UNIVERSITE D'ORLEANS
Faculté des Sciences
LIFO

Laboratoire d'Informatique Fondamentale d'Orléans
4, rue Léonard de Vinci, BP 6759
F-45067 Orléans Cedex 2
FRANCE

Rapport de Recherche

www : http://www.univ-orleans.fr/SCIENCES/LIFO/

Translating View Updates Using Inversion of Relational Algebra

Fadila Bentayeb [†], Dominique Laurent [‡]
[†] LIFO, Université d'Orléans
[‡] LI/E3I, Antenne universitaire de Blois
Université de Tours

Rapport N° **98-03**

Translating View Updates Using Inversion of Relational Algebra

Fadila Bentayeb
LIFO - IIIA
Université d'Orléans
4, rue Léonard De Vinci - BP 6759
F-45067 Orléans Cedex 02 - France
bentayeb@lifo.univ-orleans.fr

Dominique Laurent
LI/E3I - Université de Tours
Antenne universitaire de Blois
3, place Jean Jaurès
F-41000 Blois - France
laurent@univ-tours.fr

Abstract

Views over databases have been studied in various directions for many years. Among these directions, translating view updates in terms of updates on the base relations [BS81, IL83], and materialized views and their maintenance [BLT86, CW91] have motivated many research efforts. Nowadays, materialized views are finding increased research activity with applications in decision support which is relevant for data warehousing. In this paper, we study the translation process in the relational model. We propose a method for characterizing translations of view updates, based on the notion of inverse of a relational expression. Moreover, we characterize two kinds of updates: (1) deterministic updates regardless to the database state, and (2) deterministic updates according to some database state, and finally, we give an overview of performing view updates using the criterion of updates determinism.

Keywords: relational views, view update translations, inverses, inverse-images, strongly deterministic view updates, weakly deterministic view updates.

Résumé

Les vues relationnelles ont fait l'objet de plusieurs études depuis plusieurs années. Parmi ces études, la traduction des mises à jour de vues en termes de mises à jour sur les relations de la base [BS81, IL83] et la maintenance de vues matérialisées [BLT86, CW91] ont motivé plusieurs chercheurs. De nos jours, les vues matérialisées accusent un essort considérable dans le domaine du data warehousing. Dans ce papier, nous étudions le processus de traduction de mises à jour au travers de vues. Nous proposons une méthode qui caractérise ces traductions. Cette méthode est basée sur la notion d'inverse d'une expression relationnelle. De plus, nous caractérisons deux sortes de mises à jour : (1) des mises à jour déterministes indépendamment de l'état de la base, et (2) des mises à jour déterministes par rapport à un état donné de la base.

Mots clés : vues relationnelles, traductions de mises à jour au travers de vues, inverses, images réciproques, mises à jour fortement déterministes, mises à jour faiblement déterministes.

1 Introduction

In relational databases, a view is generally seen as a query definition whose name is stored in the database dictionary, and whose evaluation is *computed* each time the view is used. The user interacts with a view by issuing queries and update requests. The view definition mapping is sufficient to translate view queries into database queries. View updates, however, present a difficult problem, referred to as non-determinism in the literature [BS81, AT92b, AT92a, LS94]. This problem can be stated as follows: Given a view v and an update $\mathcal{U}$ on v, there exists in general more than one way to translate $\mathcal{U}$ into updates on the relations explicitly stored in the database.

The problem of view update naturally arises in data warehousing environments. A data warehouse is a large data store, into which, data from several, possibly heterogeneous sources (operational databases), is integrated, and used in organizational decision making. Hence, the data warehouse can be seen as a materialized view over the underlying database(s). If we assume that decisions taken on the current warehouse have an impact on the underlying databases, this implies that:

1. decisions on the data warehouse can be seen as updates on a view considered as the data warehouse, and
2. the impact of the decisions can be seen as translations of view updates into updates on the underlying database.

The objective of this paper is to study the view update translation in the context of the relational model and not only, as in [AT92b, LPS92, LS94], in the context of weak instances. We show that there is a strong relationship between view update translations and the inverse of the relational expression (excluding the difference) which defines the view. A formalism and an algorithm for computing this inverse is given. Moreover, two kinds of translations are characterized: (1) deterministic updates regardless to the database state, called strongly deterministic updates, and (2) deterministic updates according to some database state, called weakly deterministic updates.

