180. Normalisation et formes normales

Table des matières

1 Préambule

La normalisation est un processus lié à la structuration des bases de données relationnelles. Selon le principe de « relation » du modèle relationnel de Codd, l'ensemble des données utiles à une fonctionnalité de système d'information informatisé (SII) pourrait être stocké dans une seule table[1], un peu comme si nous avions une et une seule feuille Excel qui contiendrait l'ensemble des données liées aux enfants de notre crèche[2].

Naturellement, cette manière de faire induirait une redondance de données ; pour notre crèche, nous aurions les données d'une éducatrice reproduites pour chaque enfant dont elle a la charge.

Enfants

	Enf_Prenom	Enf_Sexe	Enf_DateNaiss	Enf_Allergies	Edu_Mnemo	Edu_Nom	Edu_LieuxdeVie
1	Pierre	Garçon	07.05.2005	lactose	CR	Rouge	***Montagne***
2	Marie	Fille	13.12.2003		JB	Bleu	***Océan***
3	Claude	Garçon	04.01.2004	gluten	JB	Bleu	***Océan***
4	Jeannie	Fille	22.10.2004	lactose,gluten	CR	Rouge	***Montagne***
5	Nathan	Garçon	05.11.2003		AO	Orange	Lac, ***Jura***
6	Cloée	Fille	07.04.2005		CR	Rouge	***Montagne***

Le lieu de vie principal de l'éducatrice est mis en italique et gras

Si une éducatrice se voit affecter un autre de lieu de vie ou si elle change de nom, suite à un mariage, les nouvelles données devraient être reportées dans chaque ligne de la table où l'éducatrice apparaît ; par exemple, si l'éducatrice de mnémonique CR change de lieu de vie et passe de Montagne à Mer, il faudrait modifier les lignes 1, 4 et 6.

La redondance de données dans une table est une source d'erreurs lors des opérations de lecture et d'écriture. En présence d'une redondance, nous n'avons aucune certitude en lisant une valeur de colonne sur une ligne qu'une valeur différente n'a pas été enregistrée sur une autre ligne.

Enfants

	Enf_Prenom	Enf_Sexe	Enf_DateNaiss	Enf_Allergies	Edu_Mnemo	Edu_Nom	Edu_LieuxdeVie
1	Pierre	Garçon	07.05.2005	lactose	CR	Rouge	***Montagne***
2	Marie	Fille	13.12.2003		JB	Bleu	***Océan***
3	Claude	Garçon	04.01.2004	gluten	JB	Bleu	***Océan***
4	Jeannie	Fille	22.10.2004	lactose,gluten	CR	Rouge	***Vallée***
5	Nathan	Garçon	05.11.2003		AO	Orange	Lac, ***Jura***
6	Cloée	Fille	07.04.2005		CR	Rouge	***Montagne***

Le lieu de vie principal de l'éducatrice est mis en italique et gras

[1] Le terme précis selon la terminologie initiale proposée par Codd, est relation. Toutefois, nous utilisons le terme de table pour éviter toute confusion avec la notion de relation pour mentionner un lien entre tables.
[Plus de détails dans le chapitre de modélisation logique des données]
[2] Voir notre exemple de gestion de crèche dans les chapitres précédents.

Par exemple, en ligne 1, nous lisons que l'éducatrice de mnémonique CR exerce son activité dans le lieu de vie Montagne ; si en ligne 4 et pour cette même éducatrice, nous trouvons le lieu de vie Vallée alors nous sommes face à un dilemme ; quelle est la donnée correcte, Montagne ou Vallée?

L'informatique et la modélisation ne sauraient résoudre ce dilemme de valeurs contradictoires, la seule chose que nous puissions faire est d'éviter de se trouver face à ce dilemme. Pour ce faire, la normalisation des données propose des règles, les formes normales (NF), pour transformer une table (relation selon Codd) qui contiendrait des données redondantes en un « réseau » de tables exempt de redondances ; les tables sans redondances d'un tel réseau sont dites « normalisées ».

Pour notre cours, nous appliquerons les règles de normalisation déjà en phase de conception et ferons en sorte que nos modèles conceptuels de données soient d'ores et déjà normalisés ou en 3ème forme normale (ou mieux en forme normale de Boyce-Codd) avant leur transformation en un modèle logique de données relationnel.

2 Introduction

La **normalisation** est un processus qui permet de s'assurer de la *relationnalité*[3] de la structure des tables d'un schéma de base de données et de l'absence de redondances. La normalisation est basée sur le concept de dépendance fonctionnelle entre colonnes de tables.

Les **formes normales** sont un formalisme qui permet de vérifier l'atomicité des données (élémentarité) d'une part et l'absence de redondances d'autre part.

- La première forme normale (1NF) permet de vérifier qu'une table puisse être considérée comme étant *relationnelle*, c'est-à-dire formée de colonnes contenant toutes des données atomiques.
- La deuxième forme normale (2NF) permet de valider le choix de la clé primaire en contrôlant les dépendances fonctionnelles entre clé primaire et colonne non clé primaire.
- La troisième forme normale (3NF) permet de vérifier l'absence de redondances dans la table en contrôlant les dépendances fonctionnelles entre colonnes non clé primaire.

Une table est dite **normalisée** si elle est en troisième forme normale (3NF).

Préalablement à la première forme normale (1NF) une table (relation) doit être dotée d'une clé primaire ; certains auteurs parlent de 0NF pour décrire une table qui est dotée d'une clé primaire.
Avec notre approche, cette pseudo-règle (0NF) est assumée en modélisation conceptuelle déjà ou, exceptionnellement au plus tard, lors de la transformation du MCD en MLD.

[3] Relationnalité, toujours dans le sens de tabulaire donné par Codd

3 Dépendance fonctionnelle

Une colonne (ou groupe de colonnes) B d'une table R est dit en dépendance fonctionnelle d'une autre colonne (ou groupe de colonnes) A de R, si, à tout instant chaque valeur a_i de A n'a qu'une valeur associée b_i de B. On note:

$$A \rightarrow B$$

R

	A	B	C
Tuple i	ai	bi	ci

Dans l'exemple ci-dessous toutes les colonnes non clé primaire ou secondaire unique et non nulle «UID» sont en dépendance de la clé primaire.

$$\text{Numero} \rightarrow \text{Nom}$$
$$\text{Numero} \rightarrow \text{Prenom}$$

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom
1	1000	Bleu	Claude
2	1004	Orange	Jeannie
3	1005	Marron	Marc
4	1008	Noir	Claude

Remarques importantes :

1. Le concept de dépendance fonctionnelle s'applique aux colonnes indépendamment des concepts de clés (primaires, secondaires unique et non nulles «UID» ou étrangères).

