Une Nouvelle Approche des Mises à Jour des Bases de Données

D. Laurent[◇*], V. Phan Luong[□*], N. Spyratos[*]

◇ LIFO
Université d'Orléans
BP 6759 - Orléans Cedex 2
FRANCE F-45067
email: laurent@univ-orleans.fr

□ LIM, U.R.A. 1787 du CNRS
Université de Provence
Place V. Hugo - Marseille Cedex 3
FRANCE F-13331
email: phan@gyptis.univ-mrs.fr

* LRI, U.R.A. 410 du CNRS
Bât. 490 - Université de Paris-Sud
Orsay Cedex
FRANCE F-91405
email: spyratos@lri.lri.fr

Résumé

Nous présentons une nouvelle approche des mises à jour des bases de données dont le point essentiel est que toute insertion et toute suppression est *déterministe, i.e.* toute information peut être insérée ou supprimée sans qu'il y ait à faire un choix en ce qui concerne la réalisation de la mise à jour. L'idée principale de cette approche est de *stocker* et de *marquer* les informations supprimées pour ensuite les utiliser comme des *exceptions* lors du calcul des réponses aux requêtes. Notre approche est tout d'abord présentée dans un contexte général, et est ensuite appliquée à deux formalismes différents : les interfaces de type relation universelle et les bases de données de type Datalog^{neg}.

Mots clés : base de données déductive, mise à jour, sémantique, plus petit point fixe, exception, interface, relation universelle.

1 Introduction

Dans cet article, nous nous intéressons au problème des mises à jour concernant les données *dérivées* dans les bases de données. À titre d'exemple, considérons une base de données relationnelle concernant les employés, les départements, et les projets gérés dans une entreprise. Supposons que cette base de données contienne deux relations r_1 et r_2, définies respectivement sur les schémas *EMP DEPT* et *EMP PROJ*. La présence d'un n-uplet tel que *John CS* dans r_1 signifie que l'employé de nom *John* travaille dans le département nommé *CS*, et la présence d'un n-uplet tel que *John A* dans r_2 signifie que l'employé de nom *John* travaille pour le projet *A*. Ainsi, en interrogeant la base de données, on peut obtenir les n-uplets *John CS A* (par jointure des relations r_1 et r_2), et *CS A* (par projection de la jointure ci-dessus sur le schéma *DEPT PROJ*).

Supposons maintenant qu'un utilisateur veuille insérer dans la base un nouveau projet *B* dans le département *CS*. Pour cela, l'utilisateur doit insérer le n-uplet *CS B*, mais une telle insertion n'est possible que si :

1. l'utilisateur fournit un nom d'employé, par exemple *e*, et si
2. les n-uplets *e CS* et *e B* sont respectivement insérés dans r_1 et r_2.

Ainsi, l'insertion de *CS B* doit être *traduite* en les insertions des n-uplets *e CS* et *e B*, et on remarque que cette traduction n'est pas unique, puisque l'employé *e* n'est pas précisé dans la mise à jour.

D'autre part, supposons qu'un utilisateur veuille supprimer le n-uplet *John CS A*. De nouveau, la suppression doit être *traduite* en l'une des suppressions de *John CS* ou de *John A*. Puisque les mises à jour sont supposées modifier le moins possible la base de données, on s'interdit de supprimer les deux n-uplets ci-dessus. La traduction n'est donc pas unique, puisque le n-uplet à supprimer n'est pas précisé dans la mise à jour.

Ces deux exemples très simples de mises à jour mettent en évidence les problèmes rencontrés lors de mises à jour sur des relations *intensionnelles*, *i.e.* des relations dont les schémas ne font pas partie du schéma de la base de données à mettre à jour. Ces problèmes, généralement connus sous le nom de mises à jour non déterministes (voir [LeS88, AtT89, LPS92a, LPS92b, LaS94]), sont également rencontrés lors de mises à jour concernant les prédicats intensionnels dans les bases de données déductives (voir [Win88, Bry90, AtT92, GMR92, LPS93a]).

L'approche proposée dans cet article consiste à *marquer* les informations dont la suppression est requise et à utiliser ces marques de façon à répondre

correctement aux requêtes. Nous décrivons maintenant notre modèle dans les contextes des bases de données relationnelles puis des bases de données déductives.

Bases de Données Relationnelles

Nous décrivons ici informellement notre approche dans le contexte des interfaces de type relation universelle, sous l'hypothèse que le schéma de la base n'est pas fixe. Une base de données est ainsi vue comme un ensemble de n-uplets dans lequel on insère un n-uplet ou duquel on supprime un n-uplet. Le point important dans notre approche est que l'on ne retire pas *physiquement* les n-uplets de la base : lorsqu'un n-uplet est supprimé, il est marqué comme étant supprimé et on fait en sorte de maintenir la base de données dans un état consistant. Ainsi, les n-uplets d'une base de données peuvent être divisés en deux parties disjointes : les n-uplets *insérés*, dont l'ensemble est noté INS et les n-uplets *supprimés* (donc marqués), dont l'ensemble est noté DEL ("deleted" en Anglais).

