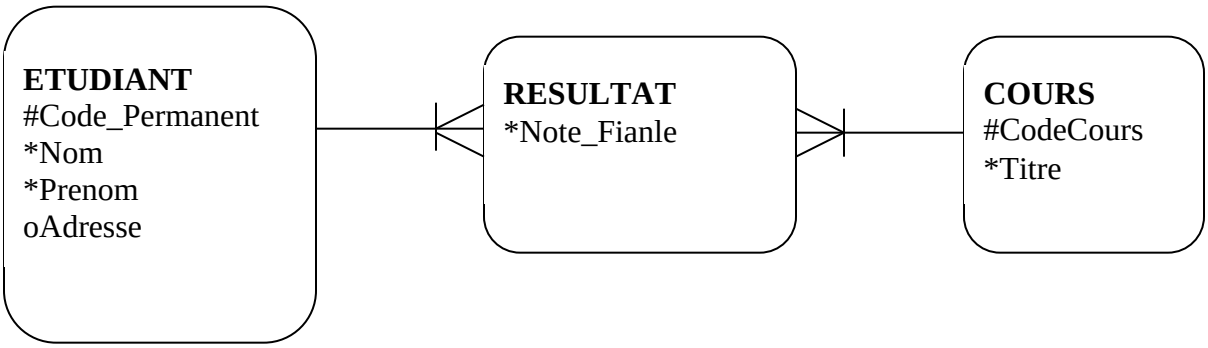
Une entité ou une relation est en deuxième forme normale 2FN, si elle est :

a. En 1FN
b. Tous les attributs de la relation ou de l'entité dépendent de **toute la clé** (identifiant) et non d'une partie de la clé.

Ce genre de vérification s'applique surtout aux relations avec cardinalités (m :N), m étant égale à 0 ou 1 et N >1

Exemple :

Pour l'ensemble des étudiants de 3eme session informatique de gestion, on souhaite garder les notes finales qu'ils ont obtenues pour chaque cours.



Dans le modèle précédent, il est clair que la note finale d'un étudiant dépend de l'étudiant et du cours qu'il suit.

Voici un exemple de notes finales que l'on peut avoir.

Code_Permanent	CodeCours	Note_Finale
Poif9859-06	420-KED	85
Lef765-06	420-KED	73
Faf231-05	420-KED	77
Poif9859-06	420-KEG	72
Lef765-06	420-KEG	82
Poif9859-06	420-KEF	80

(Les codes permanents ne sont pas exacts)

Remarquez dans le tableau précédent que si l'on connait le que code permanent Poif9859-06 on ne peut pas être renseigné sur sa note finale, il, faudrait connaître en plus du code permanent, le code cours. De la même façon, si je connais le code cours 420-KED, je ne peux pas être renseignée sur la note finale dans le cours correspondant. L'attribut Note_Finale dépend ET du Code_Pernanent ET du code cours. La place de l'attribut.

Si nous devons placer l'attribut Note_Finale dans l'entité Étudiant (Ou cours), cet attribut dépendra d'UNE PARTIE DE LA CLÉ. Nous ne sommes pas en 2FN.

Troisième forme normale : Et rien que la clé (aucune dépendance transitive.) (énoncé 3)

Une entité ou une relation est en troisième forme normale (3FN) si

c. Elle est déjà en 2FN
d. Tous les attributs non clé dépendent uniquement de la clé. Il n' y a pas de dépendance entre deux attributs non clé.

L'entité ETUDIANT suivante, est en 2FN.

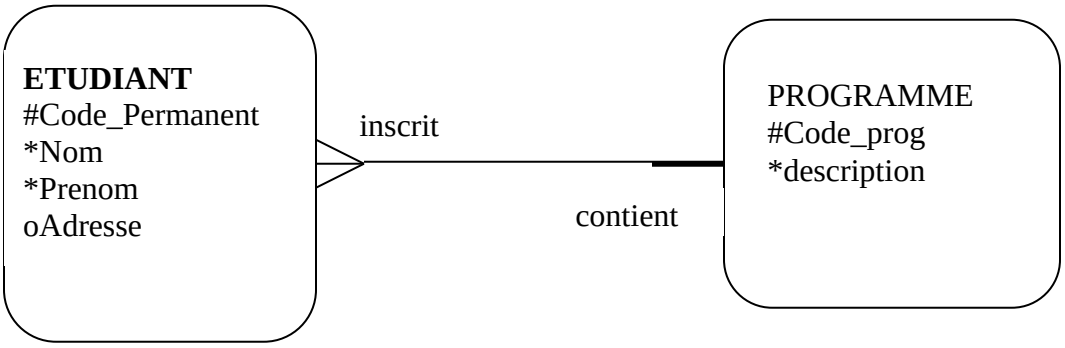
Elle est en 1FN, puis que pas de groupe répétitif, et tous les identifiants dépendent de la clé

Elle est en 2fN, puis que tous les attributs dépendent de la totalité de la clé.