2. La clé primaire est toujours source de dépendances fonctionnelles pour toutes les colonnes non clé primaire et «UID» de la table.

3. La clé secondaire unique et non nulle est toujours source de dépendances fonctionnelles pour toutes les colonnes non clé primaire et «UID» de la table.

4. Une double dépendance fonctionnelle existe toujours entre clé primaire et clé secondaire unique et non nulle «UID».
   [Traité en détail au chapitre suivant]

3.1 Dépendance fonctionnelle élémentaire pleine

Une dépendance fonctionnelle A → B est dite **élémentaire pleine** s'il n'existe pas A' ⊂ A tel que A' → B.

R

	A		B	C
	A'	A"		
Tuple i	*ai'*	ai"	bi	ci

Dans l'exemple ci-dessous toutes les colonnes non clé primaire ou secondaire unique et non nulle «UID» sont en dépendance fonctionnelle élémentaire pleine de la clé primaire.

Cat_Numero, NumeroDep → Nom
Cat_Numero, NumeroDep → Prenom

Concurrents

«PFK» Cat_Numero	«PK» NumeroDep	«UID» Dossard	Nom	Prenom
1	1	1000	Bleu	Claude
2	1	1004	Orange	Jeannie
1	2	1005	Marron	Marc
2	2	1005	Azur	René
2	3	1008	Noir	Claude
1	3	1008	Vert	Daniel

Remarque: Concaténée à Cat_Numero, Dossard est une clé secondaire «UID»

Cat_Numero est une clé étrangère ; elle dépend fonctionnellement de la clé primaire de la table de référence Categories.
NumeroDep est un compteur incrémenté de 1 à n pour chaque valeur de Cat_Numero.

Cat_Numero et NumeroDep forment ensemble la clé primaire ; la connaissance d'une seule de ces deux valeurs ne suffit pas à identifier un enregistrement ou une ligne. Par exemple, la connaissance de la valeur 1 de la catégorie seule ne permet pas de déduire le nom d'un concurrent ; nous trouvons 3 concurrents : Bleu, Marron et Vert ; en ajoutant une valeur de NumeroDep, par exemple 2, nous trouvons bien un et un seul concurrent, en l'occurrence Marron.

3.2 Dépendance fonctionnelle directe

Une dépendance fonctionnelle A → B est dite **directe** s'il n'existe pas d'attribut C tel que A → C et C → B (absence de transitivité).

R

	A	B	C
Tuple i	ai	bi	ci

Dans l'exemple ci-dessous toutes les colonnes non clé primaire ou secondaire unique et non nulle «UID» sont en dépendance fonctionnelle directe de la clé primaire.

Numero → Nom
Numero → Prenom

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom
1	1000	Bleu	Claude
2	1004	Orange	Jeannie
3	1005	Marron	Marc
4	1008	Noir	Claude

Nous voyons, par exemple, que la connaissance du prénom Claude ne permet pas de déterminer Nom qui peut valoir Bleu ou Noir ; donc, Nom est bien en dépendance fonctionnelle directe de Numero car il ne peut être déduit de Prenom:

Numero → Nom

Prenom ↛ Nom

4 Clé secondaire unique et non nulle « UID »

4.1 Problématique

Selon les préceptes que nous avons suivis en modélisation conceptuelle et logique :

- toutes nos tables sont dotées d'une clé primaire dont une des caractéristiques essentielles est qu'elle ne change jamais de valeur car elle est utilisée pour établir les (relations de) dépendances entre tables;
- sauf quelques cas particuliers, tels que les personnes, une clé secondaire unique et non nulle «UID» est ajoutée à nos tables pour servir d'identifiant naturel aux utilisateurs. Contrairement aux clés primaires, une clé secondaire unique et non nulle est porteuse d'information; de ce fait, elle doit pouvoir changer de valeur pour refléter tout changement dans l'information qu'elle véhicule.

En résumé, **une clé primaire** est nécessaire pour **construire robustement** un schéma de base de données relationnel tandis qu'une **clé secondaire unique et non nulle** est nécessaire pour faciliter la **manipulation de données par les utilisateurs**.

De par la nécessité évoquée ci-dessus, toutes les tables porteuses de clés secondaires uniques et non nulles «UID», sont automatiquement affublées d'une dépendance fonctionnelle circulaire.

Colonne(s) de clé primaire → Colonne(s) de clé secondaire unique et non nulle (I)
et
Colonne(s) de clé secondaire unique et non nulle → Colonne(s) de clé primaire (II)

La dépendance fonctionnelle (I) est utile au système de gestion de bases de données pour fournir de l'information significative (clé secondaire) à l'utilisateur à partir de données techniques (clé primaire).
La dépendance fonctionnelle (II) est utile au système de gestion de bases de données pour transformer l'information fournie par l'utilisateur (clé secondaire) en données techniques (clé primaire).

4.2 Exemple d'une table indépendante

Pour une table indépendante R dont la colonne de clé secondaire se nomme Code ; nous aurions la double dépendance :

Numero → Code (I)
et
Code → Numero (II)

R

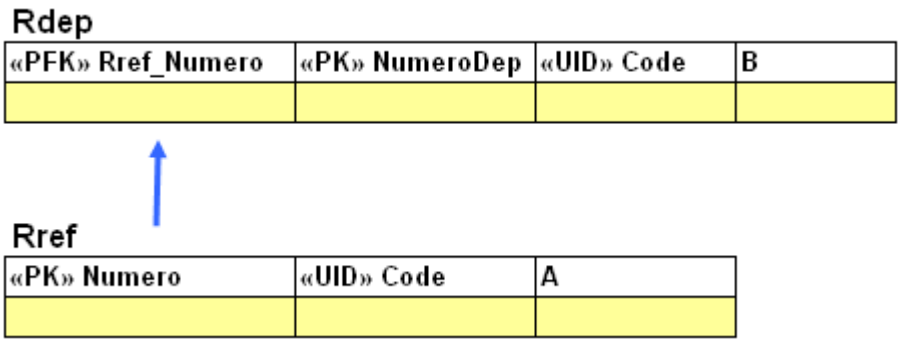
«PK» Numero	«UID» Code	A	B

4.3 Tables dépendantes

Dans le cas de tables dépendantes, la double dépendance doit impérativement se réaliser en prenant en compte les colonnes de clés étrangères identifiantes qui seront concaténées d'une part à la colonne de clé primaire NumeroDep et d'autre part à la colonne de clé secondaire.