De manière intuitive, lorsqu'un n-uplet tel que *John CS A* est inséré dans la base de données, il est considéré comme *vrai* et, de même, lorsqu'un n-uplet est supprimé de la base, il est intuitivement considéré comme *faux*. Toutefois, cette manière de considérer les n-uplets vrais ou faux a pour conséquences que si le n-uplet *John CS A* est vrai, alors tous ses sous-n-uplets doivent aussi être vrais, et donc, si un n-uplet est faux, alors tous ses sur-n-uplets doivent aussi être faux.

On introduit donc deux opérateurs, notés *SUB* et *SUPER*, définis de la manière suivante : soit T un ensemble de n-uplets, alors

- $SUB(T)$ est l'ensemble des sous-n-uplets des n-uplets de T,
- $SUPER(T)$ est l'ensemble des sur-n-uplets des n-uplets de T.

Donc, les n-uplets de $SUB(\text{INS})$ sont considérés comme vrais et les n-uplets de $SUPER(\text{DEL})$ comme faux. Ainsi, une base est dite consistante si $SUB(\text{INS}) \cap SUPER(\text{DEL}) = \emptyset$.

D'autre part, les n-uplets qui n'appartiennent ni à $SUB(\text{INS})$ ni à $SUPER(\text{DEL})$ sont appelés n-uplets *inconnus*. On notera que l'on obtient ainsi une logique à trois valeurs, en ce sens qu'un n-uplet peut être vrai, faux ou inconnu. Par conséquent, notre approche traite de manière *symétrique* les notions de vrai et de faux, contrairement aux approches traditionnelles qui, par l'hypothèse du monde clos [Rei78], considèrent la notion de faux comme *complémentaire* de la notion de vrai.

Les opérateurs *SUB* et *SUPER* peuvent être vus comme un mécanisme d'inférence permettant de dériver des n-uplets vrais ou faux à partir des

n-uplets de INS de DEL. Néanmoins, cette manière de dériver des n-uplets peut conduire à des inconsistances. Par exemple, si on insère le n-uplet *John CS A* puis on supprime le n-uplet *John CS*, le n-uplet *John CS A* peut être considéré comme vrai et faux à la fois. On évite ce type d'inconsistance en modifiant la manière de dériver des n-uplets vrais comme suit : on accepte toute dérivation par l'opérateur *SUB* à *condition que* cette dérivation ne produise pas un n-uplet de *SUPER*(DEL).

En d'autres termes, les n-uplets de *SUPER*(DEL) agissent comme des *exceptions explicites* dans le processus de déduction défini par *SUB*. On définit donc la sémantique d'une base de données comme étant la paire

$$(SUB(\text{INS}) \setminus SUPER(\text{DEL}) \ , \ SUPER(\text{DEL})).$$

En conséquence, afin de préserver la consistance de la base de données à mettre à jour, les insertions et suppressions doivent être effectuées de la manière suivante :

insérer t : placer t dans la base, et
retirer de la base tous les sous-n-uplets marqués de t.

supprimer t : placer t marqué dans la base, et
retirer de la base tous les sur-n-uplets non marqués de t.

À partir des définitions ci-dessus, il est important de remarquer que (*a*) les insertions et suppressions sont des opérations déterministes, (*b*) si la base de données est consistante, elle le reste après une insertion ou une suppression, et (*c*) l'insertion ou la suppression d'un n-uplet implique des changements concernant d'autres n-uplets, *i.e.* la base de données doit être "active" afin de préserver la consistance.

Bases de Données Déductives

Nous décrivons maintenant, toujours de manière informelle, notre approche dans le contexte des bases de données déductives, sous l'hypothèse qu'il est possible de stocker n'importe quel fait, qu'il soit sur un prédicat extensionnel ou sur un prédicat intensionnel.

Une base de données est alors vue comme une paire constituée d'un ensemble de faits et d'un ensemble de règles, paire dans laquelle on peut insérer ou supprimer n'importe quel fait. Comme précédemment, on ne retire pas *physiquement* de la base les faits à supprimer : lors de la suppression d'un fait, celui-ci est *marqué* comme étant supprimé. Ainsi, les faits d'une base de données peuvent être divisés en deux parties disjointes : les faits *insérés*, dont l'ensemble est noté INS et les faits *supprimés* (donc marqués), dont l'ensemble est noté DEL.

Par conséquent, comme dans le cas du modèle relationnel, lorsqu'un fait

tel que $assign(John, CS, A)$ est inséré dans la base, ce fait est considéré comme étant vrai. De même, lorsqu'un fait est supprimé de la base, ce fait est considéré comme étant faux. Toutefois, dans ce contexte, les inconsistances n'apparaissent que lorsqu'un fait donné est à la fois inséré et supprimé. En d'autres termes, il n'y a pas ici de notions de "sous-faits" ou de "sur-faits." Par conséquent, une base de données est consistante si $\text{INS} \cap \text{DEL} = \emptyset$.

Il est alors facile de voir que la consistance de la base est maintenue au cours des mises à jour, si celles-ci sont effectuées comme suit :

insérer f : placer f non marqué dans la base.
supprimer f : placer f marqué dans la base.

On voit donc que les faits de INS sont considérés comme étant vrais et que les faits de DEL sont considérés comme étant faux. De plus, la dérivation de nouveaux faits vrais est assurée par les règles, à partir des faits insérés et des faits supprimés présents dans la base. Par exemple, en présence de la règle $assign(E, D, P) \leftarrow works\_in(E, D), works\_for(E, P)$, et des faits insérés $works\_in(John, CS)$ et $works\_for(John, A)$, on peut dériver que le fait $assign(John, CS, A)$ est vrai.