La même entité n'est pas en 3FN Pourquoi?

ETUDIANT
#Code_Permanent
*Nom
*Prenom
oAdresse
Code_prog
NomProg

La raison pour la quelle cette n'est pas en est que l'attribut NomProg dépend aussi de l'attribut Code_prog. (le code 420 implique informatique.)
Pour avoir une entité en 3FN, il faut sortir les attributs Code_prog et NomProg de l'entité, les mettre dans une entité à part (PROGRAMME) et créer une relation entre ETUDIANT et PROGRAMME



Le Modèle Relationnel

Introduction :

Le modèle relationnel (mis de l'avant par Edgar Frank Codd**)** est le plus implémenté et le plus stable des modèles actuellement utilisés. Dans ce modèle la gestion des données est simplifiée grâce aux contraintes d'intégrité. Le système de récupération des données permet à l'utilisateur de visualiser la base de données à travers des structures de tables relationnelles.

L'unité de stockage élémentaire est la « table ». En général, c'est aux tables que les utilisateurs font référence pour accéder aux données. Une base de données est constituée de plusieurs tables reliées entre elles.

Une table est constituée de lignes et de colonnes

Exemple de table

Voici la table **Livres** qui contient des informations par rapport à des livres de notre bibliothèque.

NumeroLivre	TitreLivre	Auteur	Langue
101	Ainsi parlait Zarathoustra	Friedrich Nietzsche	Allemand
102	La paix des profondeurs	Aldous Huxley	Anglais
103	Les misérables	Victor Hugo	Français
104	La liberté n'est pas une marque de yogourt	Pierre Falardeau	Françiais

Remarquez

1- Chaque colonne définit un attribut. Un attribut est appelé également champ de la table. Les champs servent à donner la Définition d'une table.
2- Chaque ligne définit une instance (ou une occurrence) de la table Livres. Une ligne représente également un enregistrement d'une table.
3- Une table est donc un ensemble d'enregistrements. Pour définir un enregistrement on utilise des champs ou des attributs.

Dans une base de données Oracle, on utilise la commande CREATE pour créer une table.

Exemple

```
CREATE  TABLE Livres (NumLivre NUMBER(5), TitreLivre  CHAR (20), Auteur CHAR(20), Langue CHAR (10));
```

Pour insérer des données dans une table, on utilise la commande INSERT

Exemple

```
INSERT INTO Livres VALUES (101,'SQLPLUS', 'Oracle','Français');
```

Pour afficher le contenu d'une table on utilise la commande SELECT

Exemple

SELECT * FROM Livres, permet d'afficher Tous les enregistrement de la table Livres.

Définitions

Termes	Définitions et représentation
Domaine	Ensemble de valeur caractérisé par un nom. Exemple le domaine de noms de jeux vidéo peux être Jeux {Word of Warcraft, Starcraft, Vampire, Civilization } Sexe {F, M}
Relation	Sous-ensemble du produit cartésien d'une liste de domaines caractérisé par un nom exemple
Schéma d'une relation	Nom de la relation suivi de la liste des attributs : Etudiant (Nom, Prenom, Code_permanent, adresse)
Attribut	Sous groupe d'information à l'intérieure d'une relation Code_permanent, Nom Un attribut peut être considéré comme une colonne d'une relation
Valeur d'un attribut	La valeur de l'attribut en rapport avec le domaine de valeurs. Exemple valeur de l'attribut Nom est Martin
Occurrence	Instance d'une entité ou d'une relation
Identifiant	Attribut permettant d'identifier les occurrences d'une entité ou d'une relation d'une manière unique

Table	Objet d'une base de données relationnelle. Ensemble de lignes et de colonnes. (ce concept sera défini en détail dans le chapitre suivant)
Clé primaire	Identifiant dans une table. Attribut permettant d'identifier chaque occurrence d'une table de marinière unique. Ou encore attribut permettant d'identifier chaque ligne d'une table d'une manière unique.
Clé étrangère	Attribut d'une table (table A) qui est clé primaire d'une table (table B)

Passage du modèle Entité/Association au modèle relationnel :

Les règles qui vont suivre s'appliquent au modèle Entité/Association du Formalisme Oracle

Règle 1 : Entité excluant les entités surtype et sous-type :

Toute entité du modèle Entité/Association est traduite par une table dans le modèle relationnel

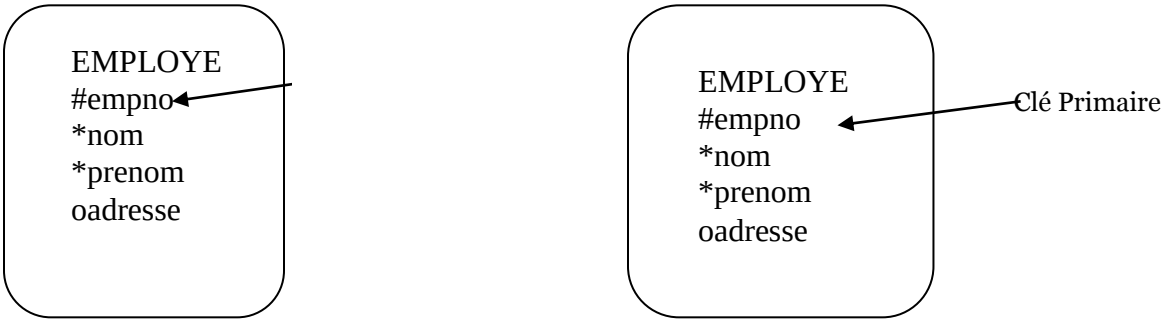
La clé primaire de cette table est l'identifiant de l'entité

Les attributs de la table sont les attributs de l'entité.

