Hiver 2019

Normalisation

introduction

Table des matières

Historique des versions

Numéro de version	Tâches/modifications	Auteur	Date
1.0	Pages 3 à 10	Marc Beaulne	Mars 2019
1.0	Pages 11 à 13	Saliha Yacoub	Mars 2019

Formes normales

Dépendance fonctionnelle

Une dépendance fonctionnelle est une relation entre deux sous-ensembles de colonnes dans une table.

Supposons les deux sous-ensembles de colonnes comme étant **X** et **Y**. Il y a une **dépendance fonctionnelle** entre ces sous-ensembles (**X → Y**), si et seulement si, pour chaque valeur **X**, dans la table, correspond une seule valeur de **Y**. Donc, si deux lignes dans la table ont la même valeur pour **X**, alors ils ont aussi la même valeur pour **Y**.

Prenons, pour exemple, une version modifiée de la table **commandes** dans laquelle on ajoute la colonne **ville**; la ville où est situé le fournisseur.

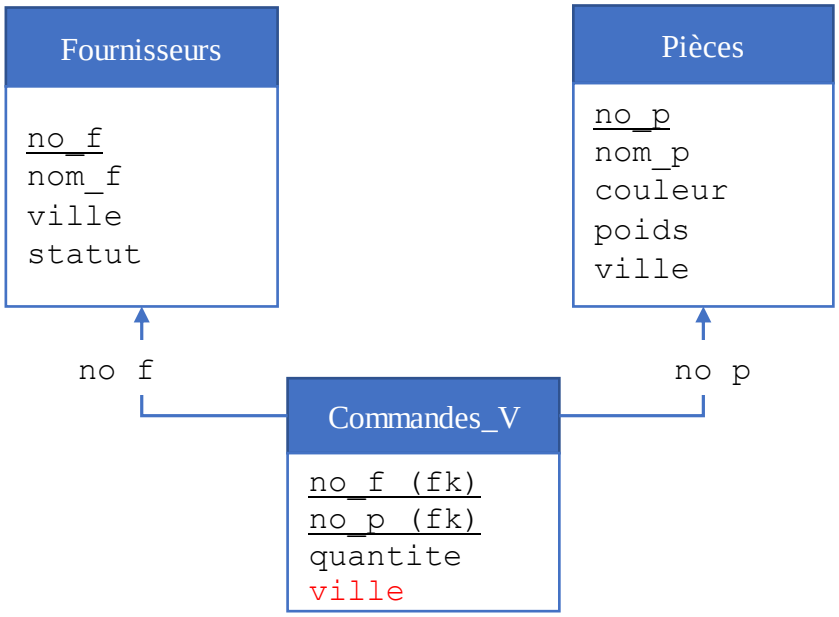


Figure 1: Nouvelle table commandes_v avec la ville du fournisseur

Tableau 1:Exemple de données pour la table commandes_v

commandes_v			
no_f	**ville**	**no_p**	**quantite**
F0001	Londres	P0001	100
F0001	Londres	P0002	100
F0002	Paris	P0001	200
F0002	Paris	P0002	200
F0003	Paris	P0002	300
F0004	Londres	P0002	400
F0004	Londres	P0004	400
F0004	Londres	P0005	400

Dans la table **commandes_V**, il existe une dépendance fonctionnelle entre les colonnes {**no_f}** → **{ville}** parce que pour un **no_f** donné correspond toujours la même valeur pour la colonne **ville**. On peut dire que la colonne **ville** est fonctionnellement dépendante du numéro de fournisseur **no_f**.

Lorsque l'on détermine les dépendances fonctionnelles dans une table, il faut toujours considérer l'ensemble des valeurs possibles que cette table peut contenir. Dans le cas du **no_f** et de la **ville**, cette condition est respectée puisqu'un **fournisseur** est situé précisément dans une seule **ville.**

Cependant, la dépendance fonctionnelle **{no_f}** → **{quantite}**, même si elle existe avec les données actuelles dans la table, n'est pas vrai dans tous les cas. Même raisonnement pour **{quantite}** → **{no_f}.**

Voici l'ensemble des **dépendances fonctionnelles** pour la table **commandes_v**.

- {no_f, no_p} → {quantite}
- {no_f, no_p} → {ville, quantite}
- {no_f, no_p} → {ville}
- {no_f} → {ville}

Notez que la DF **{no_f, no_p}** → **{ville}** n'est pas irréductible. En effet la colonne **no_p** n'est pas nécessaire en ce qui concerne la dépendance fonctionnelle. Une DF est irréductible si la partie de gauche de la DF n'a pas de colonnes superflues.