Par exemple, pour une table dépendante Rdep dont la colonne de clé secondaire se nomme Code et la colonne de clé étrangère identifiante se nomme Rref _Numero ; nous aurions la double dépendance :

Rref_Numero, NumeroDep → Rref_Numero, Code (I)
et
Rref_Numero, Code → Rref_Numero, NumeroDep (II)



Pour satisfaire aux besoins d'informations significatives pour l'utilisateur, la dépendance fonctionnelle (II) doit être basée uniquement sur des clés secondaires.

Nous avons les dépendances suivantes dans et entre nos deux tables Rref et Rdep

$$\text{Rref.Numero} \rightarrow \text{Rdep.Rref\_Numero (III)}$$
$$\text{Rref.Code} \rightarrow \text{Rref.Numero (IV)}$$

En combinant par transitivité (II), (III) et (IV), nous obtenons la dépendance fonctionnelle (II) basée uniquement sur des clés secondaires:

$$\text{Rref.Code, Rdep.Code} \rightarrow \text{Rdep.Rref\_Numero, Rdep.NumeroDep (IIa)}$$

Remarque importante :
Pour les tables dépendantes, chacune des deux dépendances fonctionnelles doit être élémentaire pleine. La connaissance de la seule valeur de la colonne de clé secondaire propre à la table dépendante (Rdep.Code pour notre exemple) ne doit pas suffire à identifier un tuple.

4.4 Conclusion

La dépendance fonctionnelle circulaire entre clé primaire et clé secondaire n'est pas une anomalie et ne doit pas être remise en cause par le processus de normalisation.

Le processus de normalisation doit se faire en s'appuyant exclusivement sur les colonnes de clé primaire lorsqu'il s'agit d'identifier des tuples.

Les colonnes de clés secondaires «UID» ne sont pas impliquées par le processus de normalisation.

5 1ère forme normale (1NF)

> Une table est en première forme normale (1NF)
> si elle est dotée d'une clé primaire (0NF) et que
> **toutes les colonnes contiennent des valeurs atomiques.**

A notre sens, deux erreurs distinctes peuvent être sources des principales causes de non-respect de la 1ère forme normale :

1. une colonne est elle-même une relation.
2. une colonne contient un groupe répétitif de données.

Si une colonne contient des données qui ne sont pas atomiques, le contenu des colonnes doit être « interprété » avec tous les risques d'erreurs induits par les modalités d'interprétation.

5.1 ¬1NF - Colonne formée d'une relation[4]

5.1.1 Règle de normalisation

Lorsqu'une table comporte une colonne qui est elle-même une relation, il faut transformer la colonne en une nouvelle table et établir une relation entre les deux tables. Cette relation aura comme source la nouvelle table et comme destination la table initiale dans laquelle la colonne ¬1NF a été remplacée par une clé étrangère.

[4] Le symbole ¬ représente la négation, en l'occurrence : n'est pas en 1NF

5.1.2 Exemple

Dans la table Concurrents ci-dessous, la colonne AgeEtCat contient l'âge du concurrent mais aussi la catégorie dans laquelle il concourt.

Nous voyons avec cet exemple, que le contenu de la colonne AgeEtCat doit être interprété pour ressortir d'une part l'âge et d'autre part la catégorie. A priori, la première partie contient l'âge et la deuxième partie, marquée par la virgule comme séparateur, contient la catégorie. Rien ne nous garantit que d'autres lignes ou tuples ne contiendront pas ces données inversées et/ou différenciées par un autre caractère que la virgule.
Par ailleurs, si nous voulions compter le nombre de concurrents d'un âge donné ou d'une catégorie particulière, nous ne pouvons pas faire une requête SQL sur une colonne qui contient directement cette donnée.

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom	AgeEtCat
1	1000	Bleu	Claude	12 , A
2	1004	Orange	Jeannie	13 , B
3	1005	Marron	Marc	12 , A
4	1008	Noir	Claude	14 , B

Pour être en première forme normale (1NF), il faudrait séparer ces deux données en deux colonnes distinctes.

> Nous pourrions le faire mais, une règle de gestion stipule que la catégorie dépend de l'âge du concurrent.

Dès lors, nous créons une nouvelle table pour enregistrer cette dépendance fonctionnelle entre âge et catégorie. Ensuite, nous mettons cette table en relation avec la table Concurrents comme stipulé par la règle de normalisation.

Ages

«PK» Numero	«UID» Valeur	Categorie
1	12	A
2	13	B
3	14	B

Concurrents

«PK» Numero	Age_Numero	«UID» Dossard	Nom	Prenom
1	1	1000	Bleu	Claude
2	2	1004	Orange	Jeannie
3	1	1005	Marron	Marc
4	3	1008	Noir	Claude

5.2 ¬1NF - Colonne contenant un groupe répétitif¨

5.2.1 Règle de normalisation

- Lorsqu'une table comporte une colonne qui est un groupe répétitif de données sans existence propre, il faut transformer la colonne en une nouvelle table dépendante ; une relation identifiante sera établie entre les deux tables.

- Lorsqu'une table (t_{init}) comporte une colonne qui est un groupe répétitif de données pouvant avoir une existence propre, il faut transformer la colonne en une nouvelle table (t_{ref}); ensuite, il faut créer une table dépendante ou associative (t_{lien}) pour lier les deux tables t_{init} et t_{ref}.

5.2.2 Exemple de données répétitives sans existence propre

Dans la table Concurrents ci-dessous, la colonne Sauts contient la valeur des deux sauts qu'effectuent chacun des concurrents.

Nous voyons avec cet exemple, que le contenu de la colonne Sauts doit être interprété pour ressortir les valeurs de chacun des sauts. A priori, la virgule est utilisée comme séparateur. Rien ne nous garantit que d'autres lignes ou tuples ne contiendront pas un autre caractère que la virgule comme séparateur.
Par ailleurs, si nous voulions rechercher le saut le plus long réalisé par les concurrents ou calculer la moyenne des sauts de tous les candidats, nous ne pouvons pas faire une requête sur une colonne qui contient directement la valeur de chaque saut.

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom	Sauts
1	1000	Bleu	Claude	3.28 , 3.07
2	1004	Orange	Jeannie	0 , 3.00
3	1005	Marron	Marc	4.10 , 0
4	1008	Noir	Claude	2.35 , 2.50

Pour être en première forme normale (1NF), il faudrait séparer ces deux données en deux colonnes distinctes : Saut1 et Saut2. Nous pourrions le faire mais, nous nous l'interdisons pour deux raisons essentielles:

1. certains traitements doivent pouvoir être appliqués indifféremment à chacune des colonnes ; avec les colonnes Saut1 et Saut2, nous ne pouvons pas faire, sans artifices, une requête SQL qui ressort le saut le plus long ou une requête SQL qui calcule la moyenne des sauts réalisés.
2. dans de nombreuses situations de gestion, le nombre d'éléments répétitifs ne saurait être figé ; dans notre cas, il se pourrait qu'il faille enregistrer un 3ème saut, ce qui impliquerait de créer une nouvelle colonne.