Exemple

Modèle Entité /Association (Oracle) **Modèle Relationnel**

L'entité **EMPLOYE** devient la table **EMPLOYE**



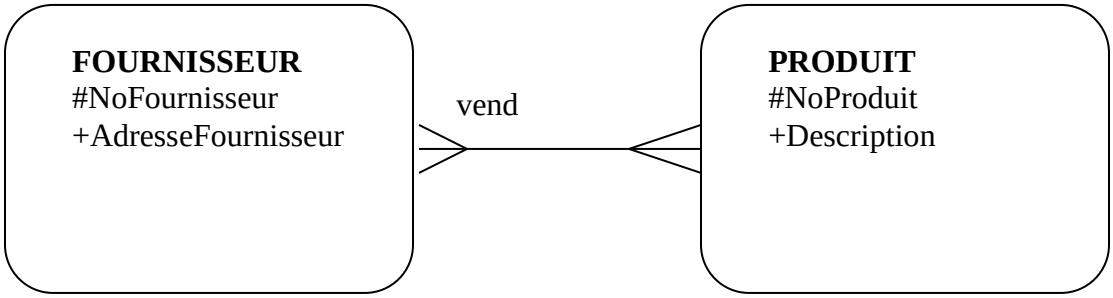
Règle 2 : Relation binaire avec participation multiple :

CAS : 0 :N —— 0 :N OU 0 :N —— 1 :N OU 1 :N —— 1 :N

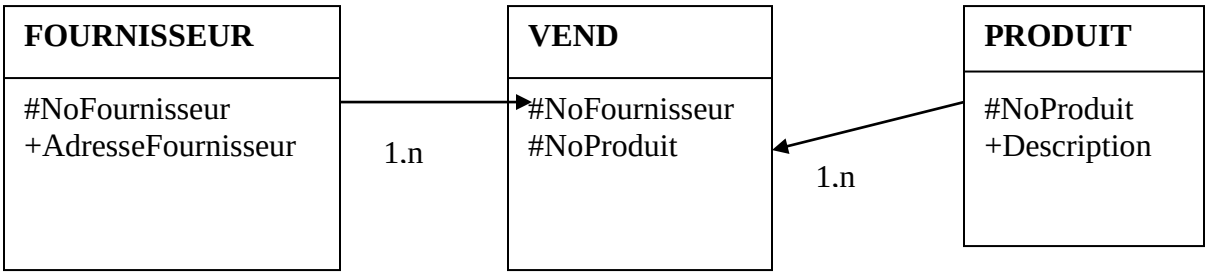
- la relation devient t une table
- la nouvelle table hérite des clés primaires provenant des tables reliées pour devenir des clés étrangères. Ces clés étrangères forment également une clé primaire composée de la nouvelle table.
- Les clés étrangères sont auront des valeurs obligatoires (puisqu'elles forment une clé primaire composée) et la participation est multiples. (la valeur de la clé étrangère peut revenir plusieurs fois).
- Dans le formalisme de MERISE, si la relation est porteuse de propriétés alors ses propriétés sont des propriétés de la nouvelle table.

Exemple et représentation graphique :

Modèle E/A



MODÈLE RELATIONNEL : Nous avons 3 tables.



Explications

Selon la règle 1, les entités sont devenues des tables :

- la table Fournisseurs, dont la clé primaire est NoFournisseur
- la table produit, dont la clé primaire est NoProduit

Selon la règle 2, nous obtenons la table **vend** dont la clé primaire est la clé composée NoFournisseur, NoProduit.

Dans la table Vend, les clés étrangères ont une participation obligatoire et multiple.

Obligatoire car ce sont des clés primaires (NOT NULL, non nullité)

Multiple, car un la participation des occurrences des entités (Fournisseurs et Produits) dans la relation VEND est multiple.