Ce type de dérivation est assuré grâce au calcul de plus petit point fixe de l'opérateur bien connu dit de "conséquence immédiate" ([Ull89, CGT90, Bid91, Bid92]). Si de plus, on autorise les corps des règles à contenir des négations, on considère alors le processus de dérivation comme étant défini à partir de *deux* opérateurs, que nous notons pour l'instant POS et NEG, et qui calculent respectivement des faits vrais et des faits faux (comme dans le cas de la sémantique bien fondée [Bid91, VRS91]).

Il faut néanmoins noter que ces dérivations peuvent conduire à des inconsistances, même si la base est consistante. En effet, à partir de

- la règle $assign(E, D, P) \leftarrow works\_in(E, D), works\_for(E, P)$,
- des faits insérés $works\_in(John, CS)$ et $works\_for(John, A)$, et
- du fait supprimé $assign(John, CS, A)$,

le fait $assign(John, CS, A)$ apparaît comme étant à la fois vrai et faux. Le fait supprimé $assign(John, CS, A)$ est dans ce cas vu comme une *exception* à la règle de dérivation ci-dessus.

Afin de prendre correctement en compte ces exceptions, on définit la sémantique d'une base de données comme étant la limite de la suite :

- $SEM^0 = \text{INS} \cup \neg.\text{DEL}$, et
- $SEM^{i+1} = (POS(SEM^i) \setminus \text{DEL}) \cup \neg.(NEG(SEM^i) \cup \text{DEL})$, pour $i > 0$.

($\neg.F$ désigne l'ensemble $\{\neg f \mid f \in F\}$, pour tout ensemble de faits F.)

Cette manière de définir la sémantique d'une base rend compte de l'intuition que les faits supprimés agissent comme des *exceptions* aux règles de dérivation. Il existe donc deux manières de déduire qu'un fait f est faux : soit f est une exception (parce que f est un fait supprimé), soit l'opérateur NEG permet de dériver f, en utilisant les faits insérés, les faits supprimés et les règles.

On notera que, contrairement au cas du modèle relationnel, les exceptions (*i.e.* les faits supprimés) contribuent au calcul de faits vrais. Par exemple, le fait supprimé $works\_in(John, CS)$ contribue à la dérivation de $works\_hard(John)$ en présence de la règle $works\_hard(E) \leftarrow \neg works\_in(E, CS)$.

On a ainsi pu voir dans le cadre des interfaces de type relation universelle et dans le cadre des bases de données du type Datalog^{neg} que l'on obtient des mises à jour déterministes si (a) on autorise toute information à pouvoir être stockée, et si (b) on garde trace des informations supprimées.

Cet article a pour but de montrer qu'il s'agit en réalité de deux cas particuliers d'une même théorie mathématique dont les idées se démarquent de manière significative des approches traditionnelles de bases de données, approches dans lesquelles seule l'information insérée est utilisée.

L'article est organisé comme suit. Dans la Section 2, nous présentons le modèle formel selon lequel sont définis la sémantique d'une base de données ainsi que les insertions et les suppressions de n-uplets (faits). Dans la Section 3, notre formalisme est appliqué aux interfaces de type relation universelle et aux bases de données de type Datalog^{neg}, montrant ainsi que ces deux modèles sont deux cas particuliers de notre approche générale. Enfin, dans la Section 4, nous concluons l'article en évoquant notamment nos travaux actuels concernant une extension de cette approche.

2 Le Modèle Formel

2.1 Définitions et Notations

On considère un ensemble T dont les éléments sont appelés *articles*. Dans le cas d'une base de données relationnelle, T est l'ensemble des n-uplets que l'on peut former à partir des constantes des domaines des attributs de l'univers. Dans le cas d'une base de données Datalog^{neg}, T est l'ensemble des faits de la base de Herbrand sous-jacente. De plus, une *a_paire* P est une paire (I, J) où I et J sont des ensembles d'articles.

Définition 1. *Soit $P = (I, J)$ une a_paire. I et J sont respectivement appelés* partie positive *et* partie négative *de P, et on note: $pos(P) = I$ et $neg(P) = J$. De plus, P est* consistante *si $pos(P) \cap neg(P) = \emptyset$.* □

On compare les a_paires comme suit : $P = (I, J)$ est *inférieure ou égale* à $P' = (I', J')$, noté $P \sqsubseteq P'$, si $I \subseteq I'$ et $J \subseteq J'$.