Our contribution to the view update problem is twofold:

1. allow more view updates than usually allowed in the actual systems based on the relational model and, in case in of non deterministic translation, propose all solutions to the user,
2. define a new approach of view updating that allows for *all* view updates in a deterministic way [BL97].

In this paper, we focus on point 1 above. The interested reader is referred to [BL97] for more details about item 2.

We consider view updates which are insertions or deletions of a given tuple over the view. Let us illustrate our approach by an example that we shall use as our Running Example throughout the paper.

Running Example: Consider an employee relation *emp* with attributes $Ename, Pname$ and a department relation *dept* with attributes $Ename, Dname$.

Assume that the database instance is as follows: $emp = \{\langle John, P\rangle\}$ and $dept = \{\langle John, M\rangle\}$, meaning that employee $John$ works in project P and that employee $John$ works in department M, respectively. Consider now the view v defined by $\Pi_{Ename,Dname}(\sigma_{Pname=P}(emp \bowtie dept))$.

First, consider inserting in v the tuple $\langle Mary, CS\rangle$ (meaning: employee Mary works in department CS). Since the view v concerns employees working in project P, the *only* way to perform this update is as follows: (1) insert $\langle Mary, P\rangle$ in *emp*, and (2) insert $\langle Mary, CS\rangle$ in *dept*. The database updates (1) and (2) form the update translation. In this case, the insert view update is deterministic and this, regardless to the database instance, since there is one unique translation to perform it.

Note that $(\langle Mary, P\rangle, \langle Mary, CS\rangle)$ is the inverse-image of $\langle Mary, CS\rangle$ with respect to v, in the sense that computing v with these two tuples as input, produces the tuple to be inserted in the view.

Next, consider the view update which consists in deleting the tuple $(\langle John, M\rangle)$ from v, meaning that it is no more true that $John$ works in project P (because of the select condition in the view definition) and in department M. There are three ways to perform this deletion: either (1) delete $\langle John, P\rangle$ from *emp*, or (2) delete $\langle John, M\rangle$ from *dept*, or (3) delete both $\langle John, P\rangle$ and $\langle John, M\rangle$ from *emp* and *dept* respectively. Although, this solution is discarded because of the minimal change in databases, the translation is not unique.

Suppose now that the database instance is the following: $emp = \{\langle John, P\rangle\}$ and $dept = \emptyset$ and consider previous deletion. With respect to minimal change, and according to the first and second translations above, it is easily seen that the second translation changes the database instance less than the first one. We say then, that the delete view update is deterministic with respect to the database instance because, according to the current database state, there is one unique translation which guarantees minimal change.

The organization of this paper is as follows: We first recall some basic definitions concerning the relational database model in Section 2, then we present the notion of view update translation in Section 3. In Section 4, we introduce our formalism about the inversion of the relational algebra (excluding the difference). In Section 5, we present our method for characterizing the set of translations of a given view update by using the set of inverses computed by the algorithm given in Section 4. In Section 5.3, we consider two kinds of deterministic updates, and finally, we conclude with some perspectives in Section 6. The proof of the theorem is not included in this report, the interested reader is referred to [BL97].

2 Basic Definitions and Notation

We consider here a relational database model with a *fixed* database scheme, where several views can be created, possibly over the same relation scheme. Our approach is based on usual notions from the relational database model [AHV95, Ull89, Vos90], but we give definitions of notions that slightly differ from the classical approach.

Definition 2.1 *A universe is a finite set U whose elements are called attributes. Each attribute A of U is associated with a domain, denoted by dom_A, and with a set of variables denoted by var_A. The values of the domain A are denoted by v_A, v'_A, v^1_A, etc ..., and the variables of var_A are denoted x_A, x'_A, x^1_A, etc ... For each A in U, we use the following notation: $\Sigma_A = dom_A \cup var_A$, $dom = \bigcup_{A\in U} dom_A$, $var = \bigcup_{A\in U} var_A$, and $\Sigma = dom \cup var$.* $\square$

Definition 2.2 *A tuple t over X is an application from X to $\bigcup_{A\in X} \Sigma_A$ such that for every A in X, $t(A) \in \Sigma_A$. The tuple t is total if for each A of X, $t(A) \in dom_A$.* $\square$

Notation: $t(A)$ is denoted $t.A$, the scheme X of t is denoted by $sch(t)$, and ϵ denotes the "empty" tuple (over any relation scheme).

