LES CONFLITS ENTRE LES CONTRAINTES DANS LES SCHEMAS CONCEPTUELS DE BASES DE DONNEES : UML – EER

Faouzi BOUFARES
Maître de conférences en informatique
boufares@lipn.univ-paris13.fr +33 1.49.40.40.71

Djamel BERRABAH
Doctorant en informatique
berrabah@math-info.univ-paris5.fr +33 1.42.86.40.42

Charles-François DUCATEAU
Professeur des universités en informatique
ducateau@iut.univ-paris5.fr +33 1.44.14.45.51

Faiez GARGOURI
Maître de conférences en informatique
faiez.gargouri@fsegs.rnu.tn

Résumé : Le travail de modélisation des données est toujours une activité délicate et nécessite une bonne expérience des concepteurs. Plusieurs formalismes permettent de modéliser les données tels que UML et EER. Afin de préserver la sémantique du monde réel, plusieurs catégories de contraintes peuvent être introduites dans ces deux formalismes telles que les contraintes de multiplicité (cardinalité) et les contraintes d'intégrité fonctionnelle. Notre objectif dans ce papier est de traiter la cohérence globale de ces contraintes. Notre méthode consiste à faire correspondre au schéma conceptuel un système linéaire d'inégalités à résoudre en utilisant les techniques de la programmation linéaire. A partir de cette résolution, nous détectons et localisons les conflits engendrés par les contraintes.

Summary: Data modeling is a delicate activity and requires that modelers have a good experience. Several formalisms, such as UML and EER, allow to model data. In order to preserve the semantics of the real world, many categories of constraints have been introduced into these two formalisms. Multiplicity (cardinality) constraints and functional integrity constraints are two examples of such constraints. Our aim in this paper is to treat the total coherence of these constraints. Our method consists in corresponding to the conceptual diagram a linear system of inequalities to solve by using the linear programming techniques. From this resolution, we detect and locate the conflicts generated by the constraints.

Mots-clés : Base de Données, Modélisation UML/ER, diagramme de classes/ER valide, Contraintes de multiplicité/cardinalité, Contraintes d'intégrité fonctionnelle, Outil d'Aide à la Conception, Programmation linéaire, Qualité des données.

Les conflits entre les contraintes dans les schémas conceptuels de Bases de Données : UML – EER

1 - INTRODUCTION

La tendance des recherches en cours est de permettre le partage des connaissances et des données entre les communautés, les utilisateurs et les applications. On peut donc être amené à intégrer des systèmes d'information et des bases de données. Parmi les problèmes qui peuvent surgir, nous pouvons citer par exemple, celui de la **cohérence des contraintes** définies sur les données, la **qualité** de ces dernières en dépend.

Plusieurs formalismes ont été utilisés pour modéliser des données, tels que **UML** (Unified Modeling Language) dans Muller (2001), Soutou (2002) et Rumbaugh (2004), **ER** (Entity-Relationship) et **EER** (Extended Entity-Relationship) dans Chen (1976), Moulin (1976), Tardieu (1979) et Smith (1977). Afin de préserver la sémantique du monde réel, plusieurs *contraintes* sont définies sur les données. Alors que les propriétés formelles de chacune de ces contraintes sont largement maîtrisées, peu de travaux sont faits pour intégrer tous les concepts à la fois et donc étudier les conflits éventuels qui peuvent exister entre les différentes catégories de contraintes. Les outils d'aide à la conception de Bases de Données (BD) qui existent aujourd'hui, tels que Power Designer et Rational Rose, n'offrent pas la possibilité de contrôler la cohérence globale des différents types de contraintes.

Les seules études faites sur la cohérence des contraintes, traitent essentiellement celles de cardinalité dans un schéma conceptuel ER. La découverte des cycles critiques dans un graphe associé au schéma conceptuel permet de détecter que les contraintes de cardinalités ne sont pas satisfiables, Lenzerini (1990). Mais cette étude ne traite pas toutes les associations réflexives. En effet, les différents rôles de ce type d'associations doivent être pris en compte. Une classification des associations réflexives dans le formalisme ER a été donnée dans Dullea (1999). L'interaction de différentes classes de contraintes, en particulier les dépendances fonctionnelles et les contraintes de cardinalités, a été brièvement étudiée par Hartmann (1998, 2000). La satisfaction des contraintes de cardinalité a été abordée par Boufarès (2002, 2004). Elle se vérifie par la résolution du système linéaire associé au schéma ER pour détecter et localiser les conflits.

Certains de ces travaux ont été réalisés dans le cadre du modèle ER. Par contre le formalisme UML, qui a connu un développement très rapide depuis sa création, a fait l'objet de peu de contributions concernant l'étude des différents types de contraintes. Les outils d'aides à la conception dédiés à la modélisation UML n'offrent pas la possibilité de vérifier la cohérence globale de ces différents types de contraintes. Notons, par ailleurs, que le langage OCL (Object Constraint Language) OMG (2003c) n'assure pas la vérification de la cohérence globale des contraintes.

L'étude détaillée des conflits éventuels entre les contraintes définies sur les grosses masses de données nous semble très intéressante à plusieurs titres. Elle pourrait être utilisée dans le processus d'intégration de données et dans l'évaluation de la qualité de celles-ci.