Notez aussi que toutes les DF ont comme opérandes de gauche la clé primaire, à part la DF **{no_f}** → **{ville}**.

L'objectif est d'avoir uniquement des DF irréductibles dont l'opérande de gauche est la clé primaire. Si on revient à la version non modifiée de la base de données **fournisseurs, commandes, pieces,** les seules dépendances fonctionnelles ont toutes comme opérande de gauche la clé primaire et comme opérande de droite l'ensemble des colonnes, restantes, qui définissent l'entité.

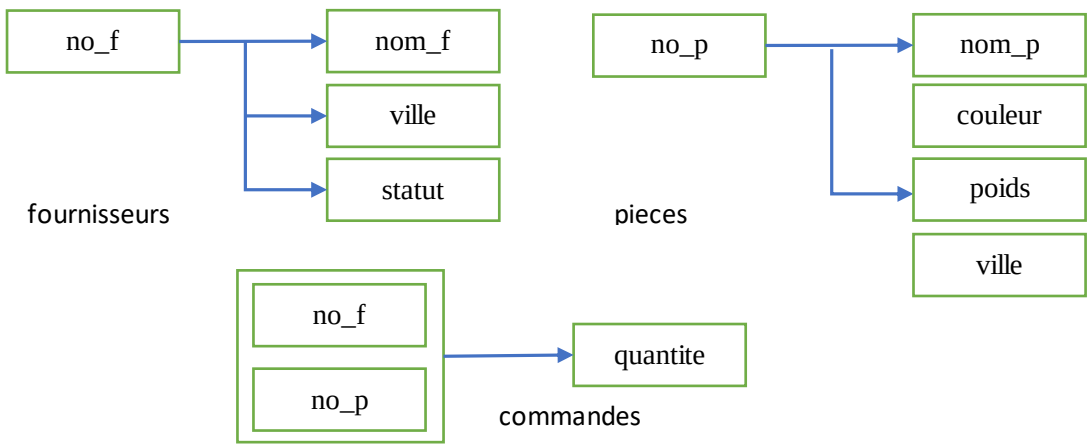


Figure 2:Dépendances fonctionnelles de la BD fournisseurs, commandes, pieces

Formes normales

Plusieurs formes normales ont été définies. Comme il apparait sur la figure 1, toutes les tables définies dans un SGBDR sont dites en première forme normale (1FN). Cette première FN est intrinsèque au modèle relationnel. Certaines tables en 1FN sont aussi en 2FN et certaines tables en 2FN sont aussi en 3FN.

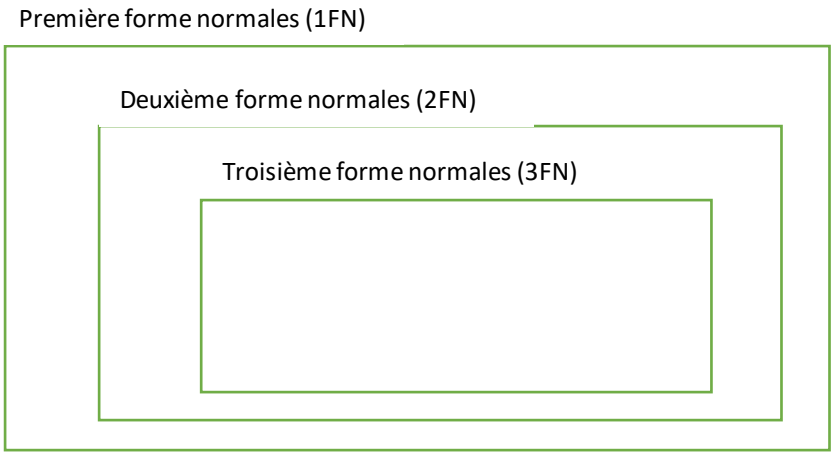


Figure 3: Formes normales

Première forme normale (1FN)

Une table est en 1FN si et seulement si, à l'intersection d'une ligne et d'une colonne il y a uniquement une valeur, jamais une collection de valeurs.

Deuxième forme normale (2FN)

Une table est en 2FN si et seulement si elle est en 1FN et que toutes les colonnes qui ne participent pas dans la clé primaire sont dépendantes (de manière irréductible) sur la clé primaire.