Exemple d'occurrences de la table Fournisseurs

NoFournisseur	AdresseFournisseur
F001	10 rue de l'oracle Montréal
F002	45 avenue de la paix Blainville

Exemple d'occurrences pour produit

NoProduit	Description
C100	Chaise de bureau
M100	Meuble pour ordinateur
P102	

Exemple d'occurrences pour VEND

NoProduit	NoFournisseur
C100	F001
M100	F001
P102	F001
C100	F002
M100	F002

IMPORTANT :

Dans la table PRODUITS, on ne peut pas trouver la clé primaire qui se rèpete.C100, ne doit se retrouver qu'une seule fois dans toute la table. (Définition d'une clé primaire). Dans la table VEND, cette occurrence de la clé primaire se répète pour chaque fournisseur qui vend le produit

Règle 3 : Relation binaire avec participation multiple d'un côté et unique de l'autre côté

Cas **0 :N** —— **0 :1** **0 :N** —— **1 :1**

OU

1 :N —— **0 :1** **1 :N** —— **1 :1**

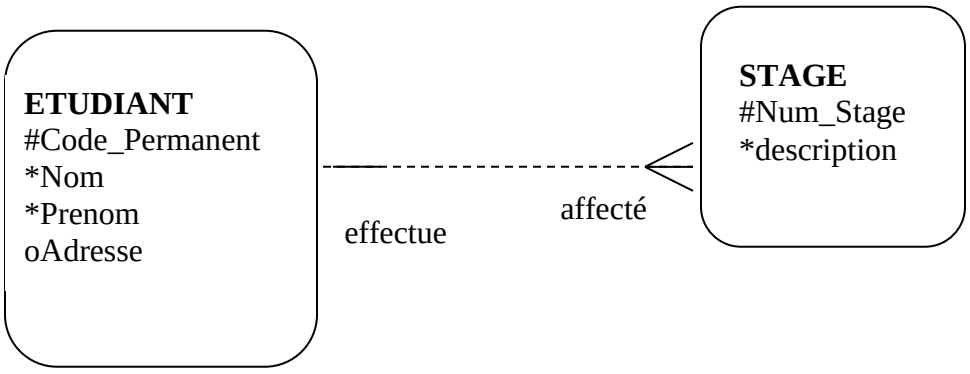
Exemple et explication

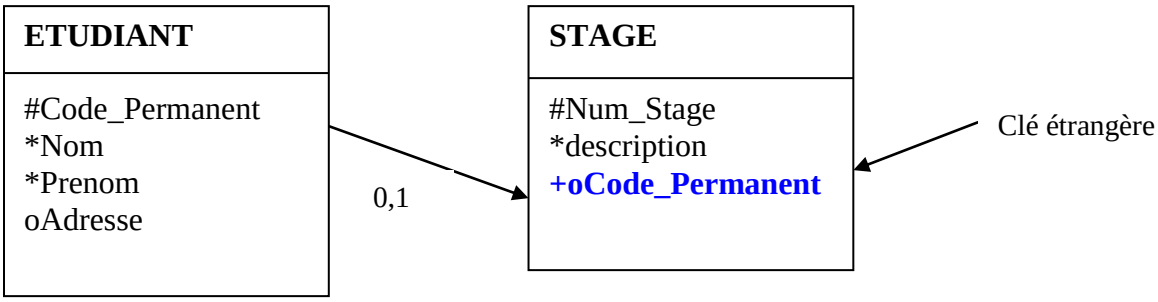
Modèle E/A (oracle)

Le modèle E/A suivant exprime le fait suivant :

- un étudiant effectue 0 à N stages (0 :N)
- un stage est affecté à 0 ou 1 étudiant. (0 :1)

Nous avons donc à faire à un cas de la règle 3





Remarque 1

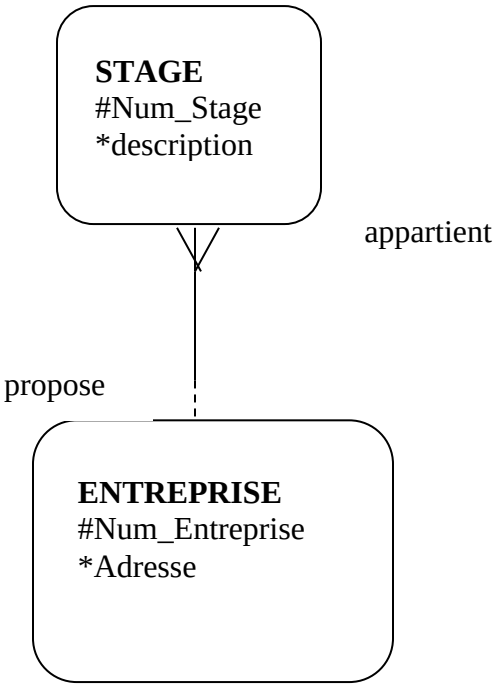
Dans la table STAGE, remarquez que Code_Permanent (clé primaire de la table Étudiant) est devenue une propriété de la table STAGE, cette propriétés est appelé CLÉ ETRANGÈRE**.**

Remarque 2

Code_Permanent a une valeur **optionnelle** dans la table STAGE.

Le nombre maximum qu'une valeur de Code_permanent va se répéter dans la table stage est 1 (UNE) fois.