In the database definition below, we distinguish *relation symbol* from its effective value, called *relation instance* or *relation* for short.

Definition 2.3

1. *Given a relation symbol R over scheme $\mathcal{R}$, we call relation instance, or relation for short, of R any finite set of tuples r over $\mathcal{R}$. The scheme of r is denoted by $sch(r)$.*

2. *A database $\mathcal{D}$ defined on the universe U is a set of relation symbols $R_1, R_2, \ldots, R_n$, where each R_i is associated with a relation scheme $\mathcal{R}_i$. An instance of $\mathcal{D}$ is a set of relations $r_1, r_2, \ldots, r_n$ where each r_i is an instance of R_i.* $\square$

A relational expression E defined from the relation symbols $R_1, \ldots, R_n$ in this order is denoted by $E(R_1, \ldots, R_n)$. Moreover, the result of the evaluation of $E(R_1, \ldots, R_n)$ over $R_1, \ldots, R_n$ is an expression obtained from E by replacing each R_i with one of its instances r_i and denoted by $\mathcal{E}(r_1, \ldots, r_n)$. The relation scheme of the resulting relation is denoted by $sch(E)$.

For example, the database $\mathcal{D}$ of our Running Example is defined over the universe $U = \{Ename, Pname, Dname\}$ as containing the two relation symbols Emp and $Dept$, over the schemes $\mathcal{R}_1 = \{Ename, Pname\}$ and $\mathcal{R}_2 = \{Ename, Dname\}$, respectively. Moreover, we previously considered an instance of $\mathcal{D}$ in which $emp = \{\langle John, P\rangle\}$ is an instance of Emp and $dept = \{\langle John, CS\rangle\}$ is an instance of $Dept$. The view v is defined by $E(Emp, Dept) = \Pi_{Ename,Dname}(\sigma_{Pname=P}(Emp \bowtie Dept))$, and we have $\mathcal{E}(emp, dept) = \{\langle John, CS\rangle\}$.

3 View Update Translation

In this section, we formally define the notion of *translation* of a given update on a view. To this end, we recall from [Vos90] that a transaction T over a database $\mathcal{D}$ is a finite sequence of updates over $\mathcal{D}$. The semantics of T, also called its *effect*, is equal to the composition of the effects of the updates u_i that make up T.

In our approach, we formally define a transaction as follows:

Definition 3.1 *Transaction: Let $\mathcal{D}$ be a database and $R_1, \ldots, R_n$ be n relation symbols. A transaction is a finite sequence of updates $T = u_1 \ldots u_m$ where each u_i is a single update which consists in inserting (or deleting) a set of tuples in (from) R_i. The insertion of a set of tuples τ in R is denoted by $ins(\tau, R)$ and the deletion of a set of tuples τ from R is denoted by $del(\tau, R)$.* $\square$

We use the following terminology, where T is a transaction and τ a set of tuples over relation scheme $\mathcal{R}$:

1. Let r be an instance of R. If $u = ins(\tau, R)$ (resp. $u = del(\tau, R)$) is a single update over R, the effect of u over r denoted by $ins(\tau, r)$ (resp. $del(t, r)$) is $r := r \cup \tau$ (resp. $r := r - \tau$).

   Note that τ may be empty.

2. $T' = u'_1.u'_2. \ldots u'_k$ is called sub-transaction of $T = u_1 \ldots u_k$ if for every $i = 1, \ldots, k$, $u_i = upd(\tau_i, R_i)$, $u'_i = upd(\tau'_i, R'_i)$, $R'_i = R_i$, $\tau'_i \subseteq \tau_i$, and $upd = ins$ or $upd = del$.

3. Let $T = u_1 \ldots u_k$ be a transaction, and R be a relational symbol.

   T_R is called the R-transaction of T if: (a) T_R is a sub-transaction of T, (b) T_R contains all the single updates over R in T, and (c) for every $R_i \neq R$, $u_i = upd(\tau_i, R_i)$ implies $\tau_i = \emptyset$.

4. We denote by $eff(T_R, r)$, the effect of T_R on an instance r of a relation symbol R, i.e. the new state of the relation r after applying T_R on r.

   Moreover if R does not occur in a given transaction T, and if r is an instance of R, then T_R is an empty transaction and $eff(T_R, r)$ is r.

5. We call insert-transaction (resp. delete-transaction), a transaction which contains only insert (resp. delete) updates.