Troisième forme normale (3FN)

Une table est en 3FN si et seulement elle est en 2FN et que les colonnes qui ne participent pas la clé primaire sont mutuellement indépendantes.

- Des colonnes sont mutuellement indépendantes s'il n'existe pas de dépendance fonctionnelle entre eux.

Une table est en 3FN si et seulement si, à tout moment, chaque ligne de la table est constituée d'une clé primaire et d'un ensemble de colonnes, mutuellement indépendantes, qui définissent l'entité.

Une table est en 3FN si les seules dépendances fonctionnelles présentent dans la table sont basées sur la clé primaire.

Prenez, pour exemple, la table **Pieces** à la figure 1. Cette table est en **3FN**. En effet, les colonnes **nom_p, couleur, poids, ville** sont toutes mutuellement indépendantes (on peut modifier la valeur de n'importe laquelle de ces colonnes sans qu'il y ait d'impact sur les autres). De plus, toutes ces colonnes sont dépendantes sur la clé primaire **no_p** (qui est irréductible).

Procédure de normalisation en 3FN

Pour illustrer la décomposition d'une table, en 1FN, en plusieurs tables en 3FN, la table **commandes** a été modifiée pour fusionner l'information des tables **Fournisseurs** et **Commandes.**

Pour les besoins de l'exemple, la DF **ville → statut** a été introduite. Donc, étant donnée une **ville** on peut déterminer le **statut** du fournisseur; le **statut** est dépendant de la ville.

Voici à quoi pourraient ressembler les données de la table **four_comm**.

Tableau 2:Exemple de données pour la table four_comm qui fusionne les tables fournisseurs et commandes.

comm_four				
no_f	**statut**	**ville**	**no_p**	**quantite**
F0001	20	Londres	P0001	300
F0001	20	Londres	P0002	200
F0001	20	Londres	P0003	400
F0001	20	Londres	P0004	200
F0001	20	Londres	P0005	100
F0001	20	Londres	P0006	100
F0002	10	Paris	P0001	300
F0002	10	Paris	P0002	400
F0003	10	Paris	P0002	200
F0004	20	Londres	P0002	200
F0004	20	Londres	P0004	300
F0004	20	Londres	P0005	400

Voici les dépendances fonctionnelles pour la table **four_comm**. Remarquez la complexité des DF par rapport à la figure 2.

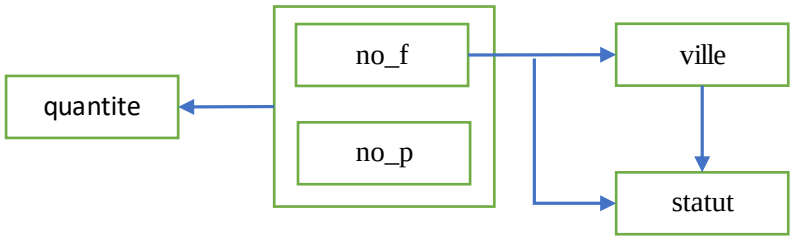


Figure 4: Dépendances fonctionnelles dans la table four_comm

La table **four_comm** ne satisfait pas la définition de la 3FN et de la 2FN pour les raisons suivantes :

- Les DF ne sont pas toutes basées (de façon irréductible) sur la clé primaire.
- Les colonnes qui ne composent pas la clé primaire ne sont pas mutuellement indépendantes (**statut** est FD de **ville**).

Le problème avec la table **four_comm** concerne toutes les flèches qui ne partent pas de la clé primaire **{no_f, no_p}**.

Uniquement à regarder les données de la table **four_comm** on perçoit une redondance. On remarque aussi que l'information contenue dans la table caractérise plus d'une entité. On arrive rapidement à la conclusion que le problème avec la table **four_comm** est qu'elle contient **trop d'information**. Le fait de mélanger l'information des **fournisseurs** avec l'information des **commandes** a pour conséquence que si une ligne est supprimée, l'information des deux entités est supprimée en même temps. De plus, le fait de supprimer la dernière commande chez un fournisseur efface toute trace de ce fournisseur. Finalement, la mise à jour de l'information d'un fournisseur devient compliquée.

La solution aux problèmes énumérés ci-haut est de séparer l'information de chaque entité dans des tables différentes.