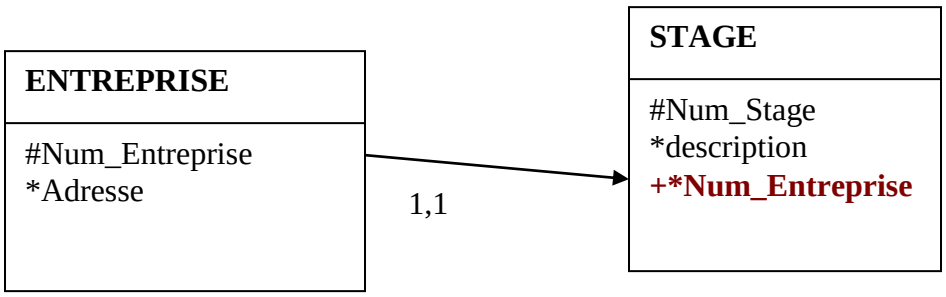
Exemple 2



Le modèle E/A ci-contre exprime le fait suivant :

- Une entreprise propose 0 à N stage (0 :N)
- un stage appartient à 1 et une seule entreprise. (1 :1)

Nous avons donc à faire à un cas de la règle 3



Remarque 1

Dans la table STAGE, remarquez que Num_Entreprise (clé primaire de la table Entreprise) est devenue une propriété de la table STAGE, cette propriétés est appelé CLÉ ETRANGÈRE**.**

Remarque 2

Num_Entreprise a une valeur **Obligatoire** dans la table STAGE.

Le nombre maximum qu'une valeur de Num_Entreprise va se répéter dans la table stage est 1 (UNE) fois.

Énoncé de la règle 3

- La table avec participation unique hérite de la clé primaire provenant de la table opposée pour devenir une clé étrangère. Si la participation est obligatoire (1 :1) alors la valeur de la clé étrangère est obligatoire. Si la relation est avec un identifiant relatif, alors la clé étrangère fait partie de la clé primaire qui devient alors une clé primaire composée.
- Le lien entre les tables pointera sur la table qui a hérité de la clé étrangère. Le lien sera obligatoire si la table qui a hérité de la clé étrangère avait une participation obligatoire à la relation et optionnel dans le cas contraire.

Règle 4 : Relation binaire avec participation d'un côté et unique des deux côtés

Cas **0 :1** —— **0 :1** **0 :N** —— **1 :1**

OU **1 :** —— **0 :1** **1 :** —— **1 :1**

Cas (0:1----1:1)

- La table avec la participation obligatoire hérite de la clé primaire provenant de la table opposée pour devenir une clé étrangère avec la participation obligatoire. Si la relation est avec un identifiant relatif, alors la clé étrangère fait partie de la clé primaire qui devient alors une clé primaire composée.
- Le lien entre les tables pointera sur la table qui a hérité de la clé étrangère. Le lien sera obligatoire avec la participation unique.

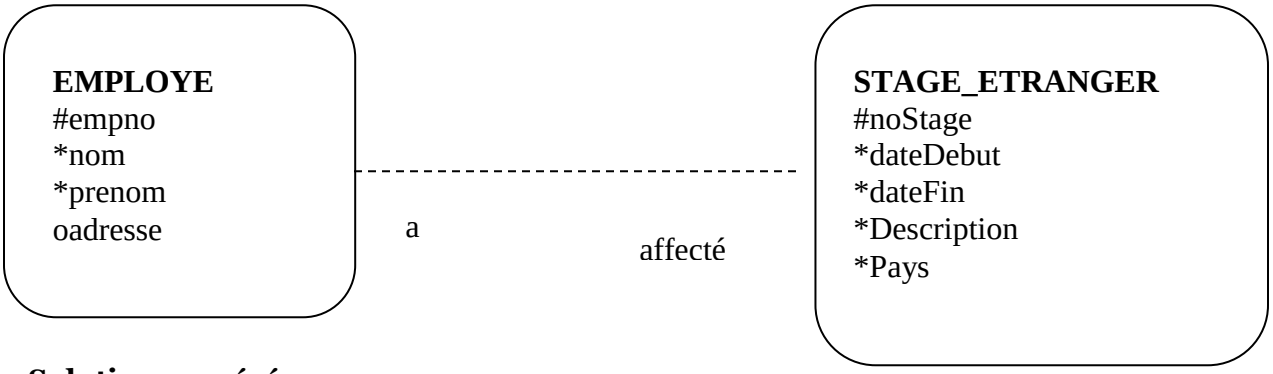
Exemple

Cas (0:1—0:1)