Ce papier est organisé comme suit. La deuxième section présente brièvement les différents formalismes étudiés (UML et EER). Les différentes catégories de contraintes sont exposées dans la section trois qui détaille leur mise en équations. Nous présentons notre algorithme de détection et localisation des conflits dans la quatrième section. Enfin, nos travaux futurs sont donnés en guise de conclusion.

2 - LES FORMALISMES UML ET EER

Dans cette section, nous décrivons très brièvement les caractéristiques des formalismes UML et EER, utiles dans la suite de ce papier dans le but de présenter les différents types de contraintes que nous étudions. Pour plus de détails, les documents qui nous paraissent ceux de base sont ceux de Rumbaugh (2004), Chen (1976), Tardieu (1979) et Rochfeld (1993)

Un schéma, conceptuel S, de BD peut être représenté graphiquement avec deux

formalismes très proches UML et EER. Un schéma de BD peut ainsi être formulé aussi bien par un *diagramme de classes* UML (D_UML) que par un *diagramme entité-association* (D_EER). Les diagrammes de classes UML expriment de manière générale la structure statique d'un système, en termes de *classes* et de *relations* entre elles, agencées selon les règles dictées par les métamodèles donnés par exemple par OMG (2004, 2003a, 2003b). Les associations représentent les relations structurelles entre classes d'objets. La plupart des associations sont binaires. Elles peuvent relier une classe à elle-même dans le cas des structures récursives, ce type d'associations est appelé association réflexive. Il existe des associations n-aires reliant n classes entre elles. Chaque extrémité d'une association peut porter une indication de *multiplicité* qui montre combien d'objets de la classe considérée peuvent être reliés à un objet de l'autre classe. Les valeurs de multiplicité expriment les *contraintes* liées au domaine de l'application. Toutes sortes de contraintes peuvent être définies sur les liens entre objets, nous citons par exemple {ordonné}, {sous-ensemble}, {ou-exclusif}. Une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité s'appelle une agrégation ou une composition. UML emploie le terme de *généralisation* pour désigner la classification entre un élément plus général et un élément plus spécifique, on parle d'héritage.

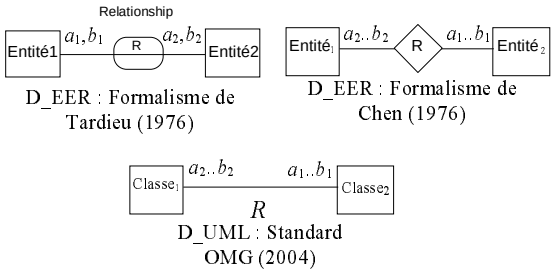


Figure.1 Différents formalismes de modélisation de schémas conceptuels de BD

Le formalisme EER est similaire à celui d'UML. Tous les concepts ci-dessus se retrouvent dans les diagrammes D_EER où on parle de types d'*entités* et d'*associations*, de contraintes de *cardinalité* et de *participation*. Les différents types d'associations cités ci-dessus, sont donnés dans un formalisme très proche, selon le métamodèle approprié, à la seule différence de l'agrégation.

Les diagrammes de la figure-1 résument les caractéristiques essentielles (entité, association, cardinalité, classe, multiplicité) dont on a besoin dans cet article. Notons que la lecture de la contrainte de cardinalité dans le formalisme ER de Tardieu est différente de celle dans les formalismes ER de Chen et celui d'UML.

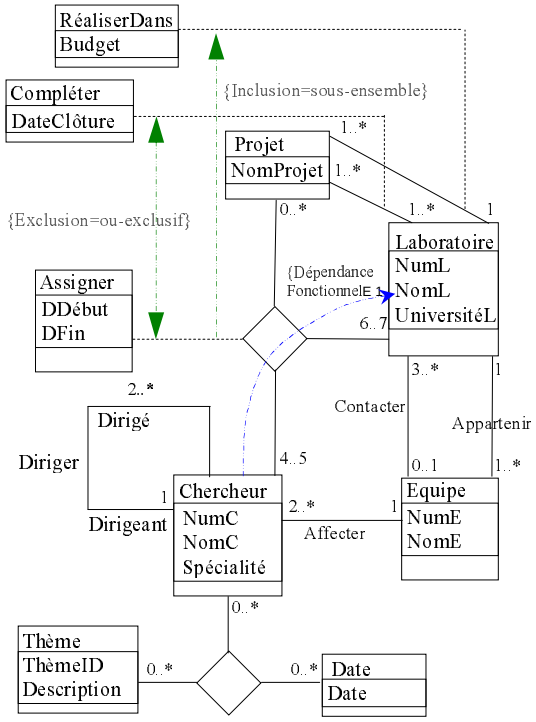


Figure.2 Gestion des laboratoires rattachés au CNRS
Diagramme de classes (D_UML)
Plusieurs catégories de contraintes