La façon de procéder est de sortir de la table les dépendances fonctionnelles qui ne sont pas basées sur la clé primaire. Intuitivement, on peut dire que nous devons sortir de la table **four_comm** l'information qui ne participe à définir une **commande**.

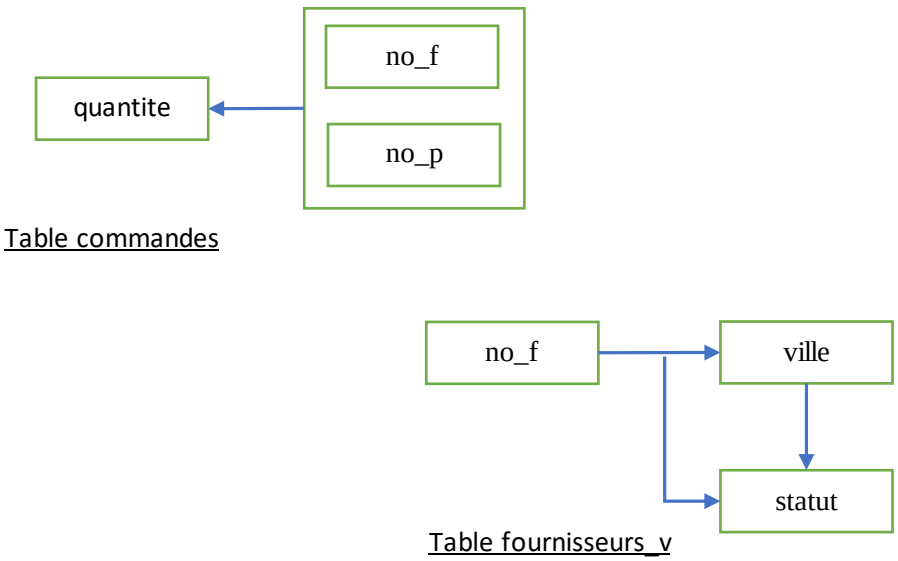


Figure 5: Séparer le DF qui ne sont pas basées sur la clé primaire.

Tableau 3: Données de la table commandes après avoir sotie l'information des fournisseurs

commandes		
no_f	**no_p**	**quantite**
F0001	P0001	300
F0001	P0002	200
F0001	P0003	400
F0001	P0004	200
F0001	P0005	100
F0001	P0006	100
F0002	P0001	300
F0002	P0002	400
F0003	P0002	200
F0004	P0002	200
F0004	P0004	300
F0004	P0005	400

Tableau 4: Nouvelle table fournisseurV résultant de la décomposition de la table four_comm

Fournisseur_V		
no_f	**statut**	**ville**
F0001	20	Londres
F0002	10	Paris
F0003	10	Paris
F0004	20	Londres
F0005	30	Montréal

Avec la décomposition de la table **four_comm** en deux tables (**commandes** et **fournisseurs_v**) nous avons fait en sorte que toutes les colonnes qui ne participent pas à clé primaire sont dépendantes de la clé primaire.

- Dans le cas de la table **commandes**, la seule colonne (**quantite**) qui de participe pas à clé primaire est FD de la clé **{no_f, no_p}** et cette colonne contribue à la définition d'une commande.
- Dans le cas de la table **fournisseurs_v**, les colonnes **statut** et **ville** sont toutes les deux dépendantes de la clé primaire **{no_f}** et ces colonnes caractérisent un fournisseur.

Selon la définition de la 1FN et 2FN, on peut dire que les tables **commandes** et **fournisseurs_v** sont en 1FN et aussi en 2FN.

Selon la définition de la 3FN, on peut dire que la table **commandes** est aussi en 3FN. En effet, la seule colonne restante qui ne participe pas dans la clé primaire ne dépend, assurément, de rien d'autre que la clé primaire.

Il est important de noter qu'aucune perte d'information n'a été occasionnée par la décomposition de la table **four_comm** en deux tables. En effet, il est toujours possible de revenir à la table originale en faisant une jointure des tables **commandes** et **fournisseur_v** basé sur les clés primaire et étrangère.

Le processus de normalisation vers la 3FN n'est pas terminé pour la table **fournisseurs_v**. Si on regarde les dépendances fonctionnelles de cette table on s'aperçoit qu'elles ne sont pas toutes basées sur la clé primaire. Il faut se rappeler que la DF **ville → statut** qui a été mentionné au début de la section. On doit donc comprendre que le **statut** accordé au **fournisseur** est dépendant de la **ville** dans laquelle il est situé. La clé primaire **no_f** détermine la **ville** et c'est la **ville** qui détermine le **statut** qu'aura le **fournisseur**.