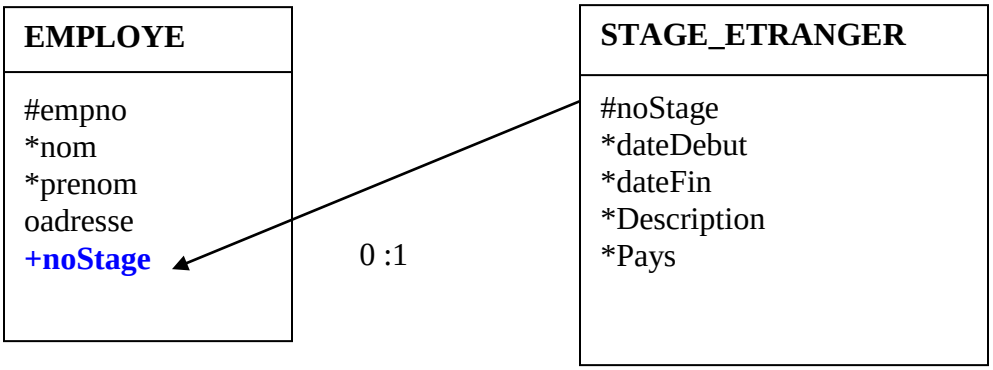
- Dans ce cas n'importe qu'elle table peut hériter de la clé primaire de l'autre table pour devenir une clé étrangère avec valeur optionnelle. Le choix de la table qui va hériter de la clé dépendra du système.
- Le lien entre les tables pointera sur la table qui a hérité de la clé étrangère. Le lien sera optionnel avec la participation unique

Exemple

Dans une petite PME, les employés ont droit à au maximum 1 stage à l'étranger.
Un stage à l'étranger est affecté à une seule personne. (Certains stages ne sont pas affectés car ils n'intéressent pas les employés)



Solution suggérée



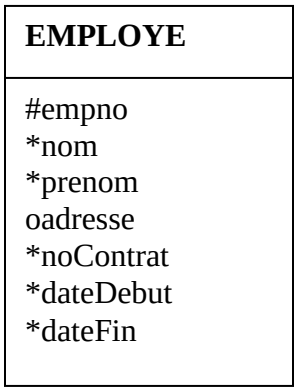
Dans cette PME, ce qui nous intéresse est de savoir pour chaque employé s'il a effectué un stage à l'étranger. Si **noStage** dans la table EMPLOYE a une valeur (non Nulle), cela veut dire que l'employé a eu son stage. Pour avoir les informations du stage, il faut aller dans la table STAGE_ETRANGER.

Cas (1:1----1:1)

- On doit d'abord vérifier si l'on ne peut pas inclure les deux tables dans une seule, ce qui est en général le cas. Dans ce cas-ci, la clé primaire de la nouvelle table sera l'une ou l'autre des clés primaires des deux tables initiales, une clé composées des deux clés primaires, ou alors une nouvelle clé primaire
- Si l'on ne peut pas scinder les deux tables en une seule, alors les mêmes règles citées pour le cas optionnel s'appliquent, avec valeur obligatoire.

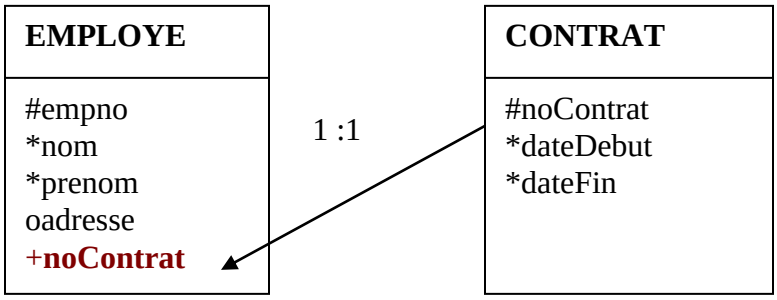
Exemple

Soit on garde une seule table comme suit et on choisi de garder le empno comme clé primaire. (on peut également choisir de garder le noContrat comme clé primaire ou une composition des deux clé primaires)



Autre solution

Cette solution consiste à faire migrer la clé primaire de la table CONTRAT, dans la table EMPLOYE. Cette Clé aura une valeur unique obligatoire



Remarque : il existe une autre solution. Elle consiste à faire migrer la clé primaire de la table EMPLOYE, dans la table CONTRAR. Cette Clé aura une valeur unique obligatoire

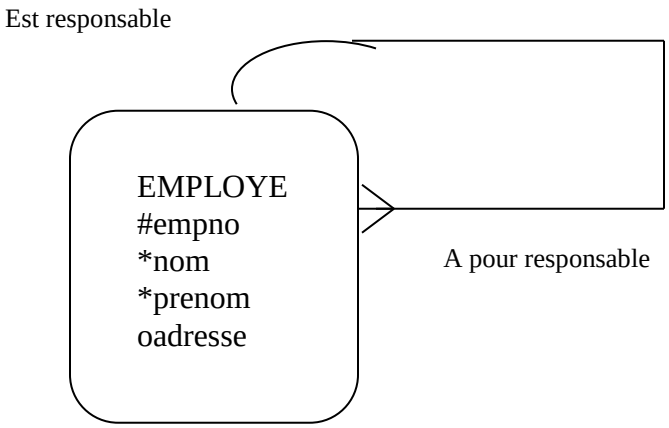
Règle 5: relation récursive avec participation multiple.