Les figures 2 et 3 représentent un exemple de schéma conceptuel de données respectivement UML et EER (les cardinalités sur la figure 3 sont à interpréter selon le modèle de Tardieu). Cet exemple décrit la gestion des laboratoires rattachés au centre national de recherche scientifique (CNRS). Chaque laboratoire doit contenir au moins une équipe. Chaque équipe doit contacter minimum trois laboratoires. Elle doit être composée d'au moins deux chercheurs. Un chercheur a au plus un directeur, ce dernier dirige au minimum deux chercheurs. Un chercheur n'est affecté qu'à une seule équipe. Cette dernière peut effectuer des projets dans des périodes différentes. Un chercheur travaille sur minimum 4 projets et

maximum 5 mais toujours dans un même laboratoire. Dans un laboratoire, il faut assigner minimum 6 couple (projet, chercheur) différents et maximum 7. Les diagrammes de ces deux figures (2 et 3) sont équivalents. Ils permettent de montrer que les mêmes problèmes peuvent être rencontrés dans les deux formalismes. En effet, ces deux diagrammes sont syntaxiquement corrects et donc acceptés par les différents outils d'aide à la conception. Ils sont cependant sémantiquement incorrects car certaines contraintes ne pourront jamais être vérifiées et par conséquent nous ne pouvons pas leur associer une instance de base de données cohérente. Nous prenons une partie de cet exemple et nous appliquons notre algorithme pour détecter les conflits générés par les contraintes.

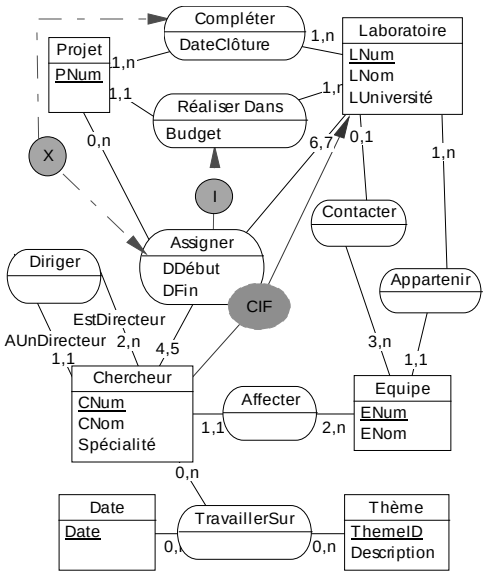


Figure.3 Gestion des laboratoires rattachés au CNRS Diagramme ER (D_EER)

3 - FORMALISATION DES CONTRAINTES

Un schéma conceptuel S, quel que soit le formalisme utilisé, est composé d'un ensemble de structures décrivant les données et d'un ensemble de contraintes sur ces données. Dans cette section nous illustrons notre approche pour formaliser les contraintes de multiplicité / cardinalité et les contraintes d'intégrité fonctionnelle.

Nous associons aux contraintes définies dans le schéma conceptuel S un système d'inéquations Σ_S défini par :

- Un ensemble **X** de variables $\boldsymbol{x_i}$ ($\boldsymbol{x_i}$ entier positif pour tout i) chacune correspond au nombre d'objets d'une classe C_i (type d'entité) ou celui des liens dans une association R_i.
- Un ensemble **A** de constantes $\boldsymbol{a_i}$ ($\boldsymbol{a_i}$ entier positif pour tout i) représentant les multiplicités (cardinalités) minimales.
- Un ensemble **B** de constantes $\boldsymbol{b_i}$ ($\boldsymbol{b_i}$ entier positif pour tout i) représentant les multiplicités (cardinalités) maximales.

La figure 4 représente un exemple d'un diagramme de classes avec une association binaire et comment nous associons des variables aux nombres d'objets de ses classes et au nombre de lien de son association.

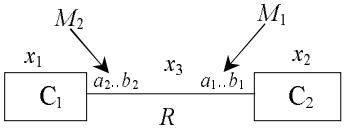


Figure.4 Diagramme de classes avec une association binaire

Nous considérons que x_1 et x_2 sont respectivement les nombres d'objets des classes C_1 et C_2, et x_3 est le nombre de liens de l'association R.

3.1 - Contraintes de multiplicité

La première catégorie de contraintes traitée est celle de multiplicité de type intervalle. Nous allons considérer ci-dessous les associations binaires et ternaires. Nous verrons les cas particuliers de l'agrégation et de la composition dans UML.

Une multiplicité, dans une association binaire, indique le nombre possible d'objets de la classe destination du rôle qui peuvent être reliés à un objet de la classe origine du rôle.

DEFINITION.1 : La multiplicité M_1, notée «$a_1..b_1$», de la classe C_1 appliquée sur l'association R (figure 4) est satisfaite si et seulement si pour tout objet v de la classe C_1, le nombre t de liens de R qui lui sont connectés doit être compris entre la multiplicité minimale et la multiplicité maximale « $a_1 \leq t \leq b_1$ ».