Une base de données Δ est un *quintuplet* $(\text{INS}, \text{DEL}, \xi, \pi, \nu)$ où INS et DEL sont des ensembles d'articles et où ξ, π, ν sont des fonctions opérant sur les articles et respectivement appelées *opérateur d'exception, opérateur positif* et *opérateur négatif.*

Définition 2. *Une* base de données *est un quintuplet* $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ *tel que:*

- INS *est un ensemble d'articles (l'ensemble des articles insérés).*
- DEL *est un ensemble d'articles (l'ensemble des articles supprimés).*
- ξ *prend en argument un ensemble d'articles et retourne un ensemble d'articles. On suppose que:*
  - *pour tout ensemble d'articles I, $I \subseteq \xi(I)$, et*
  - *ξ est union-compatible: pour tous ensembles d'articles I et I', $\xi(I \cup I') = \xi(I) \cup \xi(I')$.*
- *π et ν prennent chacun en argument une a_paire et retournent un ensemble d'articles. On suppose que:*
  - $\text{INS} \subseteq \pi(\text{INS}, \text{DEL})$,
  - *π et ν sont monotones: pour toutes a_paires P et P', $P \sqsubseteq P' \Rightarrow \pi(P) \subseteq \pi(P')$ et $\nu(P) \subseteq \nu(P')$, et*
  - *π et ν sont mutuellement consistants: si P est consistante, alors $(\pi(P), \nu(P))$ est consistante.* □

Les trois opérateurs de la Définition 2 sont utilisés pour calculer la sémantique d'une base de données selon le processus suivant: ξ calcule, à partir de DEL, l'ensemble des articles qui agiront en tant qu'exceptions dans le processus de dérivation, lui-même défini par les opérateurs π et ν.

D'après la définition ci-dessus, il est aisé de montrer que l'opérateur ξ est monotone (à cause de la propriété d'union-compatibilité). Par conséquent, la suite définie par:

$\xi^0(\text{DEL}) = \text{DEL}$,
$\xi^\alpha(\text{DEL}) = \xi(\xi^{\alpha-1}(\text{DEL}))$, pour tout ordinal α ayant un prédécesseur,
$\xi^\alpha(\text{DEL}) = \bigcup_{\beta<\alpha} \xi^\beta(\text{DEL})$, pour tout ordinal limite α

a une limite, que l'on appelle *plus petit point fixe de ξ par rapport à* DEL, et que l'on note $lfp(\xi(\text{DEL}))$, ou ξ_Δ. De plus, grâce aux propriétés de ξ, on a: $\xi_\Delta = \bigcup_{\tau \in \text{DEL}}(lfp(\xi(\{\tau\})))$ et $\text{DEL} \subseteq \xi_\Delta$.

2.2 Sémantique d'une Base de Données

Soit $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ une base de données. On définit un opérateur σ qui prend en argument une a_paire P et qui retourne une a_paire $\sigma(P)$ définie par :

$$pos(\sigma(P)) = \pi(P) \setminus \xi_\Delta \qquad \text{et} \qquad neg(\sigma(P)) = \nu(P) \cup \xi_\Delta.$$

Donc, pour calculer $\sigma(P)$, on calcule d'abord ξ_Δ. Ensuite, $pos(\sigma(P))$ et $neg(\sigma(P))$ sont obtenus en retirant de $\pi(P)$ les articles de ξ_Δ et en ajoutant ces articles dans $\nu(P)$. Le théorème suivant donne des propriétés fondamentales de σ.

Théorème 1. *Soit $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ une base de données. Alors :*
1. *σ est monotone, i.e. $P \sqsubseteq P' \Rightarrow \sigma(P) \sqsubseteq \sigma(P')$.*
2. *Si P est consistante, alors $\sigma(P)$ est consistante.*
3. *$\text{DEL} \subseteq neg(\sigma(\text{INS}, \text{DEL}))$, et $\text{INS} \cap \xi_\Delta = \emptyset \Rightarrow \text{INS} \subseteq pos(\sigma(\text{INS}, \text{DEL}))$.* □

L'ensemble des a_paires muni du préordre $\sqsubseteq$ défini précédemment est un treillis complet si, pour toutes a_paires $P = (I, J)$ et $P' = (I', J')$, on définit $lub(P, P') = (I \cup I', J \cup J')$ et $glb(P, P') = (I \cap I', J \cap J')$, où *lub* et *glb* désignent respectivement le plus petit majorant et le plus grand minorant. Donc, puisque σ est monotone, la suite

$$\sigma^0(\text{INS}, \text{DEL}) = (\text{INS}, \text{DEL}),$$
$$\sigma^\alpha(\text{INS}, \text{DEL}) = \sigma(\sigma^{\alpha-1}(\text{INS}, \text{DEL})),$$
pour tout ordinal α ayant un prédécesseur,
$$\sigma^\alpha(\text{INS}, \text{DEL}) = \left(\bigcup_{\beta<\alpha} pos(\sigma^\beta(\text{INS}, \text{DEL})), \bigcup_{\beta<\alpha} neg(\sigma^\beta(\text{INS}, \text{DEL}))\right),$$
pour tout ordinal limite α

a une limite appelée *plus petit point fixe de σ par rapport à* (INS, DEL). Ce point fixe définit ce qui est appelé la *sémantique* de Δ.

Définition 3. *Soit $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ une base de données. Le plus petit point fixe de σ par rapport à (INS, DEL) est appelé* sémantique de Δ, *et est noté σ_Δ.* □

Le Théorème 1 admet le corollaire suivant :

Corollaire 1. *Pour toute base de données $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ on a :*
1. *Si (INS, DEL) est consistante alors σ_Δ est consistante.*
2. *Si $\text{INS} \cap \xi_\Delta = \emptyset$ alors $\text{INS} \subseteq pos(\sigma_\Delta)$ et $\text{DEL} \subseteq neg(\sigma_\Delta)$.* □

On définit alors la consistance d'une base de données comme suit :

Définition 4. *Une base de données* $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ *est dite* consistante *si* $\text{INS} \cap \xi_\Delta = \emptyset$. □

Par conséquent, si Δ est consistante alors σ_Δ est consistante et, dans ce cas, les articles de $pos(\sigma_\Delta)$ sont dits *vrais* et les articles de $neg(\sigma_\Delta)$ sont dits *faux*, relativement à Δ. De plus, les articles qui ne sont ni vrais ni faux sont dits *inconnus*, relativement à Δ.