- La relation devient une table de lien avec une clé primaire composé de deux fois l'identifiant de l'entité. les attributs constituant la clé primaire sont également qualifiés de clé étrangères..
- Les lient entre les deux tables pointeront sur la nouvelles table. Les liens sont obligatoires car les clés étrangères sont devenues une clé primaire composée. La participation est multiple

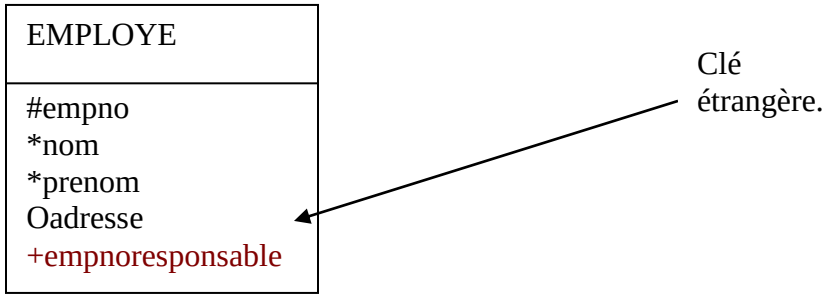
Règle 6: relation récursive avec participation unique d'un côté

- Las clé primaire est dupliquée et devient une clé étrangère avec appellation différente de la clé primaire.
- Le lien reste récursif et sera défini en fonction de la participation de l'entité à la relation.

Exemple :



Résultat



Règle7: relation entité surtype et sous-type.

Il existe plusieurs solutions qui permettent de passer une telle représentation, la plus simple est la suivante :

- Regrouper toutes les propriétés (attributs) dans une seule table.
- Un nouvel attribut est créer afin d'identifier le type d'occurrence
- L'identifiant unique du surtype devient la clé primaire de la table. Les attributs des entités sous-type deviennent les attributs de la table avec valeurs optionnelle.

L'exemple de la page 11 va donner le résultat suivant :

EMPLOYE
#empno *nom *prenom oadresse **TypeEmp** oNoContrat oTaux_Horraire oSalaire_Annuel oEchelon

On obtient une seule table EMPLOYE avec la clé primaire empno
Nous avons ajouté un attribut **TypeEmp** pour identifier le type d'employé.

TypeEmp Peut prendre la valeur 1 s'il s'agit d'un employé permanent et la valeur 2 s'il s'agit d'un employé à contrat.

NoContrat, Taux_Horraire, Salaire_Annuel et Echelon on des valeurs optionnelles. Si l'employé est permanent alors la valeur de l'attribut Taux_Horraire peut être à NULL (pas zéro). Le contrôle de ces rubriques optionnelles peut se faire avec une contrainte CHECK dans ORACLE

Normalisation du modèle relationnel

Un modèle relationnel issu d'un modèle Entité/Relation normalisé (en 3FN) est aussi en 3FN

Les règles de normalisation **(énoncé 1), (énoncé 2) (énoncé 3)** s'appliquent au modèle relationnel

Comme la normalisation d'une base de données est très importante alors il convient de rappeler les énoncés de normalisation. (1FN,2FN,3FN)

Définition : une base de données relationnelle est dite normalisée, si elle est au moins en 3FN.

Énoncé 1 :La 1FN : (La clé)

Les relations (les tables) ne doivent pas contenir des groupes de données répétitives. Si tel est le cas, il faut sortir le groupe de données et créer une autre entité qui va contenir ce groupe de données.

Énoncé 2 : La 2FN (Toute la clé)

Une relation (table) est en deuxième forme normale 2FN, si elle est :

a. En 1FN

Tous les attributs de la relation ou de l'entité dépendent de **toute la clé** (identifiant) et non d'une partie de la clé

Énoncé 3 : La 3FN (Rien que la clé)

Une relation (table) est en troisième forme normale (3FN) si

a. Elle est déjà en 2FN
b. Tous les attributs non clé dépendent uniquement de la clé. Il n'y a pas de dépendance entre deux attributs non clé.

La forme normale de Boyce-Codd.(BCNF) (énoncé 4)

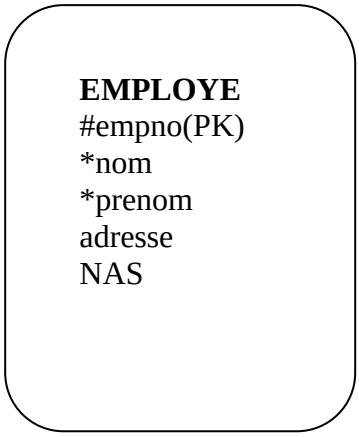
Une relation (table) est en BCNF si

a. Elle est déjà en 3FN
b. Les seules dépendances fonctionnelles sont celles dans lesquelles une clé (entière si composée) détermine un attribut.

Lorsque la clé primaire est composée, il arrive que certaine partie de la clé détermine un attribut non clé. La table peut être en 2FN, en 3FN mais pas en BCNF

Exemple1 :

Voici une table employes avec les attributs suivants :



Cette table, bien qu'elle soit en 3FN n'est pas en BCNF puisque le nom et prénom dépendent aussi du NAS.