En conformité avec la règle de normalisation énoncée, nous créons une table dépendante Sauts.
La clé primaire de cette table est formée de la colonne de clé étrangère Conc_Numero référant la table identifiante et de la colonne NumeroDep contenant une valeur incrémentée de 1 à n pour chaque valeur de la colonne Conc_Numero.

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom
1	1000	Bleu	Claude
2	1004	Orange	Jeannie
3	1005	Marron	Marc
4	1008	Noir	Claude

Sauts

«PFK» Conc_Numero	«PK» NumeroDep	Valeur
1	1	3.28
1	2	3.07
2	1	0
2	2	3
3	1	4.1
3	2	0
4	1	2.35
4	2	2.5

5.2.3 Exemple de données répétitives avec existence propre

Dans la table Concurrents ci-dessous, la colonne Equipes contient le nom des équipes auxquelles un concurrent appartient ; un concurrent peut appartenir à plusieurs équipes.

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom	Equipes
1	1000	Bleu	Claude	Carrés
2	1004	Orange	Jeannie	
3	1005	Marron	Marc	Carrés, Cercles
4	1008	Noir	Claude	Triangles

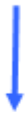
Les problèmes de redondance décrits dans l'exemple précédent se posent aussi pour ce cas; nous pourrions envisager de créer les colonnes Equipe1 & Equipe2 pour être en première forme normale mais, nous nous l'interdisons pour les mêmes raisons que citées précédemment :

- certains traitements doivent pouvoir s'appliquer à l'information « Fait partie d'équipe(s) » d'un concurrent indifféremment de noms spécifiques de colonnes tels que Equipe1 ou Equipe2.
- le nombre d'équipes auxquelles peut appartenir un concurrent n'est pas forcément arrêté à deux.

En conformité avec la règle de normalisation énoncée, nous créons la table Equipes ; cette table répond au besoin d'existence propre des équipes auxquelles peuvent appartenir les concurrents.
La colonne Equipes de la table initiale Concurrents devient la clé secondaire unique et non nulle «UID» de la table Equipes.
Ensuite, nous créons la table associative Equipiers ; la clé primaire de cette table est formée des deux clés étrangères référant les tables Concurrents et Equipes.

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom
1	1000	Bleu	Claude
2	1004	Orange	Jeannie
3	1005	Marron	Marc
4	1008	Noir	Claude



Equipiers

«PFK» Conc_Numero	«PFK» Equi_Numero
1	3
3	3
3	2
4	1

Equipes

«PK» Numero	«UID» Nom
1	Triangles
2	Cercles
3	Carrés

Remarque : La table Equipiers pourrait être réalisée sous forme d'une table dépendante ; la colonne de clé étrangère Equi_Numero ne serait, alors, plus identifiante.

Equipiers

«PFK» Conc_Numero	«PK» NumeroDep	«FK» Equi_Numero
1	3	3
3	3	3
3	2	2
4	1	1

6 2ème forme normale (2NF)

Une table associative ou dépendante est en deuxième forme normale (2NF) si elle est déjà en 1ère forme normale (1NF) et que **les dépendances fonctionnelles entre la clé primaire et les autres colonnes[5] sont élémentaires et pleines.**

Seules les tables comportant une clé primaire composite, c'est-à-dire composée de plusieurs colonnes, sont concernées.

Remarque importante :

En fonction des règles de modélisation que nous nous sommes fixées dans les chapitres précédents, une clé primaire composite n'existe qu'en présence de tables associatives ou de tables dépendantes.

Les clés primaires composites de tables associatives sont formées des colonnes de clés étrangères de chacune des tables sources de relation.
Les clés primaires composites de tables dépendantes sont formées des colonnes de clés étrangères des tables identifiantes (une seule pour les cas simples) et de leurs propres colonnes de clés primaires que nous nommons NumeroDep.

6.1 Règle de normalisation

Les colonnes non clés qui ne dépendent fonctionnellement que d'une des clés étrangères sont reportées dans la table source de la relation matérialisée par cette clé étrangère.

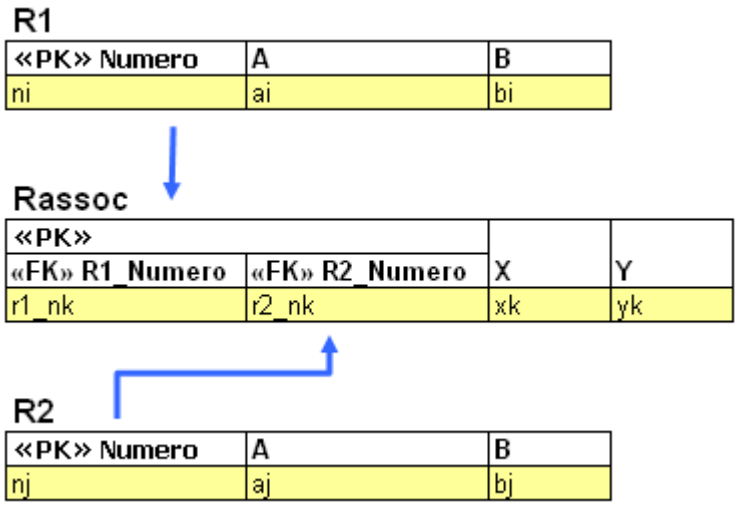
Remarque: Une colonne non clé de table dépendante ne peut en aucun cas dépendre de la seule colonne de clé primaire NumeroDep.
Si tel devait être le cas, le caractère dépendant et les sources de dépendances d'une telle table devraient être repensés.