On notera donc que notre approche est fondée sur une logique à *trois valeurs*, contrairement à la plupart des autres approches qui, en utilisant l'hypothèse du monde clos [Rei78], ignorent la notion d'inconnu.

2.3 Sémantique des Mises à Jour

Étant donné une base de données consistante Δ et un article τ, nous définissons la manière de modifier Δ pour insérer ou supprimer τ.

Définition 5 - Mise à Jour. *Soit* $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ *une base de données consistante et* τ *un article.*
Insertion : *Pour insérer* τ *dans* Δ, *effectuer les modifications suivantes :*

- *Placer* τ *dans l'ensemble* INS.
- *Retirer de l'ensemble* DEL *tous les articles* μ *tels que* $\tau \in lfp(\xi(\{\mu\}))$.

On note $ins(\Delta, \tau)$ *la base de données ainsi obtenue.*
Suppression : *Pour supprimer* τ *de* Δ, *effectuer les modifications suivantes :*

- *Retirer de l'ensemble* INS *tous les articles de* $lfp(\xi(\{\tau\}))$.
- *Placer* τ *dans l'ensemble* DEL.

On note $del(\Delta, \tau)$ *la base de données ainsi obtenue.* □

La proposition suivante montre que les mises à jour sont effectuées de manière à préserver la consistance.

Proposition 1. *Soit* Δ *une base de données et* τ *un article. Alors :*
1. *Si* τ *est inséré dans* Δ, *alors* $\tau \in pos(\sigma_{ins(\Delta,\tau)})$.
2. *Si* τ *est supprimé de* Δ, *alors* $\tau \in neg(\sigma_{del(\Delta,\tau)})$.
3. *Si* Δ *est consistante alors* $ins(\Delta, \tau)$ *et* $del(\Delta, \tau)$ *sont consistantes.* □

3 Applications

3.1 Interfaces de Type Relation Universelle

Dans les interfaces de type relation universelle, une base de données sur un univers U peut être vue comme une paire (δ, F) où δ est un ensemble de n-uplets sur des schémas de U et F un ensemble de dépendances fonctionnelles

sur U. On suppose de plus que l'on peut insérer ou supprimer de la base n'importe quel n-uplet.

Afin d'appliquer notre approche dans ce contexte, les articles sont ici les n-uplets qu'il est possible de former à partir des symboles des domaines des attributs de U. De plus, nous considérons que, partant de la base vide, il a été gardé trace des insertions *et* des suppressions des n-uplets dans la base depuis sa création. On définit ainsi une base de données "auxilliaire" $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ de la manière suivante :

- INS est un ensemble de n-uplets, supposé être l'ensemble des n-uplets insérés dans la base et qui n'ont pas été supprimés. Donc $\text{INS} = \delta$.
- DEL est un ensemble de n-uplets, supposé être l'ensemble des n-uplets qui ont été supprimés de la base et qui n'ont pas été réinsérés.
- Pour tout I, $\xi(I)$ est l'ensemble des sur-n-uplets des n-uplets de I.
- π est défini par : pour toute a_paire $P = (I, J)$,
$$\pi(P) = \{q \mid \exists \tau \in I : q \text{ est un sous-n-uplet de } \tau\} \cup \{xya \mid xy, xa \in I, X \rightarrow A \in F\}.$$
- ν est défini par : pour toute a_paire $P = (I, J)$, $\nu(P) = \emptyset$.

Il est facile de voir que ξ est monotone, union-compatible et que $I \subseteq \xi(I)$ est satisfait pour tout ensemble de n-uplets I. De plus, π et ν sont monotones et mutuellement consistants, et π satisfait $I \subseteq \pi(P)$, pour toute a_paire $P = (I, J)$.

Par suite, le formalisme introduit dans la Section 2 s'applique ici, et on définit la sémantique de δ comme étant la sémantique de Δ_δ.

On notera de plus que π correspond à l'opérateur relationnel de projection-jointure, où la jointure est conditionnée par l'existence d'une dépendance fonctionnelle appropriée. Il a par ailleurs été montré dans [LPS92b] que π calcule les n-uplets obtenus par la procédure d'"extension chase" dans le cadre de la sémantique des *instances faibles* [Mai83].

D'autre part, le fait que, dans le modèle relationnel classique, seules les informations positives sont prises en compte se traduit ici par le fait que ν retourne toujours l'ensemble vide.

En appliquant la Définition 4, une base $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ est consistante si INS ne contient aucun sur-n-uplet d'un n-uplet de DEL.

Exemple 1. Soit $U = \{A, B, C, D\}$ un univers relationnel, et soit la base (δ, F) sur U définie par : $\delta = \{ac, cd, ac', c'd\}$, et par $F = \{B \rightarrow D, C \rightarrow D, AD \rightarrow C\}$.