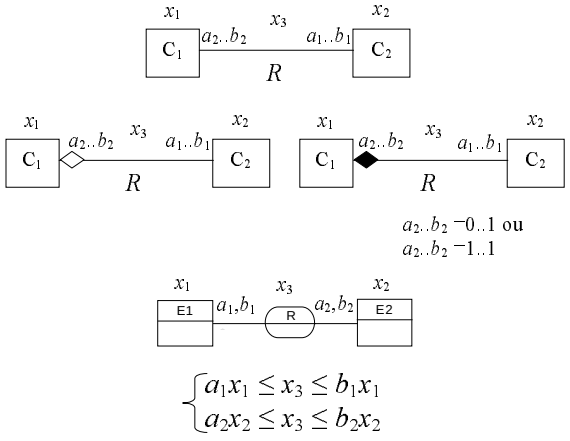
Cette inégalité est démontrée dans Boufarès (2002). D'une manière générale, la multiplicité M_1 est satisfaite si et seulement si x_3 (nombre de liens de l'association R) est compris entre

a_1x_1 (nombre minimum de participations de tous les objets de la classe C_1 dans l'association R) et b_1x_1 (nombre maximum de participations de tous les objets de la classe C_1 dans R).

DEFINITION.2 : Une multiplicité M_1 est satisfaite, si et seulement si, l'inéquation : $a_1x_1 \leq x_3 \leq b_1x_1$ est satisfaite.

Remarque : si la multiplicité minimum (resp. maximum) est égale à "0" (resp. est égale à "*") alors sa vérification est évidente, elle ne serra pas formalisée.

Les figures 5, 6 et 7 ci-dessous résument les inéquations associées aux contraintes de multiplicité définies sur les associations binaires non réflexives et réflexives ainsi que les associations ternaires dans les deux formalismes UML et EER. Les cas particuliers de l'agrégation et de la composition sont présentés.



$$\begin{cases} a_1x_1 \leq x_3 \leq b_1x_1 \\ a_2x_2 \leq x_3 \leq b_2x_2 \end{cases}$$

Figure.5 Multiplicité/Cardinalité dans une association binaire et les inéquations associées

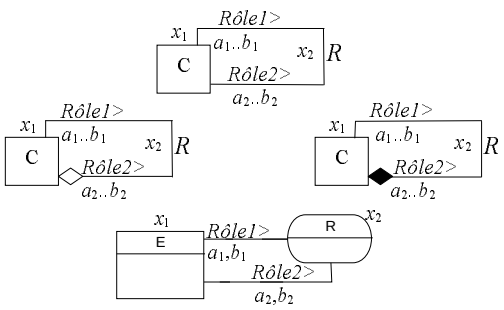
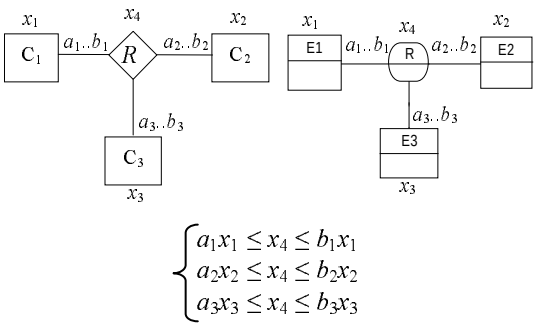


Figure.6 Multiplicité/Cardinalité dans une association récursive et les inéquations associées



$$\begin{cases} a_1x_1 \leq x_4 \leq b_1x_1 \\ a_2x_2 \leq x_4 \leq b_2x_2 \\ a_3x_3 \leq x_4 \leq b_3x_3 \end{cases}$$

Figure.7 Multiplicité/Cardinalité dans une association ternaire et les inéquations associées

3.2- Contraintes d'intégrité fonctionnelle

3.2.1 - Dépendance fonctionnelle

Une dépendance fonctionnelle (DF) entre deux classes C_1 et C_2 exprime qu'à un objet c_1 de C_1 correspond au plus un objet c_2 de C_2.

Pour satisfaire cette condition, il faut que le nombre d'objets de la classe C_2 soit inférieur ou égal à celui de la classe C_1. Cela se traduit par : $\boldsymbol{x_2 \leq x_1}$.

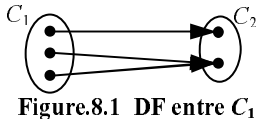


Figure.8.1 DF entre C_1 et C_2

Une DF, définie dans une association ternaire, entre deux classes C_1 et C_2 doit satisfaire les règles de correspondances entre les objets selon la figure-8.2.

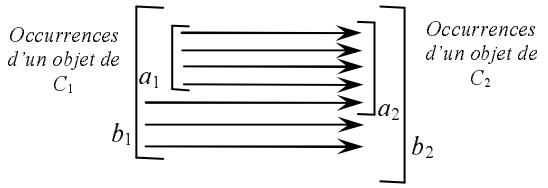


Figure.8.2 Correspondances entre les occurrences de C_1 et C_2

Il faut qu'il y ait, dans R, suffisamment d'occurrences des objets de C_2 pour satisfaire les occurrences des objets de C_1 :

- La multiplicité $a_1..b_1$, définie sur l'association R, exige la présence d'au minimum a_1 liens pour un même objet de la classe C_1, et d'au maximum b_1 liens pour ce même objet ;
- La multiplicité $a_2..b_2$, définie sur l'association R, exige la présence d'au minimum a_2 liens pour un même objet de la