[5] Pour rappel : hormis les colonnes de clé secondaire unique et non nulle «UID»

6.2 Démonstration

La table associative Rassoc n'est pas en 2ème forme normale (2NF) si les dépendances élémentaires pleines suivantes ne sont pas vérifiées :

$$R1\_Numero\ ,\ R2\_Numero \rightarrow X$$
$$R1\_Numero\ ,\ R2\_Numero \rightarrow Y$$



Prenons le cas de la colonne Y qui transgresse la 2ème forme normale par l'existence de la dépendance fonctionnelle suivante:

$$R2\_Numero \rightarrow Y$$
ou
$$Rassoc.R2\_Numero \rightarrow Rassoc.Y\ (I)$$

Nous avons la dépendance fonctionnelle suivante entre la table source et la table cible de la relation :

$$R2.Numero \rightarrow Rassoc.R2\_Numero\ (II)$$

En combinant par transitivité (I) et (II) nous obtenons :

$$R2.Numero \rightarrow Rassoc.R2\_Numero \rightarrow Rassoc.Y$$

En simplifiant nous obtenons :

$$R2.Numero \rightarrow Rassoc.Y$$

Comme la colonne Y de la table Rassoc est en dépendance fonctionnelle directe de la clé primaire de la table R2 , nous devons rapatrier cette donnée en tant que colonne dans la table R2. En finalité nous obtenons :

R2.Numero → R2.Y (III)
Ou dans le contexte de la table R2
Numero → Y

R1

«PK» Numero	A	B
ni	ai	bi

Rassoc

«PK»		
«FK» R1_Numero	«FK» R2_Numero	X
r1_nk	r2_nk	xk

R2

«PK» Numero	A	B	Y
nj	aj	bj	yj

6.3 Exemple

Pour les colonnes non clé primaire de la table Concurrents ci-après, nous devons vérifier les dépendances fonctionnelles suivantes :

Cat_Numero , NumeroDep → Nom
Cat_Numero , NumeroDep → Prenom
Cat_Numero , NumeroDep → Points

Categories

«PK» Numero	«UID» Code
1	A
2	B

Concurrents

«PFK» Cat_Numero	«PK» NumeroDep	«UID» Dossard	Nom	Prenom	Points
1	1	1000	Bleu	Claude	15
2	1	1004	Orange	Jeannie	20
1	2	1005	Marron	Marc	15
2	2	1005	Azure	René	20
2	3	1008	Noir	Claude	20
1	3	1008	Vert	Daniel	15

La colonne Points représente un capital de points attribué aux concurrents indépendamment des résultats obtenus aux différents concours (sauts et autres).

> Toutefois, nous avons une règle de gestion qui stipule que le nombre de points n'est pas propre à un concurrent mais à la catégorie à laquelle le concurrent appartient.

Dès lors, nous avons la dépendance fonctionnelle suivante:

$$\text{Cat\_Numero} \rightarrow \text{Points}$$

Conformément à la règle de normalisation énoncée, nous rapatrions la colonne Points dans la table Categories source de la relation matérialisée par la colonne de clé étrangère Cat_Numero.

Categories

«PK» Numero	«UID» Code	Points
1	A	15
2	B	20

Concurrents

«PFK» Cat_Numero	«PK» NumeroDep	«UID» Dossard	Nom	Prenom
1	1	1000	Bleu	Claude
2	1	1004	Orange	Jeannie
1	2	1005	Marron	Marc
2	2	1005	Azure	René
2	3	1008	Noir	Claude
1	3	1008	Vert	Daniel

7 3ème forme normale (3NF)

Une table est en troisième forme normale (3NF)
si elle est déjà en 2ème forme normale (2NF) et que
les dépendances fonctionnelles entre la clé primaire et les autres colonnes[6] sont directes.

Dit plus simplement, la troisième forme normale (3NF) postule qu'il ne doit pas y avoir de dépendances fonctionnelles entres colonnes non clé primaire de tables.

Rappel : Une colonne est en dépendance fonctionnelle directe avec la clé primaire si elle ne dépend pas fonctionnellement d'une autre colonne non clé primaire.
[Dépendance fonctionnelle directe]

7.1 Règle de normalisation

Lorsqu'une table comporte une ou plusieurs colonnes qui ne sont pas en dépendance fonctionnelle directe avec la clé primaire, il faut créer une ou plusieurs tables pour enregistrer spécifiquement les dépendances entre colonnes non clé primaire.

Les colonnes de ces nouvelles tables seront issues du transfert de la colonne source et des colonnes cibles de la dépendance fonctionnelle de la table à normaliser.
Pour chaque nouvelle table :

- la colonne source de la dépendance deviendra la clé secondaire unique et non nulle ;
- une colonne Numero, base de clé primaire, sera ajoutée et servira à établir la relation de dépendance fonctionnelle avec la table normalisée.

Cette ou ces nouvelles tables seront sources de relations vers la table initiale qui devient ainsi normalisée. Pour chaque nouvelle table, une colonne de clé étrangère sera ajoutée dans la table normalisée pour servir de cible de relation.

Remarques : Nous postulons que la source de la dépendance fonctionnelle de la table à normaliser est formée d'une seule colonne ; si la source devait être formée de plus d'une colonne, il y aurait lieu de faire une analyse spécifique pour déterminer si la nouvelle table est une table associative ou une table dépendante. Qu'il s'agisse d'une table associative ou d'une table dépendante, il y a lieu de créer en plus la ou les tables sources de la ou des relations identifiantes dont l'une ou l'autre de nos deux tables est la cible.

Plusieurs colonnes peuvent être cibles d'une source identique de dépendance fonctionnelle. Chacune des colonnes est transférée dans la nouvelle table dont la colonne source en est la clé secondaire unique et non nulle.

[6] Pour rappel : hormis les colonnes de clé secondaire unique et non nulle «UID»

7.2 Exemple

Pour les colonnes non clé primaire de la table Concurrents ci-après, nous devons vérifier les dépendances fonctionnelles directes suivantes :

Numero → Nom
Numero → Prenom
Numero → Age
Numero → Categorie

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom	Age	Categorie
1	1000	Bleu	Claude	12	A
2	1004	Orange	Jeannie	13	B
3	1005	Marron	Marc	12	A
4	1008	Noir	Claude	14	B

> Une règle de gestion stipule que la catégorie à laquelle appartient un concurrent dépend de son âge.

Dès lors, nous avons les dépendances fonctionnelles suivantes :

Numero → Age → Categorie

Donc, la colonne Categorie n'est pas en dépendance fonctionnelle directe de la clé

Numero → … → Categorie

Conformément à la règle de normalisation énoncée, nous rapatrions les colonnes source (Age) et cible (Categorie) dans une nouvelle table.
Pour faciliter la lecture du modèle, nous avons choisi de nommer cette table Ages et la colonne Age de la table initiale est devenue la colonne Valeur.