Par conséquent, les n-uplets ac, cd, ac' et $c'd$ ont été insérés dans la base et n'ont pas été supprimés. Supposons de plus que les n-uplets bd et

$ac'd$ aient été supprimés de la base de données depuis sa création. La base auxilliaire $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ est donc définie par :

$$\text{INS} = \{ac, cd, ac', c'd\} \qquad \text{et} \qquad \text{DEL} = \{bd, ac'd\}$$

(ξ, π et ν étant définis comme indiqué précédemment). Ainsi, ξ_Δ contient tous les sur-n-uplets de bd et $ac'd$, ce qui montre que INS et ξ_Δ sont disjoints. Donc Δ est consistante et on peut vérifier que σ_Δ est définie par :

$$pos(\sigma_\Delta) = \{a, c, c', d, ac, cd, ac', c'd, acd, ad\} \quad \text{et} \quad neg(\sigma_\Delta) = \xi_\Delta. \qquad \square$$

On notera que le rôle des dépendances fonctionnelles est ici double, puisque (1) les dépendances fonctionnelles agissent en tant que *règles de dérivation* dans la définition de l'opérateur π, et (2) les dépendances fonctionnelles agissent en tant que *contraintes* qui doivent être satisfaites par les n-uplets de $pos(\sigma_\Delta)$.

Par conséquent, les insertions sont conditionnées par un test effectué sur la sémantique $\sigma_{\Delta'}$ de la base mise à jour Δ'. Les suppressions n'engendrant jamais de contradictions par rapport aux dépendances fonctionnelles, aucun test n'est nécessaire pendant leur exécution. On a donc :

Insertion – Pour insérer τ dans δ :

1. Placer τ dans l'ensemble INS.
2. Retirer de l'ensemble DEL tous les sur-n-uplets de τ.
3. Si $pos(\sigma_{\Delta'})$ satisfait les dépendances de F alors accepter l'insertion, sinon rejeter l'insertion.

Suppression – Pour supprimer τ de δ :

1. Retirer de l'ensemble INS les sur-n-uplets de τ.
2. Placer τ dans l'ensemble DEL.

Exemple 1. (suite) Insérons le n-uplet abd dans la base de données (δ, F) définie précédemment. bd étant un sous-n-uplet de abd, bd est retiré de DEL. La base auxilliaire $\Delta' = (\text{INS}', \text{DEL}', \xi, \pi, \nu)$ résultant de l'insertion est donc définie par :

$$\text{INS}' = \{ac, cd, ac', c'd, abd\} \qquad \text{et} \qquad \text{DEL}' = \{ac'd\};$$

et la sémantique $\sigma_{\Delta'}$ de Δ' est alors :

$$pos(\sigma_{\Delta'}) = \quad pos(\sigma_\Delta) \cup \{abd, ab, bd, b\} \quad \text{et}$$
$$neg(\sigma_{\Delta'}) = \quad \xi_{\Delta'} = \{q \mid q \text{ est un sur-n-uplet de } ac'd\}.$$

Comme les dépendances fonctionnelles de F sont satisfaites par les n-uplets de $pos(\sigma_{\Delta'})$, la mise à jour est acceptée, et a pour résultat la base Δ'.

Supprimons le n-uplet a de la base (δ, F). Comme a est un sous-n-uplet de ad et ac', ceux-ci sont retirés de INS. La base auxilliaire $\Delta'' = (\text{INS}'', \text{DEL}'', \xi, \pi, \nu)$ résultant de la suppression est donc définie par :

$$\text{INS}'' = \{cd, c'd\} \qquad \text{et} \qquad \text{DEL}'' = \{bd, ac'd, a\};$$

et sa sémantique $\sigma_{\Delta''}$ est alors définie par :

$pos(\sigma_{\Delta''}) =$ INS$''$ et

$neg(\sigma_{\Delta''}) = \xi_{\Delta''} = \{q \mid q$ est un sur-n-uplet de bd ou de $a\}$. □

3.2 Bases de Données de Type Datalog^{neg}

Dans les approches traditionnelles ([Ull89, CGT90, Bid91]), une base de données Datalog^{neg} δ est une paire (F, R) où F est un ensemble de faits et où R est un ensemble de règles saines sur un alphabet **A**. Si $\delta = (F, R)$ est une telle base de données, on note $inst\_\delta$ l'ensemble de toutes les règles obtenues en instanciant complètement les règles de R grâce aux constantes présentes dans la base. Pour chaque règle r de $inst\_\delta$, $head(r)$ et $body(r)$ désignent respectivement la tête et le corps de r.

Nous rappelons maintenant les points essentiels concernant la sémantique bien fondée de [VRS91]. Étant donné un alphabet **A**, on appelle *interprétation partielle* de **A** tout ensemble consistant de littéraux positifs ou négatifs, c'est-à-dire tout ensemble de la forme $Int = I \cup \neg.J$, où I et J sont des ensembles de faits tels que $I \cap J = \emptyset$ et où $\neg.J = \{\neg f \mid f \in J\}$.