Exemple2 :
Soit la table suivante : Cette table est bel et bien en 3FN

- Dans la table LocalisationVin, il y'a une clé primaire composée de (Cru et Pays). Le Cru et le pays détermine la qualité.
- Le Cru et le pays détermine la région.
- MAIS, Région détermine Pays, donc il y a un attribut élémentaire qui détermine une partie de la clé

On peut dite que la table n'est pas en BCNF.

Pour qu'elle soit en BCNF, il faut

Avoir une table CRUS avec les 3 attributs : **cru** (clé primaire), **Pays**, **Qualité**
Et une table REGIONS avec les attributs **région** (clé primaire), **Pays**

La forme quatrième forme normale 4FN (énoncé 5)

Une table (relation) est en quatrième forme normale si

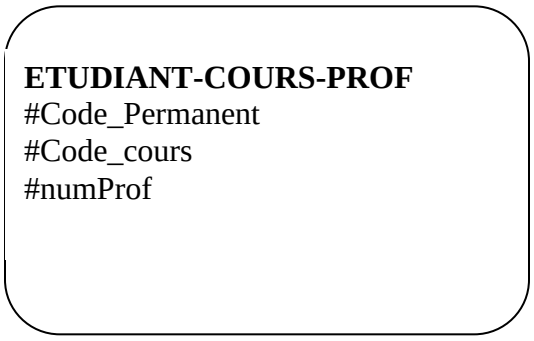
a. Elle est déjà en 3FN
b. Les seules dépendances multivaluées élémentaires sont celles dans lesquelles une clé composée détermine un attribut.

La quatrième forme est une généralisation de la FNBC

Exemple
Soit la table suivante :

Dans cet exemple on considère qu'un cours est donné par un seul prof et qu'un prof peut donner plus d'un cours.

Un étudiant peut suivre plusieurs cours et un cours et suivi par plusieurs étudiants



Ce qu'on veut représenter est la relation entre prof, cours et etudiants.

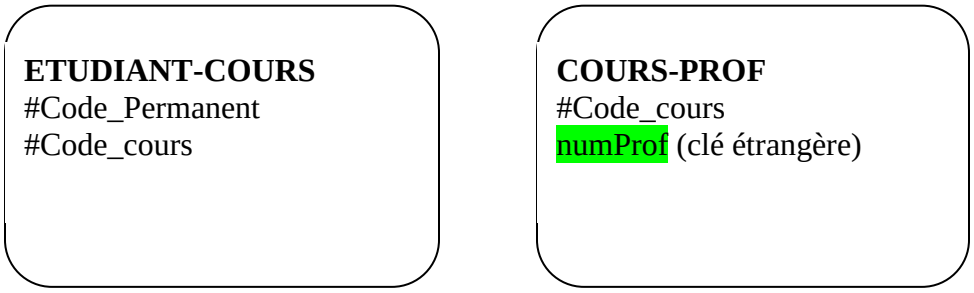
Exemple d'occurrences

Code_Permanent	Code_cours	Num_Prof
100	Français 101	50
100	Français 102	51
100	Anglais101	53
101	Français102	50
100	Anglais102	55
101	Français 101	50
102	Français 101	50

La table Etudiant-Cours-Prof est en 3FN deux problèmes se posent :

1- contient beaucoup de redondances
2- le fait que le **code-cours** implique le **numProf** alors elle n'est pas en 4FN

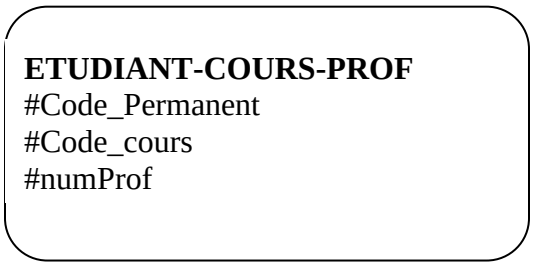
Ce que nous pouvons faire est la chose suivante :



On peut retrouver facilement avec une jointure d'un prof dans un cours donné. (Pas de perte d'information)

Attention !!!
Si un cours est donné par plusieurs professeurs alors le numProf doit être une clé primaire dans la table **COURS-PROF. Dans ce cas nous avons une perte d'information : On ne peut pas retrouver les étudiants d'un prof dans un cours**

Pour pouvoir retrouver les étudiants d'un prof dans un cours, il faudra revenir à la première table qui est :



Sources

Shamkant B. Navath, Evolution of Data Modeling for Databases, communication of the ACM, 1992

James A.O'Brien, Guy Marion et Gilles Saint Amant, **Les systèmes d'information de gestion : La perspective du gestionnaire utilisateur,** 1995

Ryan K.Stephens & Ronald R.Plew, Conception de bases de données, ISBN 2-7440-1176-2

Georges Gardarin, Bases de Données objet et relationnel, ISBN 2-212-0906-9