Ages

«PK» Numero	«UID» Valeur	Categorie
1	12	A
2	13	B
3	14	B

Concurrents

«PK» Numero	«FK» Age_Numero	«UID» Dossard	Nom	Prenom
1	1	1000	Bleu	Claude
2	2	1004	Orange	Jeannie
3	1	1005	Marron	Marc
4	3	1008	Noir	Claude

Remarques :

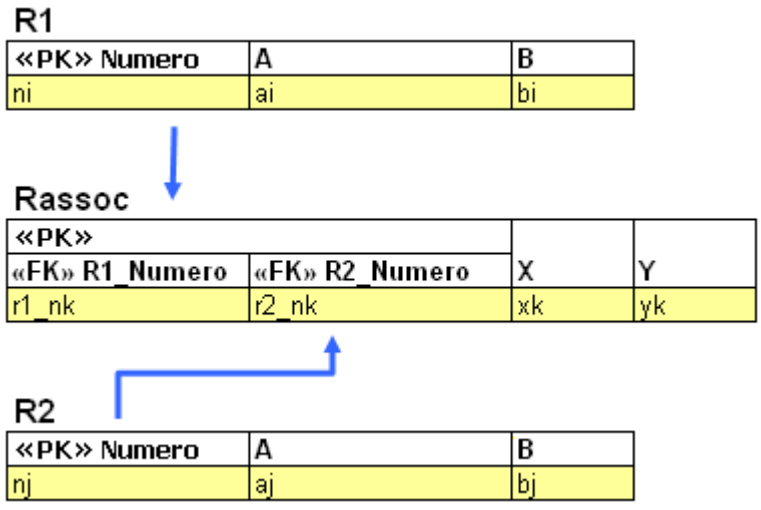
La table initiale des concurrents de cet exemple est telle que nous aurions pu la concevoir pour résoudre de manière simpliste l'exemple de non respect de la 1ère forme normale s'agissant du cas d'une relation au sein d'une colonne.
De ce fait la solution ci-dessus est, naturellement, identique à celle proposée pour l'exemple précité.

8 Autres formes normales

8.1 Forme normale de Boyce-Codd (BCNF)

Une table (associative ou dépendante)
est en forme normale de Boyce-Codd (BCNF)
si elle est déjà en 3ème forme normale (3NF) et
qu'il n'existe pas de colonne[7] source d'une dépendance fonctionnelle avec une partie de la clé primaire.

Seule les tables comportant une clé primaire composée de plusieurs colonnes (clé composite) sont concernées.



Rassoc est en forme normale de Boyce-Codd si :

$$X \not\rightarrow R1\_Numero$$

[7] Pour rappel : hormis les colonnes de clé secondaire unique et non nulle «UID»

$X \nrightarrow$ R2_Numero

$Y \nrightarrow$ R1_Numero

$Y \nrightarrow$ R2_Numero

8.1.1 Règle de normalisation

Important : Nous formalisons pour la première fois une règle de normalisation et la donnons avec toutes les réserves qui s'appliquent à une proposition non éprouvée.

1. Lorsqu'une table (R_{assoc}) comporte une colonne qui est source d'une dépendance fonctionnelle (df_1) avec une partie de la clé primaire, il faut créer une nouvelle table (R_{nouv}).

2. La colonne source de la dépendance fonctionnelle (df_1) doit être transférée dans cette nouvelle table (R_{nouv}), en principe comme clé secondaire unique et non nulle. Une colonne Numero, base de clé primaire doit être créée.

3. La colonne de clé étrangère cible de la dépendance fonctionnelle (df_1) doit aussi être transférée dans cette nouvelle table (R_{nouv}). La table de référence de cette clé étrangère est conservée.

4. Une colonne de clé étrangère est ajoutée dans la table initiale (R_{assoc}) ; cette colonne référera la nouvelle table (R_{nouv}).

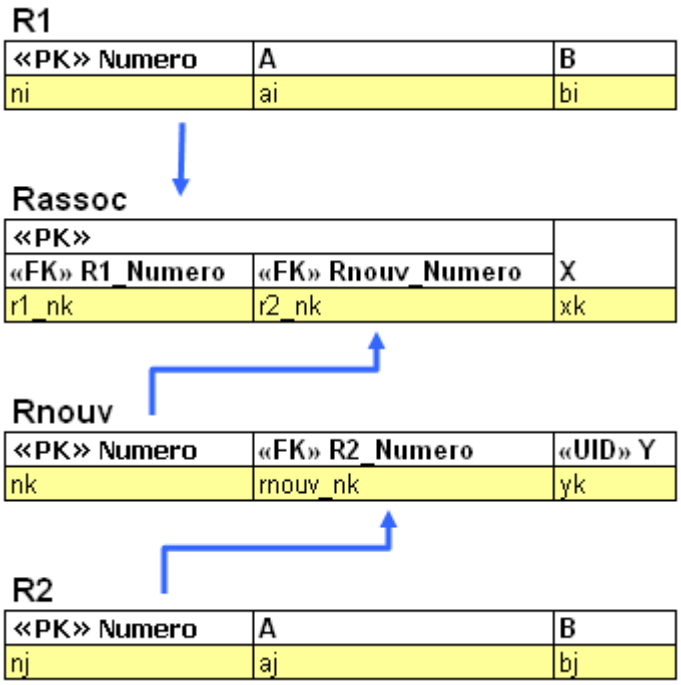
En résumé, la normalisation consiste à insérer une nouvelle table entre la table à normaliser et la table référencée par la partie de clé primaire qui est en dépendance fonctionnelle.

8.1.2 Illustration

Sur la base du modèle présenté initialement, nous admettons qu'il existe la dépendance fonctionnelle suivante pour la table Rassoc:

$$Y \rightarrow R2\_Numero$$

L'application de notre règle de modélisation nous a conduit au modèle suivant



De manière plus circonstanciée, voici les différentes opérations que nous avons réalisées pour normaliser la table Rassoc.

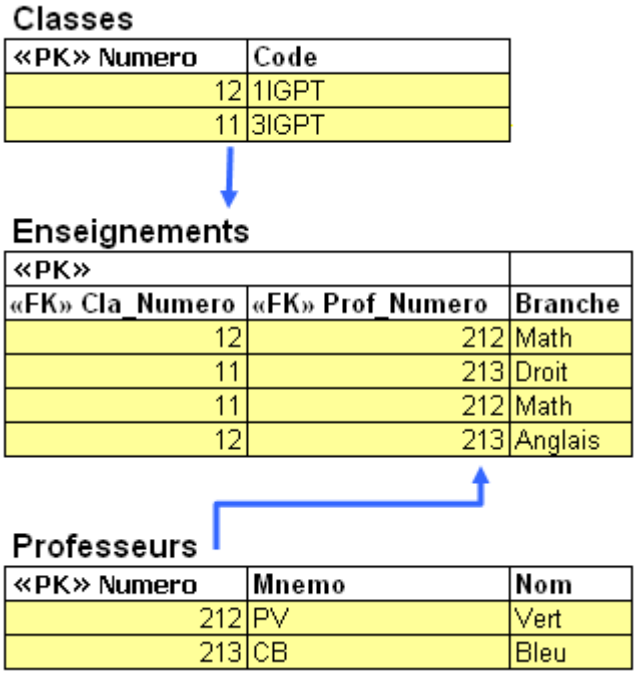
1. Nous avons créé la table Rnouv

2. Nous avons doté la table Rnouv d'une clé primaire Numero ; nous avons rapatrié la colonne Y de la table Rassoc dans la table Rnouv et l'avons spécifiée comme clé secondaire unique et non nulle «UID».