Si $\delta = (F, R)$ est une base de données sur un alphabet **A**, le *modèle bien fondé* de δ est une interprétation partielle de **A** définie comme étant un plus petit point fixe calculé à partir de deux opérateurs $T^{\in}$ et GUS dont nous rappelons les définitions ci-dessous :

1. L'opérateur de conséquence bien fondée, noté $T^{\in}$ et défini par
$$T^{\in}(Int) = \{head(r) \mid r \in inst\_\delta, body(r) \subseteq Int\},$$
pour toute interprétation partielle Int.
2. Si Int est une interprétation partielle, le plus grand ensemble de faits non fondés par rapport à Int, noté $GUS(Int)$, est l'union de tous les ensembles U dont les éléments f satisfont :
$$(\forall r \in inst\_\delta)(head(r) = f \Rightarrow \exists L \in body(r) : \neg L \in Int \vee L \in U).$$

Le modèle bien fondé de δ est alors la limite de la suite d'interprétations partielles définie par : $Int_1 = T^{\in}(\emptyset) \cup \neg.GUS(\emptyset)$, et pour $i > 1$, $Int_i = T^{\in}(Int_{i-1}) \cup \neg.GUS(Int_{i-1})$.

Afin d'appliquer le formalisme présenté dans la Section 2 dans ce contexte, on remarque tout d'abord que les articles sont ici les faits de la base de Herbrand sous-jacente. Ainsi, toute interprétation partielle $I \cup \neg.J$ peut être associée à une a_paire consistante P définie par $pos(P) = I$ et par $neg(P) = J$.

De plus, $\delta = (F, R)$ étant une base de données de type Datalog^{neg}, on définit une base de données "auxilliaire" $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ où il a été gardé trace des insertions *et* des suppressions depuis la création de la base, *i.e.* à partir de la base vide. Cette base auxilliaire Δ est définie par :

- INS est un ensemble de faits, supposé être l'ensemble des faits insérés qui n'ont pas été supprimés. Donc INS est l'ensemble F.
- DEL est un ensemble de faits, supposé être l'ensemble des faits qui ont été supprimés de la base et qui n'ont pas été réinsérés.
- ξ est défini par : $\xi(I) = I$, pour tout I.
- π est défini par : $\pi(P) = T^{\in}(I \cup \neg.J)$, où $P = (I, J)$ est une a_paire.
- ν est défini par : $\nu(P) = GUS(I \cup \neg.J)$, où $P = (I, J)$ est une a_paire.

Il est immédiat de voir que ξ est monotone, union-compatible et que $I \subseteq \xi(I)$ est satisfait pour tout ensemble de faits I. De plus, d'après les résultats de [VRS91], π et ν définis ci-dessus sont monotones, mutuellement consistants et, pour toute interprétation partielle $I \cup \neg.J$, tout fait inséré τ appartient à $T^{\in}(I \cup \neg.J)$. Par conséquent, π satisfait $\text{INS} \subseteq \pi(\text{INS}, \text{DEL})$.

Par suite le formalisme introduit dans la Section 2 s'applique ici et on définit la sémantique de δ comme étant la sémantique de Δ. Comme ξ est la fonction identité, pour tout fait τ, on a : $lfp(\xi(\{\tau\})) = \{\tau\}$. Par conséquent, d'après la Définition 4, la base $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ est consistante si $\text{INS} \cap \text{DEL} = \emptyset$.

Exemple 2. Considérons une base de données $\delta = (F, R)$ où $F = \{r(a, b), s(bc)\}$ et où R est l'ensemble des règles suivantes :

$$R : \quad \begin{array}{l} q(x, y, z) \leftarrow r(x, y), s(y, z) \\ p(x, z) \leftarrow q(x, y, z) \\ t(x, z) \leftarrow q(x, y, z), \neg p(x, z) \end{array}$$

Si l'on suppose de plus que le fait $p(a, c)$ a été supprimé de δ, on définit la base auxilliaire $\Delta = (\text{INS}, \text{DEL}, \xi, \pi, \nu)$ par :

$$\text{INS} = \{r(a, b), s(bc)\} \qquad \text{et} \qquad \text{DEL} = \{p(a, c)\}$$

(et ξ, π, ν comme ci-dessus). On a donc $\xi_\Delta = \{p(a, c)\}$ et, par conséquent Δ est consistante. De plus, le calcul de la sémantique σ_Δ de Δ donne le résultat suivant :

$$pos(\sigma_\Delta) = \{r(a, b), s(b, c), q(a, b, c), t(a, c)\} \text{ et } neg(\sigma_\Delta) = HB \setminus pos(\sigma_\Delta)$$

où HB désigne la base de Herbrand sous-jacente. □

Comme l'on a ici $lfp(\xi(\{\tau\})) = \{\tau\}$ pour tout fait τ, la Définition 5 s'applique de la manière suivante :

Insertion – Pour insérer τ dans δ :

1. Placer τ dans l'ensemble INS.
2. Retirer τ de l'ensemble DEL.

Suppression – Pour supprimer τ de δ :

1. Retirer τ de l'ensemble INS.
2. Placer τ dans l'ensemble DEL.

Exemple 2. (suite) Insérons le fait $p(a,c)$ dans la base δ définie ci-dessus. Comme $p(a,c)$ est dans DEL, on doit le retirer de DEL. La base auxilliaire résultant de cette insertion, notée $\Delta' = (\text{INS}', \text{DEL}', \xi, \pi, \nu)$ est définie par $\text{INS}' = \{r(a,b), s(b,c), p(a,c)\}$ et $\text{DEL}' = \emptyset$. La sémantique $\sigma_{\Delta'}$ de Δ' est : $pos(\sigma_{\Delta'}) = \{r(a,b), s(b,c), q(a,b,c), p(a,c)\}$ et $neg(\sigma_{\Delta'}) = HB \setminus pos(\sigma_{\Delta'})$.