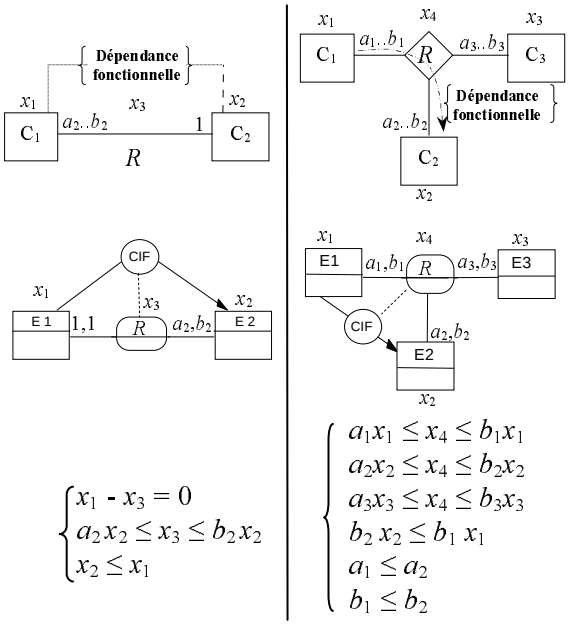
classe C_2, et d'au maximum b_2 liens pour ce même objet ;

- La dépendance existant entre C_1 et C_2 exige qu'un objet de C_1 ne puisse être relié qu'à un seul objet de C_2.

L'interaction de ces deux types de contraintes (multiplicité et dépendance fonctionnelle) génère un ensemble de critères. Ces derniers doivent être respectés afin de préserver la sémantique du monde réel.

- Le fait d'avoir la possibilité de prendre minimum a_1 occurrences d'un objet de C_2 exige que la multiplicité minimum a_2 soit supérieure ou égale à a_1. ($\boldsymbol{a_1 \leq a_2}$ d'où $\boldsymbol{a_1 \leq b_2}$)
- Le fait d'avoir la possibilité de prendre un maximum b_2 d'occurrences d'un objet de C_2 exige que la multiplicité maximum b_2 soit supérieure ou égale à b_1. ($\boldsymbol{b_1 \leq b_2}$)
- Le fait d'avoir une correspondance entre les objets de C_1 et C_2 exige avoir suffisamment d'occurrences des objets de la classe C_2 pour satisfaire les occurrences des objets de la classe C_1 ce qui donne : $b_2 x_2 \leq b_1 x_1$.

La figure.9 résume les différents cas possibles.



$$\begin{cases} x_1 - x_3 = 0 \\ a_2 x_2 \leq x_3 \leq b_2 x_2 \\ x_2 \leq x_1 \end{cases}$$

$$\begin{cases} a_1 x_1 \leq x_4 \leq b_1 x_1 \\ a_2 x_2 \leq x_4 \leq b_2 x_2 \\ a_3 x_3 \leq x_4 \leq b_3 x_3 \\ b_2 x_2 \leq b_1 x_1 \\ a_1 \leq a_2 \\ b_1 \leq b_2 \end{cases}$$

Figure.9 Interaction entre Multiplicité/Cardinalité et DF ainsi que les inéquations associées

3.2.2 - Contraintes de participation

Ces conditions concernent fréquemment la coexistence d'objets de plusieurs associations au départ d'une classe commune. La cohérence globale de ces contraintes devra être contrôlée afin d'empêcher la production des conflits. Dans ce paragraphe, nous présentons les contraintes d'exclusion et de simultanéité dans le formalisme UML. Il en est de même pour le formalisme EER (Figure.10).

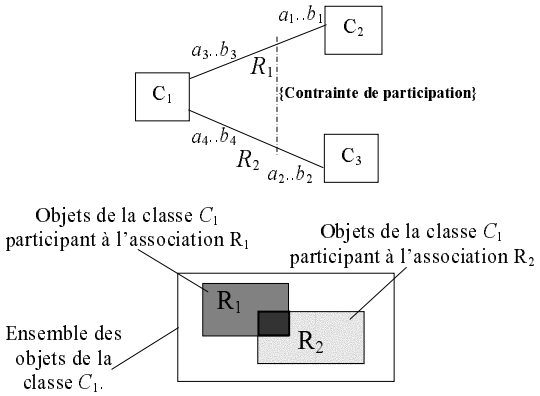


Figure.10 Contrainte de participation

3.2.2.1- Contrainte d'exclusion

Les objets d'une classe peuvent participer à deux ou plusieurs associations différentes. Cette participation peut être mutuellement exclusives. Cette contrainte est notée {**Exclusion**} (**X** en EER). Des définitions plus détaillées sont données par Matheron (1991), Rochfeld (1993) et Nanci (2001). Une contrainte d'exclusion d'une association R_1 vis-à-vis d'une association R_2 exprime le fait que les objets de la classe C_1 participant à l'association R_1 ne peuvent pas participer à l'association R_2 (figure.11).

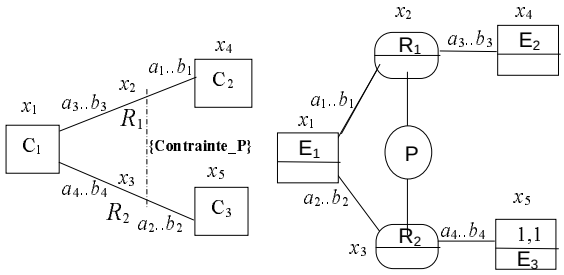


Figure.11 Interaction entre Multiplicité / Cardinalité et exclusion/simultanéité.