3. Nous avons déplacé la colonne de clé étrangère R2_Numero de la table Rassoc dans la table Rnouv et conservons la table R2 en référence.

4. Nous avons créé la colonne de clé étrangère Rnouv_Numero dans la table Rassoc et l'avons mise en relation avec la table Rnouv.

8.1.3 Exemple



Classes

«PK» Numero	Code
12	1IGPT
11	3IGPT

Enseignements

«PK»		
«FK» Cla_Numero	«FK» Prof_Numero	Branche
12	212	Math
11	213	Droit
11	212	Math
12	213	Anglais

Professeurs

«PK» Numero	Mnemo	Nom
212	PV	Vert
213	CB	Bleu

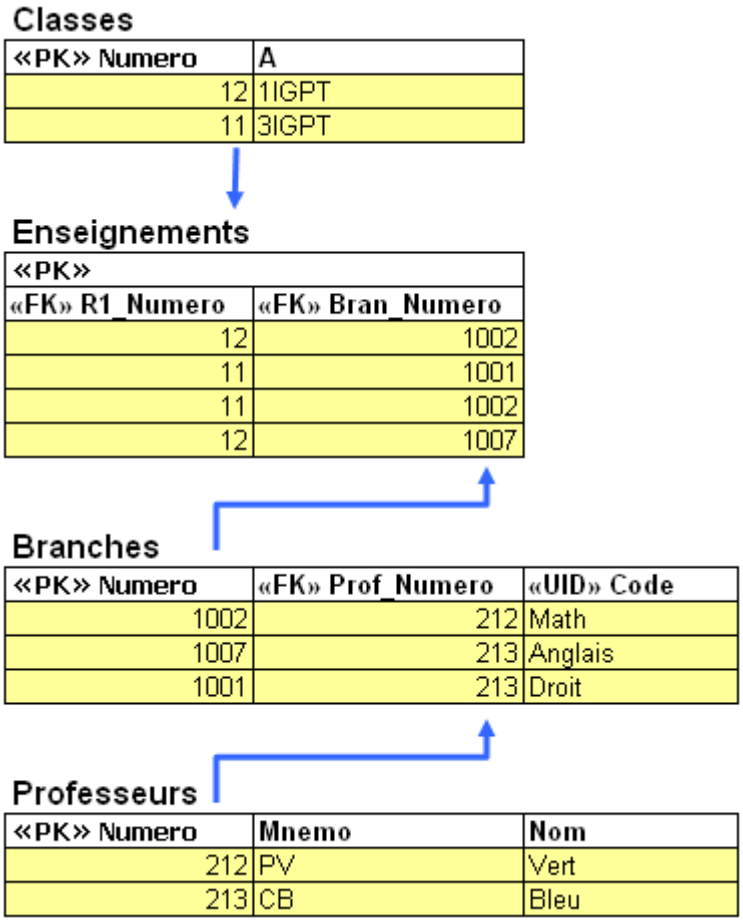
> Une règle de gestion stipule qu'une branche n'est toujours enseignée que par un et un seul professeur.

Dès lors, pour la table Enseignements, nous avons la dépendance fonctionnelle suivante :

$$\text{Branche} \rightarrow \text{Prof\_Numero}$$

Conformément à la règle de normalisation énoncée, nous insérons une nouvelle table Branches entre les tables Professeurs et Enseignements.

La table Professeurs reste référencée par la table Enseignements mais par transitivité via la nouvelle table Branches.



Classes

«PK» Numero	A
12	1IGPT
11	3IGPT

Enseignements

«PK»	
«FK» R1_Numero	«FK» Bran_Numero
12	1002
11	1001
11	1002
12	1007

Branches

«PK» Numero	«FK» Prof_Numero	«UID» Code
1002	212	Math
1007	213	Anglais
1001	213	Droit

Professeurs

«PK» Numero	Mnemo	Nom
212	PV	Vert
213	CB	Bleu

8.2 Dépendance multivaluée

Une colonne B d'une table R est en dépendance multivaluée d'une colonne A de R si à une valeur a_i de A plusieurs valeurs de B existent.

Une dépendance multivaluée est notée

$$A \twoheadrightarrow B$$

En respectant les principes de modélisation et de normalisation que nous avons énoncés, nous ne devrions pas être confrontés à la problématique de dépendances multivaluées.

8.2.1 Discussion

Une dépendance multivaluée est essentiellement due à une transformation malheureuse d'une colonne qui n'est pas en première forme normale (1NF).

Reprenons notre exemple de groupe répétitif que nous avons utilisé au chapitre traitant de la première forme normale pour expliquer notre propos.

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom	Sauts
1	1000	Bleu	Claude	3.28 , 3.07
2	1004	Orange	Jeannie	0 , 3.00
3	1005	Marron	Marc	4.10 , 0
4	1008	Noir	Claude	2.35 , 2.50

Certains auteurs font une réflexion basée uniquement sur les colonnes de tables en faisant abstraction de l'apport du concept de clé (primaire, étrangère et «UID»); ainsi, ils peuvent voir notre table de concurrents sous la forme ci-dessous.

Concurrents

Dossard	Nom	Prenom	Sauts
1000	Bleu	Claude	3.28 , 3.07
1004	Orange	Jeannie	0 , 3.00
1005	Marron	Marc	4.10 , 0
1008	Noir	Claude	2.35 , 2.50

Tout comme nous, ils détectent la non atomicité de la colonne Sauts ; par contre et contrairement à nous qui créons une table dépendante en nous appuyant sur le concept de clés primaires et étrangères, ils remplacent simplement la colonne Sauts qui est une chaîne de caractères formatée pour enregistrer plusieurs sauts par une colonne Saut qui, pour ce cas, contiendra une simple valeur numérique ; ensuite, ils distribuent les données multiples dans autant d'enregistrement que nécessaires en dupliquant les données identiques.

Concurrents

Dossard	Nom	Prenom	Saut
1000	Bleu	Claude	3.28
1004	Orange	Jeannie	0
1005	Marron	Marc	4.1
1008	Noir	Claude	2.35
1000	Bleu	Claude	3.07
1004	Orange	Jeannie	3
1005	Marron	Marc	0
1008	Noir	Claude	2.5

La table présentée ci-dessus est bien en 1ère forme normale (1NF) mais elle comporte une dépendance multivaluée :

$$\text{Dossard} \twoheadrightarrow \text{Saut}$$

Par exemple, pour la valeur 1000 de Dossard nous trouvons deux lignes ; l'une contient la valeur 3.28 de Saut et l'autre la valeur 3.07. Dans ces deux lignes, les valeurs de Nom, Prenom et Dossard sont dupliquées.