Definition 3.2 *Let $E(R_1, \ldots, R_n)$ be a relational expression and $\mathcal{U}(t, E)$ a view update of a tuple t over E. A* translation *of $\mathcal{U}(t, E)$ is a transaction $T = u_1.u_2 \ldots u_n$ such that:*

1. $u_i = upd(\tau_i, R_i)$ where, for $i = 1, \ldots, n$, $upd = ins$, or $upd = del$

2. (a) If $\mathcal{U}(t, E)$ is an insertion, then for all instances $r_1, \ldots, r_n$, of $R_1, \ldots, R_n$ resp. $t \in \mathcal{E}(eff(T_{R_1}, r_1), \ldots, eff(T_{R_n}, r_n))$,

(b) if $\mathcal{U}(t, E)$ is a deletion, then for all instances $r_1, \ldots, r_n$, of $R_1, \ldots, R_n$ resp. $t \notin \mathcal{E}(eff(T_{R_1}, r_1), \ldots, eff(T_{R_n}, r_n))$.

3. For each sub-transaction T' of T, point 1 above does not hold. $\square$

In Definition 3.2, the first point expresses the fact that for each database instance, the relational expression E applied on the relations $r_1 \cup \tau_1, \ldots, r_n \cup \tau_n$ (respectively $r_1 - \tau_1, \ldots, r_n - \tau_n$) generates (respectively does not generate) the tuple t. Moreover, the second point expresses a condition of minimal change independently from the database state. In other words, every view update translation is a "minimal" transaction that realizes the update.

Let $E(R_1, \ldots, R_n)$ be a relational expression not containing the difference operator. It can be seen that, if $\mathcal{U}(t, E)$ is an update of a tuple t over $sch(E)$, then:

1. If $\mathcal{U}(t, E)$ is an insertion, then every translation $T = u_1 \ldots u_n$ of $\mathcal{U}(t, E)$ is an insert-transaction such that, for every $i = 1, \ldots, n$, $u_i = ins(t_i, R_i)$, where t_i is a tuple over $\mathcal{R}_i$.

2. If $\mathcal{U}(t, E)$ is a deletion, then every translation $T = u_1 \ldots u_n$ of $\mathcal{U}(t, E)$ is a delete-transaction such that, for every $i = 1, \ldots, n$, $u_i = del(\tau_i, R_i)$, where τ_i is a set of tuples over $\mathcal{R}_i$.

As a consequence under the restriction that E does not contain the set difference operator, every translation of $\mathcal{U}(t, E)$ is written as $T = upd((\tau_1, R_1), \ldots, (\tau_n, R_n))$ where τ_i, $(i = 1, \ldots, n)$ is a set of tuples over the relation scheme $\mathcal{R}_i$ associated to R_i and where, $upd = ins$ or del, depending on whether $\mathcal{U}(t, E)$ is an insertion or a deletion.

We note that, as a consequence of this notation, item 2 in Definition 3.2 can be expressed as follows:

2': For each $T' \neq T$ such that $T' = upd((\tau'_1, R_1), \ldots, (\tau'_n, R_n))$ with $\tau'_i \subseteq \tau_i$ $(i = 1, \ldots, n)$ and $\tau'_{i_0} \neq \tau_{i_0}$ for one i_0, item 1 of Definition 3.2 does not hold.

Example 3.1 *Consider the database $\mathcal{D}$ of our Running Example and the insert view update $\mathcal{U}(\langle Mary, CS\rangle, E)$ over the view v. According to Definition 3.2, $T = ins((\{\langle Mary, P\rangle\}),$ $(\{\langle Mary, CS\rangle\}))$ is a translation of $\mathcal{U}(\langle Mary, CS\rangle, E)$ since $\mathcal{E}(\{\langle Mary, P\rangle\}, \{\langle Mary, CS\rangle\})$ $= \{\langle Mary, CS\rangle\}$ and since no sub-transaction of T is a translation of $\mathcal{U}(\langle Mary, CS\rangle, E)$.*

It is important to note that if the relational expression E contains the set difference operator, then translations of updates such as $\mathcal{U}(t, E)$ are neither insert- nor delete-transactions.

In the remaining of this paper, we consider *only* relational expressions not containing the set difference operator, usually referred to as *positive* relational expressions.

In the next section, we address the issue of translating view updates with respect to minimal change. We solve then the following problem: Given a database view update $\mathcal{U}$, find the set of all translations of $\mathcal{U}$. The solution is based on the concept of inverse-image.