Si l'on supprime maintenant le fait $r(a,b)$ de la base δ, on doit retirer ce fait de INS. La base auxilliaire résultant de cette suppression, notée $\Delta'' = (\text{INS}'', \text{DEL}'', \xi, \pi, \nu)$ est donc définie par $\text{INS}'' = \{s(b,c)\}$ et $\text{DEL}'' = \{p(a,c), r(a,b)\}$. La sémantique $\sigma_{\Delta''}$ de Δ'' est :

$$pos(\sigma_{\Delta''}) = \text{INS}'' \qquad \text{et} \qquad neg(\sigma_{\Delta''}) = HB \setminus pos(\sigma_{\Delta''}). \qquad \square$$

4 Conclusions et Perspectives

Nous avons présenté une nouvelle approche des bases de données dont l'idée principale est d'utiliser les informations insérées *et* les informations supprimées. Les informations supprimées sont utilisées comme des *exceptions* lors du calcul des réponses aux requêtes.

Une conséquence majeure de notre approche est qu'elle offre une solution *générale* au problème du non-déterminisme des mises à jour. Cette approche, d'abord exprimée dans un formalisme général, a ensuite été appliquée au cas des interfaces de type relation universelle et au cas des bases de données déductives du type Datalog^{neg}.

Nous étudions actuellement une généralisation de la notion d'exception de façon à engendrer des exceptions positives *et* des exceptions négatives lors de chaque mise à jour [HLS94]. Cette généralisation vise à rendre la base de données *active*, en ce sens que, pour chaque mise à jour, le calcul des exceptions engendre les informations qui doivent devenir vraies et les informations qui doivent devenir fausses à cause de cette mise à jour.

Références

[**AtT89**] P. Atzeni, R. Torlone, "Updating Databases in the Weak Instance Model," ACM PODS, 1989.

[**AtT92**] P. Atzeni, R. Torlone, "Updating Intensional Predicates in Datalog," *Data & Knowledge Engineering,* 8, 1992.

[**Bid91**] N. Bidoit, "Negation in Rule-Based Database Languages: a Survey," *TCS,* 78, 1991.

[**Bid92**] N. Bidoit, *Bases de Données Déductives, Présentation de Datalog,* Armand Colin, 1992.

[**Bry90**] F. Bry, "Intensional Updates: Abduction via Deduction," Intl. Symposium on Logic Programming, 1990.

[**CGT90**] S. Ceri, G. Gottlob, L. Tanca, *Logic Programming and Databases,* Surveys in Computer Science, Springer-Verlag, 1990.

[**GMR92**] G. Grahne, A.O. Mendelzon, P.Z. Revesz, "Knowledgebase Transformations," ACM PODS, 1992.

[**HLS94**] M. Halfeld Ferrari Alves, D. Laurent and N. Spyratos, "Passive and Active Rules in Deductive Databases," à paraître dans Mathematical Foundations of Computer Science, MFCS'94, *LNCS,* Springer-Verlag, 1994.

[**LaS94**] D. Laurent, N. Spyratos, "A Partition Approach to Updating Universal Scheme Interfaces," *IEEE Trans. on Knowledge and Data Eng.,* 6(2), 1994.

[**LPS92a**] D. Laurent, V. Phan Luong, N. Spyratos, "Deleted Tuples are Useful when Updating through Universal Scheme Interfaces," 8th *IEEE* ICDE, Phoenix, 1992.

[**LPS92b**] D. Laurent, V. Phan Luong, N. Spyratos, "The Use of Deleted Tuples in Database Querying and Updating," Rapport de Recherche *LIFO* n° 92-5 et *LRI* n° 757, 1992.

[**LPS93a**] D. Laurent, V. Phan Luong, N. Spyratos, "Updating Intensional Predicates in Deductive Databases," 9th *IEEE* ICDE, Vienne, 1993.

[**LPS93b**] D. Laurent, V. Phan Luong, N. Spyratos, "Database Updating Revisited," in Intl. Conf. DOOD, *LNCS* 760, Springer-Verlag, 1993.

[**LeS88**] Ch. Lécluse, N. Spyratos, "Implementing Queries and Updates on Universal Scheme Interfaces," VLDB Intl. Conf., Los Angeles, 1988.

[**Mai83**] D. Maier, *The Theory of Relational Databases,* Computer Science Press, 1983.

[**Rei78**] R. Reiter,"On closed-world databases," In H. Gallaire and J. Minker, editors, *Logic and Data Bases,* Plenum Press, New York, 1978.

[**Ull89**] J.D. Ullman, *Principles of Databases and Knowledge Base Systems,* Computer Science Press, 1989.

[**VRS91**] A. Van Gelder, K.A. Ross, J.S. Schlipf, "The Well-founded Semantics for General Logic Programs," *J. of the ACM,* 38(3), 1991.

[**Win88**] M. Winslett, "A Model-Based Approach to Updating Databases with Incomplete Information," ACM *TODS,* 13(2), 1988.