Dans le cas d'une exclusivité de participation d'une classe à deux associations, il faut que les multiplicités minimales de cette classe sur ces associations soient nulles. Dans le cas contraire, multiplicités minimales non nulles, on constate deux cas possibles. Le premier est celui où l'une des multiplicités n'est pas nulle ce qui oblige la participation totale de tous les objets de la classe C_1 dans cette association et non plus dans l'autre. On déduit alors que la

présence de cette dernière dans le schéma conceptuel est inutile. Le deuxième cas est celui où les deux multiplicités minimales ne sont pas nulles. Par conséquent, la participation de tout objet de C_1 est obligatoire dans R_1 et R_2 en même temps. La contrainte d'exclusion devient inutile dans ce cas.

Pour que la contrainte d'exclusion soit respectée, Il faut vérifier le nombre total des participations des objets de C_1 dans R_1 et R_2 par rapport au nombre minimum et le nombre maximum possibles de participations des objets de C_1, d'où :

$$\boldsymbol{x_2 + x_3 \leq \max(b_1,b_2)\, x_1}$$

Dans le cas d'interaction des contraintes de multiplicité avec la contrainte d'exclusion (P = X), le système d'inéquations est le suivant :

$$\begin{cases} x_2 \leq b_1 x_1 \\ x_3 \leq b_2 x_2 \\ a_3 x_4 \leq x_2 \leq b_3 x_4 \\ a_4 x_5 \leq x_3 \leq b_4 x_5 \\ x_2 + x_3 \leq \max(b_1,b_2)\, x_1 \end{cases}$$

3.2.2.2- Contrainte de simultanéité

Les définitions données à la contrainte de simultanéité dans la littérature, sont ambiguës. Cette dernière est notée {**Simultanéité**} (**S** en EER). Nous présentons dans ce papier deux cas différents de cette contrainte :

- Simultanéité de participation des objets d'une classe dans deux associations en terme d'existence.
- Simultanéité de participation des objets d'une classe dans deux associations en terme du nombre de participations de ces objets dans ces associations.

Le premier cas ne pose pas de problème puisqu'il n'a pas d'influence sur le nombre des occurrences des objets de la classe participant aux associations. Le deuxième cas pose problème d'une part dans la définition des contraintes de multiplicité, et d'autre part dans l'interaction des cette contrainte avec les contraintes de multiplicité. D'après la deuxième définition de la contrainte de simultanéité, les objets d'une classe doivent participer dans une association autant de fois que dans l'autre. Donc les multiplicités, minimales et maximales, de cette classe dans ces associations doivent être égales. Un autre critère est que le nombre d'objets des associations sur lesquelles est définie cette contrainte doit être compris entre $\boldsymbol{2\ a_1\ x_1}$ (tel que $a_1=a_2$) et $\boldsymbol{2\ b_1\ x_1}$ (tel que $b_1=b_2$) d'où l'inégalité :

$$\boldsymbol{2\ a_1\ x_1 \leq x_2 + x_3 \leq 2\ b_1\ x_1}$$

Le fait que la simultanéité soit définie sur le nombre de participations, on déduit que le nombre de liens de R_1 est égal à celui de R_2 ($\boldsymbol{x_2 = x_3}$). Donc, dans le cas d'interaction des contraintes de multiplicité avec la contrainte de simultanéité (P = S), le système d'inéquations est le suivant :

$$\begin{cases} a_1 x_1 \leq x_2 \leq b_1 x_1 \\ a_3 x_4 \leq x_2 \leq b_3 x_4 \\ a_4 x_5 \leq x_3 \leq b_4 x_5 \\ x_2 = x_3 \\ a_1 = a_2 \text{ et } b_1 = b_2 \end{cases}$$

Exemple 1 : Formalisation des contraintes d'une partie nommée S du schéma conceptuel de données de la figure 2.

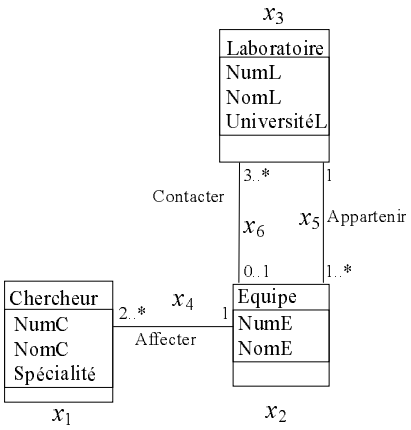


Figure.12 une partie du Schéma conceptuel de la figure.2 nommée S

L'ensemble des inéquations correspondant aux contraintes définies dans le schéma conceptuel S

$$\begin{cases} x_1 \leq x_4 \leq x_1 \\ 2.\ x_2 \leq x_4 \\ x_2 \leq x_5 \leq x_2 \\ x_3 \leq x_5 \\ 3.\ x_2 \leq x_6 \\ x_6 \leq x_3 \end{cases}$$

4 - DETECTION ET LOCALISATION DES CONFLITS

Nous associons à tout schéma conceptuel S, un système linéaire d'inéquations Σ_S. Nous étudions dans cette section la manière de définir ce système à partir des inégalités

associées aux différentes contraintes introduites dans le schéma conceptuel (section.3). Ce système sera ensuite résolu en faisant appel aux techniques de la programmation linéaire. Pour plus de détail, se reporter à Schrijver (1986) et Williams (1993). Dans notre approche, nous avons adapté l'algorithme Fourier-Motzkin (figure 13) adapté par Boufarès (2002). La solution fournie par cet algorithme permet de décider de la validité du schéma conceptuel. Les conflits sont engendrés soit par des contraintes de multiplicité (cardinalité) ou par l'interaction de ces dernières avec d'autres contraintes (contraintes d'intégrité fonctionnelle). Notre but est de détecter et localiser les conflits.