4 Inverse-images: Formalism and Computation

The problem of inverting (positive) relational expressions in a relational database model has already been studied in [IL83] in the context of view equivalence. The relational expressions considered in [IL83] are defined over one base relation, and the inverses are represented by tables, which are structures usually used to represent incomplete informations in databases. In our approach, inverses are represented by means of linear equations (c-k-tuples) whose solutions are called inverse-images and are used to translate view updates.

4.1 Definitions, Notation

Definition 4.1 *k-tuple : A k-tuple is a tuple of tuples denoted by $(t_1, t_2, \ldots, t_n)$, where $n \geq 1$ and t_i $(i = 1, \ldots, n)$ is a tuple.* □

Note that a single tuple t can be seen as the k-tuple (t). A k-tuple of the form (t) or of the form $(\epsilon, \ldots, \epsilon, t, \epsilon, \ldots, \epsilon)$, where $t \neq \epsilon$, is called *unary.* Roughly speaking, a k-tuple is unary if it "contains" only one non-empty tuple. Moreover, any k-tuple of the form $(\epsilon, \ldots, \epsilon)$ is simply denoted by ϵ whenever no ambiguity is possible.

We define now the notion of constraint as follows:

Definition 4.2 *An elementary constraint is a boolean expression c of the form: $\top$ (true), $\bot$ (false) or $x\theta y$ where x and y are variables or constants of Σ and $\theta \in \{<, >, \geq, \leq, =\}$.*

A constraint is a boolean expression $C = c_1 \wedge c_2 \wedge \ldots \wedge c_l$ $(l \geq 1)$ such that for each $i = 1, \ldots, l$, c_i is an elementary constraint. □

Associating a constraint with a k-tuple N produces a *constrained k-tuple* as follows:

Definition 4.3 *A constrained k-tuple, or c-k-tuple, is a pair $P = [N, C]$ where N is a k-tuple and C a constraint over N. The c-k-tuple P is called unary if N is a unary k-tuple.* □

We note that every unary k-tuple corresponds to the c-k-tuple $[(t), \top]$.

Selection conditions are associated with constraints as follows:

Given a relational expression $E = \sigma_\Gamma(E')$, where Γ is a selection condition over the attributes of $sch(E)$, we define as in [AHV95, Vos90], an elementary selection condition as an expression of the form $X\theta Y$ or $\neg(X\theta Y)$ where X is an attribute and Y is either an attribute or a constant, and $\theta \in \{<, >, \geq, \leq, =\}$. Without loss of generality, we consider that any selection condition Γ is of the form $\Gamma = \Gamma_1 \vee \Gamma_2 \vee \ldots \vee \Gamma_k$, where $\Gamma_i = \gamma_i^1 \wedge \gamma_i^2 \wedge \ldots \wedge \gamma_i^l$ and γ_i^j, $j = 1, \ldots, l$ is an elementary selection condition.

Given a selection condition Γ and a tuple t over $\mathcal{R}$ such that all attributes occurring in Γ are in $\mathcal{R}$, we consider the set $\bar{\Gamma} = \{\bar{\Gamma}_1, \ldots, \bar{\Gamma}_k\}$ where, for every $i = 1, \ldots, k$, $\bar{\Gamma}_i$ is obtained from Γ_i as follows: $\bar{\Gamma}_i = c_i^1 \wedge c_i^2 \wedge \ldots \wedge c_i^l$ $(l \geq 1)$ where for each $j = 1, \ldots, l$, c_i^j is obtained from γ_i^j in the following way: If X_i^j (respectively Y_i^j) is an attribute, then X_i^j (respectively Y_i^j) is replaced by $t.X_i^j$ (respectively $t.Y_i^j$), else X_i^j (respectively Y_i^j) does not change.

Clearly, we have the following: $t \models \Gamma$ iff there exists i such that $\bar{\Gamma}_i$ is true.

Example 4.1 *Let* $\Gamma = \sigma_{(B=b)\vee(B=b')}$ *a select condition over a tuple* $t = abc$ *defined over the scheme* ABC*. We have then* $\bar{\Gamma} = \{(b = b), (b = b')\}$*.* $t \models \Gamma$ *because there exists* $i = 1$ *such that* $\bar{\Gamma}_i = \{b = b\}$ *is true.*

In the following section, we present our method of inverting the relational algebra, under the restriction that the expression does *not* contain the operator of difference.