Il est vrai que fonctionnellement le **statut** est FD du **no_f**, mais cela se fait à travers la colonne ville.

Selon la définition de la 3FN, les colonnes qui ne participent pas dans la clé primaire ne sont pas mutuellement indépendantes. On ne peut pas changer le statut sans avoir un effet sur la ville et vice versa.

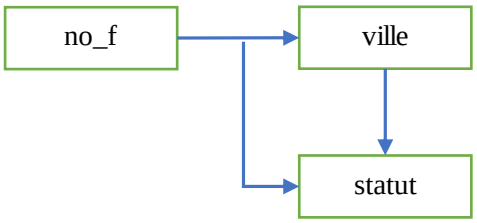


Figure 6: Dépendances fonctionnelles de la table fournisseurs_v

La façon de procéder est de sortir de la table les dépendances fonctionnelles qui ne sont pas basées sur la clé primaire. Intuitivement, on peut dire que nous devons sortir de la table **fournisseurs_v** l'information qui ne participe à définir l'entité identifiée par la clé **no_f**.

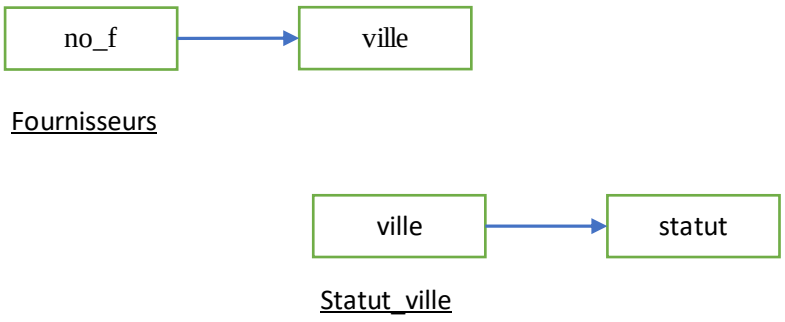


Figure 7: Décomposition de la table fournisseur_v en deux tables founisseurs et statut_ville

Selon la définition de la 3FN, on peut dire que la table fournisseurs et statut_ville sont en 3FN. Pour les deux tables, les seules DF sont basées sur la clé primaire. De plus, les colonnes qui ne participent à la clé primaire sont mutuellement indépendantes.

Voici les tables fournisseurs et statut_ville résultant de la décomposition de la table fournisseur_v.

Encore une fois, notez qu'aucune perte d'information n'a été occasionnée par la décomposition de la table **fournisseurs_v** en deux tables. En effet, il est toujours possible de revenir à la table originale en faisant une jointure des tables **fournisseurs** et **statut_ville** basée sur les clés primaire et étrangère.

Il maintenant possible de modifier le statut associé à une ville de façon indépendante. On peut même conserver le statut de ville pour lesquelles nous n'avons, encore, aucun fournisseur. Finalement, la maintenance du statut des villes est simplifiée et moins sujette à briser l'intégrité des données.

fournisseurs	
no_f	**ville**
F0001	Londres
F0002	Paris
F0003	Paris
F0004	Londres
F0005	Montréal

Statut_ville	
ville	**statut**
Londres	20
Paris	10
Montréal	30
New York	15

A retenir absolument

Après la normalisation, il faudra vérifier :

- Tous les attributs **non calculés** sont présents dans des tables.
- Aucun attribut calculé n'est présent dans une table
- Aucun attribut n'est redondant, sauf la clé primaire qui devient clé étrangère dans d'autres tables.
- Toutes les tables ont une clé primaire.
- Tous les attributs sont élémentaires (non décomposable ou se traitent comme un tout, exemple l'attribut adresse peut-être décomposable).
- Toutes les tables sont au moins en 3FN.

La normalisation a pour buts de garantir :

- La non redondance des données-→1FN
- L'intégrité des données-→2FN,3FN
- La facilité de mise à jour→3FN

Une base de données est normalisée si et seulement si elle est au moins en 3FN

Énoncé 1 : La 1FN : (La clé)

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 table qui va contenir ce groupe de données.

Énoncé 2 : La 2FN (Toute la clé)

Une table 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

Énoncé 3 : La 3FN (Rien que la clé)

Une 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.