DEFINITION.3 : Un schéma conceptuel de données S est valide, si et seulement s'il existe au moins une instance (un diagramme d'objet, une base de données) qui vérifie toutes les contraintes définies sur S.

DEFINITION.4 : Un schéma conceptuel de données S est valide, si et seulement si le système linéaire correspondant Σ_S a une solution, Boufarès (2002, 2004).

Nous avons montré dans la section précédente que toute contrainte peut être traduite sous la forme $a\,x \le y \le b\,x$ (exemple 2), où a, b, x et y sont des entiers positifs. Cette inégalité sera transformée comme suit :

- Si $a = b$ alors $ax - y = 0$.
- Si $a \neq b$ alors $\begin{cases} ax - y \le 0 \\ y - bx \le 0 \end{cases}$

Exemple 2 : La transformation des inégalité de l'exemple 1 sera comme suit. Cet exemple sera traité par l'algorithme de la figure 13.

Σ_S

Le système d'inéquations linéaires correspondant au schéma conceptuel S deviendra comme ci-contre

$$\begin{cases} x_1 - x_4 = 0 \\ 2.\ x_2 - x_4 \le 0 \\ x_2 - x_5 = 0 \\ x_3 - x_5 \le 0 \\ 3.\ x_2 - x_6 \le 0 \\ x_6 - x_3 \le 0 \\ x_i \text{ sont des entiers positifs } \forall\ i=1,6 \end{cases}$$

Le principe de l'algorithme est de générer de nouvelles inégalités. Si l'une parmi celles-ci a tous ses coefficients positifs, il s'agit alors d'une condition d'arrêt qui traduit l'incohérence des contraintes introduites dans le schéma conceptuel.

```
Etant donné ΣS associé au diagramme de classe S ;
Si (association récursive) et (multiplicité dans {(1-1..*) ;
    (1-0..*) ; (1-1)}) Alors S n'est pas valide
Sinon
    Pour chaque contrainte de la forme y – a x = 0
        Substituer y par a x et éliminer la contrainte ;
    V ← { liste des variables de ΣS } ;
    Sol ← vrai ;
    Tant que V non vide et Sol faire
        Choisir une variable x de V ;
        V ← V - { x } ;
        CP ← {liste des contraintes de ΣS dont le
        coefficient de x est positif} ;
        CN ← {liste des contraintes de ΣS dont le
        coefficient de x est négatif} ;
        Si CP et CN ne sont pas vide
        Alors
            Diviser chaque contrainte de CP et CN par la
            valeur absolue du coefficient de x ;
            Additionner les contraintes de CP à celles de
            CN deux à deux
                    et ajouter les inégalités générées au
            système ΣS;
            Eliminer du système toutes les contraintes
            nouvellement générées n'ayant que des
            coefficients négatifs ;
            Si parmi les nouvelles contraintes générées il
            existe une dont les coefficients sont positifs
            Alors Sol ← Faux ;
            Fin Si
        Fin Si
        Eliminer de ΣS toutes les contraintes qui ont été
        sélectionnées dans CP et CN ;
    Fin tant que
    Si Sol = Faux
    Alors ΣS n'a pas de solution; S n'est pas valide ;
    Sinon ΣS a une solution entière; S est  Valide ;
    Fin Si ;
Fin Si.
```

Figure.13 Algorithme de détection

Ainsi notre algorithme nous permet de détecter différents types de conflits sur les schémas conceptuels des figures 1 et 2. Sur la figure 2, par exemple, les conflits détectés sont :

- Ceux engendrés par les contraintes de multiplicité des associations *Contacter* et *Appartenir* reliant les classes *Laboratoire* et *Equipe* ;
- Ceux générés par la sémantique de l'association réflexive *Diriger* ;
- Ceux engendrés par l'interaction entre la dépendance fonctionnelle définie sur l'association *Assigner* entre les classes *Chercheur* et *Laboratoire* et les contraintes de multiplicité de l'association *Assigner* appliquées sur les classes *Chercheur*, *Laboratoire* et *Projet*.

5 - CONCLUSION

Dans ce papier, nous avons présenté une étude sur le traitement de la cohérence globale des contraintes pouvant être définies dans les formalismes UML ou EER. Il s'agit de déterminer où et par quelles contraintes sont engendrés les conflits. Nous avons explicité d'une part, les conflits au sein d'une même catégorie de contraintes (les contraintes de multiplicité Berrabah (2003) et Boufarès (2004)), et d'autres part, nous avons montré certains conflits qui peuvent exister entre les deux catégories de contraintes étudiées : les contraintes de multiplicité (cardinalité) et celles d'intégrité fonctionnelle.