4.2 Inverting the Relational Algebra

We define now the set of inverse-images of a tuple t with respect to a given relational expression E.

Definition 4.4 *Let* $E(R_1, R_2, \ldots, R_n)$ *be a relational expression and* t *be a tuple over* $sch(E)$*. The set of the inverse-images of* t *by* E*, denoted by* $Inv(t, E)$*, is the set of the k-tuples* $(t_1, t_2, \ldots, t_n)$ *such that:*

1. $\mathcal{E}(\{t_1\}, \{t_2\}, \ldots, \{t_n\}) = \{t\}$ *such that, for* $i = 1, \ldots, n$, $sch(t_i) = sch(r_i)$
2. *for every* $i = 1, \ldots, n$*, if* $t_i \neq \epsilon$ *then* $\mathcal{E}(\{t_1\}, \ldots, \{t_{i-1}\}, \{\}, \{t_{i+1}\}, \ldots, \{t_n\}) \neq \{t\}$ □

Example 4.2 *For* $E = \Pi_{AC}(\sigma_{(B=b)}(R \bowtie R'))$ *where* R *and* R' *are defined over the schemes* AB *and* BC *respectively, and for* $t = ac$, $Inv(t, E) = \{(ab, bc)\}$*. In this case,* $Inv(t, E)$ *is a finite set. However, if* $E = \Pi_{AC}(R \bowtie R'))$*, for* $t = ac$, $Inv(t, E) = \{(ax_B, x_Bc) \mid x_B \in dom_B\}$*. In this case,* $Inv(t, E)$ *is an infinite set.*

Because of the monotonicity of relational algebra excluding the difference operator, Definition 4.4 is independent of the database state, that is, if $(t_1, t_2, \ldots, t_n) \in Inv(t, E)$, then for all instances r_1, r_2, ..., r_n, $\mathcal{E}(r_1 \cup \{t_1\}, r_2 \cup \{t_2\}, \ldots, r_n \cup \{t_n\}) = \{t\}$.

We note that, if relational expressions are allowed to contain the difference operator, then Definition 4.4 depends on the database state. Indeed, for $E = R - S$, if r and s are two instances of R and S respectively, and if t is a tuple defined over $sch(E)$, then $(t, \epsilon) \in Inv(t, E)$ according to Definition 4.4. But, if $r = \emptyset$ and $s = \{t\}$, then $t \notin \mathcal{E}(r \cup \{t\}, s \cup \{\epsilon\})$.

In order to compute the set of inverse-images by a given relational expression of a given tuple we define the following four operators:

Definition 4.5 *Let* $P = [(t), C]$ *be a unary c-k-tuple such that* t *is defined over* $\mathcal{R}$*, and* $\mathcal{R}'$ *be a relation scheme such that* $\mathcal{R}' \subseteq \mathcal{R}$*. The projection operator of* P *over* $\mathcal{R}'$*, denoted by* $proj(P, \mathcal{R}')$*, is:* $proj(P, \mathcal{R}') = [(t.\mathcal{R}'), C]$*.* □

Definition 4.6 *Let* $P = [(t), C]$ *be a unary c-k-tuple such that* t *is defined over* $\mathcal{R}$*, and let* $\mathcal{R}'$ *be a relation scheme such that* $\mathcal{R} \subseteq \mathcal{R}'$*. The extension operator of* P *over* $\mathcal{R}'$*, denoted by* $ext(P, \mathcal{R}')$*, is:* $ext(P, \mathcal{R}') = [(N\bar{N}), C]$ *such that* $N = (t)$ *and* $N' = (t')$ *where* t' *is a tuple of* var *defined over* $\mathcal{R}' - \mathcal{R}$*. At each application of the extension operator, ext generates new variables.* □

Definition 4.7 *Given two c-k-tuples* $P = [N, C]$ *and* $P' = [N', C']$ *where* $N = (t_1, t_2, \ldots, t_m)$ *and* $N' = (t'_1, t'_2, \ldots, t'_n)$*, the append operator of* P *and* P'*, denoted by* $P \odot P'$*, is:*
$P \odot P' = [(N, N'), C \wedge C']$ *where* $(N, N') = (t_1, t_2, \ldots, t_m, t'_1, t'_2, \ldots, t'_n)$. $\Box$