8.3 4ème forme normale (4NF) et 5ème forme normale (5NF)

Les quatrième et cinquième formes normales traitent des dépendances multivaluées.
Les règles énoncées de normalisation d'attributs non atomiques (1ère forme normale – 1NF) et l'utilisation de clés primaires nous évitent de créer des dépendances multivaluées ; en conséquence et jusqu'à preuve du contraire, les 4ème et 5ème forme normale qui tendent à réduire les dépendances multivaluées n'ont pas d'intérêt pour notre cours.

9 Redondances « horizontales »

9.1 Définition

Les formes normales règlent les problèmes de redondances « verticales » ; nous entendons par verticales, les redondances qui sont détectables par l'observation de la structure des colonnes d'une table[8] .

Il nous reste à résoudre le problème de redondances « horizontales » ; nous entendons par horizontales, les redondances qui sont liées au caractère répétitif d'enregistrement de données identiques en de multiples lignes ou tuples.

Il y a redondance horizontale lorsqu'une colonne peut être source de données dotée d'une existence propre; il n'y a pas redondance horizontale pour une colonne dont les données ne sauraient avoir une existence propre.

9.2 Règle de normalisation

Lorsqu'une table comporte une colonne source de données avec existence propre, il faut transformer la colonne en une nouvelle table et établir une relation entre les deux tables.

9.3 Exemple

Dans le table Concurrents ci-dessous, nous pouvons constater que chaque colonne non clé primaire ou «UID» contient des données redondantes horizontalement.

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom	Sexe	Rue	Localite
1	1000	Bleu	Claude	G	Rue de la Gare	Cornaux
2	1004	Orange	Jeannie	F	Rue de la Gare	Cressier
3	1005	Marron	Marc	G	Rue de la Poste	Cressier
4	1105	Azur	René	G	Rue du Temple	Cornaux
5	1108	Noir	Claude	F	Rue de la Gare	Marin
6	1008	Vert	Daniel	G	Rue des Fontaines	Cornaux
6	1010	Vert	Cynthia	F	Rue des Fontaines	Cornaux

Nous admettons pour les besoins de notre gestion de concurrents que :

- Nom, Prenom et Rue sont des données qui ne doivent pas avoir d'existence propre ; nous ne ferons pas de traitement spécifique à partir de ces colonnes.
- Localité et Sexe sont des données qui doivent avoir leur propre existence ; nous ferons très certainement des traitements qui devront se baser sans ambiguïté sur une localité ou sur le sexe des concurrents.

[8] Relation selon la terminologie initiale de Codd

Conformément à la règle de normalisation, nous rapatrions les colonnes Sexe et Localite dans deux nouvelles tables Sexes et Localites ; nous mettons ces deux nouvelle tables en relation avec la table Concurrents.

Sexes

«PK» Numero	«UID» Code
1	F
2	G

Localites

«PK» Numero	«UID» Nom
1	Cornaux
2	Cressier
3	Marin

Concurrents

«PK» Numero	«UID» Dossard	«FK» Sexe_Numero	«FK» Loc_Numero	Nom	Prenom	Rue
1	1000	2	1	Bleu	Claude	Rue de la Gare
2	1004	1	2	Orange	Jeannie	Rue de la Gare
3	1005	2	2	Marron	Marc	Rue de la Poste
4	1105	2	1	Azur	René	Rue du Temple
5	1108	1	3	Noir	Claude	Rue de la Gare
6	1008	2	1	Vert	Daniel	Rue des Fontaines
6	1010	1	1	Vert	Cynthia	Rue des Fontaines

9.4 Critères de détermination de redondance

La détermination d'existence propre ou pas des données d'une colonne est spécifique aux finalités attendues de chaque structure de données.
Néanmoins, les éléments suivants doivent être pris en compte pour déterminer les colonnes qui peuvent être source de données propres donc de redondances horizontales :

- un traitement doit pouvoir s'appliquer avec pertinence à partir d'une donnée redondante; pour notre exemple, nous pourrions rechercher tous les concurrents qui habitent Cornaux. Si la localité n'est pas transformée en une table, nous courrons le risque de fournir des résultats erronés au cas où la saisie de la colonne Localite n'est pas identique dans tous les enregistrements de la table non normalisée ;
- le contenu de la donnée redondante peut devoir être étendu ; pour notre exemple, nous avons saisi un code G pour les garçons et F pour les filles. Nous pouvons tout à fait admettre q'il faille à terme ajouter un libellé qui permettra d'imprimer des listes en mentionnant Garçon ou Fille au lieu de G et F ;
- le contenu de la donnée redondante peut devoir être traduit dans la vision d'une application multi-langues.

10 Colonnes redondantes

Que ce soit pour mettre en 1ère forme normale ou spécifié directement, il est faux de créer de multiples colonnes qui enregistrent une information de même nature.

Dans la table Concurrents ci-dessous, les colonnes Saut1 et Saut2 introduisent une redondance non pas de données mais de colonnes ; en effet, au lieu d'avoir une colonne pour les sauts nous avons deux colonnes qui contiennent des données de même nature (la longueur du saut d'un concurrent).

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom	Saut1	Saut2
1	1000	Bleu	Claude	3.28	3.07
2	1004	Orange	Jeannie	0	3
3	1005	Marron	Marc	4.1	0
4	1008	Noir	Claude	2.35	2.5

Cette redondance de colonnes qui contiennent des informations de même nature est source de difficultés multiples de manipulation de données; par exemple, comme nous l'avons déjà cité précédemment, nous ne pouvons pas faire, sans artifices, une requête SQL qui ressort le saut le plus long ou une requête SQL qui calcule la moyenne des sauts réalisés.

Pour résoudre cette redondance, nous appliquons une des deux recettes que nous avons décrites pour la mise en 1ère forme normale (1NF) de colonnes contenant un groupe répétitif. Pour notre exemple, il s'agit de la recette pour les données sans existence propre.

Concurrents

«PK» Numero	«UID» Dossard	Nom	Prenom
1	1000	Bleu	Claude
2	1004	Orange	Jeannie
3	1005	Marron	Marc
4	1008	Noir	Claude

Sauts

«PFK» Conc_Numero	«PK» NumeroDep	Valeur
1	1	3.28
1	2	3.07
2	1	0
2	2	3
3	1	4.1
3	2	0
4	1	2.35
4	2	2.5

Remarque : La clé primaire NumeroDep ou mieux un attribut Ordre peuvent servir à enregistrer l'éventuel ordre de chacun des sauts.