Notre approche se résume dans le fait de traduire les contraintes du schéma conceptuel sous forme de système d'inéquations linéaires. Ce dernier sera résolu en faisant appel aux techniques de la programmation linéaire. La solution obtenue permet de valider le schéma conceptuel, le cas échéant de déterminer et localiser les conflits.

L'objectif de cette recherche est d'intégrer ces techniques de vérification dans les outils d'aide à la conception. On pourra ainsi « *compiler* » un schéma conceptuel de bases de données avant de générer les schémas logique et physique correspondants. Notre étude devra être généralisée pour étudier les différents types de contraintes en incluant tous les concepts des formalismes UML et EER (contraintes de multiplicité de type énuméré, héritage, contraintes de participation…).

BIBLIOGRAPHIE

Berrabah, D., Boufares, F.,Ducateau, C. F., Heiwy, V.(2003), "Etude de la cohérence des contraintes de multiplicité dans un schéma conceptuel de BD avec UML", *National Conference of Research in IUT* (CNRIUT 2003), Tarbes, France, May 2003, Pages 375-382.

Boufarès, F., Bennaceur, H. (2002), "Consistency Problems in ER-Schemas for Database systems", *Proceedings of the 2002 International Arab Conference on Information Technology* (ACIT'2002), Doha, Qatar, December 16-19, 2002 ; pages 699-706; Vol.2.

Boufarès, F., Bennaceur, H. (2004), "Consistency Problems in ER-Schemas for Database systems", *Information Sciences journal*, Volume 163, 18 June 2004, Pages 263-274.

Chen, P.P. (1976), "The Entity-Relationship Model -Towards a Unified View of Data", volume 1, *ACM Transactions On Database Systems*, March 1976, pages 9-36.

Dullea, J., Il-Yeol Song (1999), "A Taxonomy of Recursive Relationships and Their Structural Validity in ER Modelling", ER'99, Conceptual Modelling, *18th International Conference on Conceptual Modelling*, LNCS 1728, Paris, France, November 1999 , pages 384-398.

Hartmann, S. (1998), "On the Consistency of Int-cardinality Constraints", ER'98 : Conceptual Modelling, *17th International Conference on Conceptual Modelling*, LNCS 1507, Singapore, November 1998, pages 150-163.

Hartmann, S. (2000), "On Interactions of Cardinality Constraints, Key, and Functional Dependencies", FolKS'2000 : *Foundations of Information and Knowledge Systems, First International Symposium*, LNCS 1762, Burg, Germany, February 2000, pages 136-155.

Lenzerini, M., Nobili, P. (1990), "On the satisfiability of dependency constraints in Entity-Relationship schemata", Info. Syst. 90, *Information Systems*, volume 15, N° 4, 1990, pages 453-461.

Matheron, J.P. (1991), "Approfondir Merise", Tome1, Edition Eyrolles 1991.

Moulin, P., Randon, J., Spaccapietra, S., Tardieu, H., Teboul, M. (1976), "Conceptual model as database design tool", *Proceedings of the IFIP Working conference on modelling in database Management Systems*, G.M. Nijssen Ed., North-Holland, 1976.

Muller, P.A. (2001), "*Modélisation objet avec UML*", 2ème édition Eyrolles, 2001.

Nanci, D., Espinasse, B., Cohen, B., Asselborn, J.C., Heckenroth, H. (2001), "*Ingénierie des systèmes d'information : Merise deuxième génération*, 4e édition". Edition Vuibert, 2001.

OMG (2004), editor: "*UML 2.0 Superstructure Specification*", OMG, 2004. http://omg.org.

OMG (2003a), editor: "*UML 2.0 Diagram Interchange Specification*", OMG, 2003. http://omg.org.

OMG (2003b), editor: "*UML 2.0 Infrastructure Specification*", OMG, 2003. http://omg.org.

OMG (2003c), editor: "*UML 2.0 Object Constraint Language Specification*", OMG, 2003. http://omg.org.

Rochfeld, A., Negros, P.(1993), "Relationship of relationships and other inter-relationship links in ER model", *Data and Knowledge Engineering*, volume 9, 1993, pages 205-221.

Rumbaugh, J., Jacobson, I., Booch, G. (2004) *UML 2.0 Guide de Référence*, Edition CampusPress, 2004.

Schrijver, A. (1986), "*Theory of linear and integer programming*", Edition Wiley, 1986.

Smith, J.M., Smith, D.C.P. (1977), "Database abstractions: Aggregation and Generalization", *ACM Transactions on Database Systems*, volume 2, N°2, June1977, pages 105-133.

Soutou, Ch. (2002), "*De UML à SQL, Conception de bases de données*", Edition Eyrolles, 2002.

Tardieu, H., Nanci, A., Pascot, D. (1979), "A method, A Formalism and Tools for Database Design (three years of Experimental practice)". *Proceedings of the international Conference on Entity-Relationship Approach to Systems Analysis and Design*, 1979.

Williams, H.P. (1993), "*Model Solving in Mathematical Programming*" John Wiley and Sons Ltd, England, 1993.