Représentation (modélisation) du modèle relationnel

Définition, le modèle de 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 tables, les types de données, les relations, et les contraintes qui définissent le gabarit de la base de données.

Tous les SGBDs offre des outils qui permettent de modéliser la base de données. Certains outils permettent même de générer le code SQL issu du modèle de données.

Définition, cardinalité

Une cardinalité est le nombre de fois minimum et maximum qu'une occurrence d'une table participe à une relation.

Attention : 

Le nombre minimum est tout le temps égal à 0 ou 1.

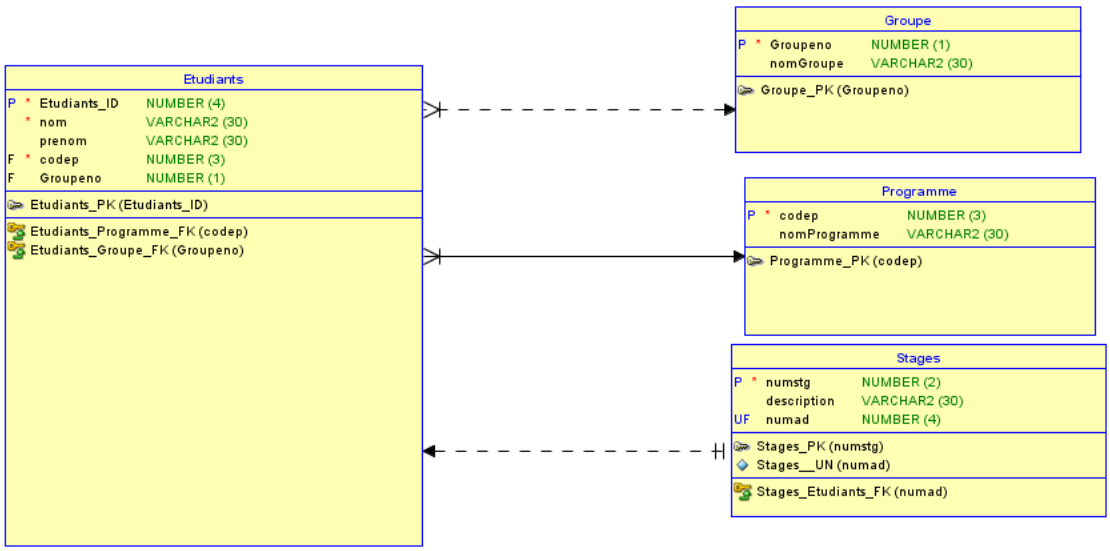
Lorsque la clé primaire migre dans une table pour devenir une clé étrangère alors on se pose la question suivante : La clé étrangère est-elle Obligatoire (not null), si OUI alors la cardinalisé est 1 si la clé étrangère est optionnelle alors la cardinalité est 0.

Attention : 

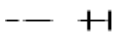
Le nombre maximum est tout le temps égal à 1 ou N.

Lorsque la clé primaire migre dans une table pour devenir une clé étrangère alors on se pose la question suivante : La clé étrangère dit-elle avoir plusieurs valeurs ? si oui alors la cardinalité est N sinon cela veut dire que la clé étrangère est UNIQUE et donc la valeur de la cardinalité est 1

Représentation et exemple



 La fourche indique la multiplicité. Dans notre exemple cela veut dire que le codep dans la table etudiants n'est pas unique. Il peut se répéter plusieurs fois. Ce qui est normal car plusieurs étudiants peuvent être dans le même programme. Comme vous, vous êtes une centaine à être dans le code 420.

 les deux barres verticales indique l'unicité. Dans notre exemple cela veut dire que le numéro de l'étudiant dans stage a la contrainte **UNIQUE** (il est précédé) de la lettre U. Ce qui veux dire que le stage est affecté à un seul étudiant.

 le trait plein indique l'obligation. Dans notre cas cela veut dire que le Codep dans Etudiants est OBLIGATOIRE, ce qui veut dire que tous les étudiants sont OBLIGÉS d'être dans un programme. CodeP dans étudiant est précédé du symbole * qui veut dire Obligatoire. Donc cet attribut a la contrainte de **NOT NULL**

 le trait en pointillé indique l'OPTIONNEL dans notre cas cela veut dire Groupeno dans la table Etudiants n'est obligatoire (il n' a pas la contrainte de not null).l'étudiant n'est pas obligé d'être dans un groupe.