Definition 4.8 *Let* $P = [(t_1, t_2, \ldots, t_m), C]$ *and* $P' = [(t'_1, t'_2, \ldots, t'_n), C']$ *be two c-k-tuples where their respective k-tuples with arities* m *and* n*. The completion operator of* P *and* P'*, denoted by* $P \diamond P'$*, is:*
$P \diamond P' = \{[(t_1, t_2, \ldots, t_m, \underbrace{\epsilon, \ldots, \epsilon}_{n\ times}), C], [(\underbrace{\epsilon, \ldots, \epsilon}_{m\ times}, t'_1, t'_2, \ldots, t'_n), C']\}$.
$P \diamond P'$ *is simply denoted by:* $\{[(N, \epsilon), C], [(\epsilon, N'), C']\}$. $\Box$

The operators of $Proj$, Ext, $\bigodot$ and $\Diamond$ over sets of c-k-tuples, respectively extend the operators $proj$, ext, $\odot$ and $\diamond$ as follows: Let $\mathcal{P}$ and $\mathcal{P}'$ two sets of c-k-tuples and $\mathcal{R}$ a relation scheme, then we have:

- $Proj(\mathcal{P}, \mathcal{R}) = \{proj(P, \mathcal{R}) \mid P \in \mathcal{P}\}$, if $\mathcal{P}$ is a set of unary c-k-tuples.
- $Ext(\mathcal{P}, \mathcal{R}) = \{ext(P, \mathcal{R}) \mid P \in \mathcal{P}\}$, if $\mathcal{P}$ is a set of unary c-k-tuples.
- $\mathcal{P} \bigodot \mathcal{P}' = \{P \odot P' \mid P \in \mathcal{P}, P' \in \mathcal{P}'\}$.
- $\mathcal{P} \Diamond \mathcal{P}' = \{P \diamond P' \mid P \in \mathcal{P}, P' \in \mathcal{P}'\}$.

The inverse of a set of c-k-tuples with respect to a given relational expression can be computed based on the following definition.

Definition 4.9 *Let* E *be a relational expression and let* $\mathcal{P}$ *be a set of unary c-k-tuples. We call inverse of* $\mathcal{P}$ *with respect to* E*, the set of c-k-tuples denoted by* $\widetilde{E}(\mathcal{P})$ *and defined by:*

- *If* $E = R$*, where* R *is a relation symbol, then* $\widetilde{E}(\mathcal{P}) = \mathcal{P}$*.*
- *If* $E = \sigma_\Gamma(E')$*, where* E' *is a relational expression and where* $\Gamma = \Gamma_1 \vee, \ldots \vee \Gamma_k$*, then* $\widetilde{E}(\mathcal{P}) = \widetilde{E'}(\{[N, C \wedge \bar{\Gamma}_i]) \mid [N, C] \in \mathcal{P}, \bar{\Gamma}_i \in \bar{\Gamma}\})$*, where* $\bar{\Gamma} = \{\bar{\Gamma}_1, \ldots, \bar{\Gamma}_k\}$ *is the set of constraints associated with* Γ*.*
- *if* $E = \Pi_X(E')$*, where* E' *is a relational expression, then* $\widetilde{E}(\mathcal{P}) = \widetilde{E'}(Ext(\mathcal{P}, sch(E')))$*.*
- *If* $E = E' \bowtie E''$*, where* E' *and* E'' *are two relational expressions, then* $\widetilde{E}(\mathcal{P}) = \widetilde{E'}(Proj(\mathcal{P}, sch(E'))) \bigodot \widetilde{E''}(Proj(\mathcal{P}, sch(E'')))$*.*
- *If* $E = E' \cap E''$*, where* E' *and* E'' *are two relational expressions, then* $\widetilde{E}(\mathcal{P}) = \widetilde{E'}(\mathcal{P}) \bigodot \widetilde{E''}(\mathcal{P})$*.*
- *If* $E = E' \cup E''$*, where* E' *and* E'' *are two relational expressions, then* $\widetilde{E}(\mathcal{P}) = \widetilde{E'}(\mathcal{P}) \Diamond \widetilde{E''}(\mathcal{P})$*.* $\Box$

Example 4.3 *Let* $E = \Pi_{AC}(\sigma_{B=b}(R \bowtie R'))$ *be a relational expression such that* R *and* R' *are two relation symbols defined over* AB *and* BC*, respectively. For* $\mathcal{P} = \{[(ac), \top]\}$*, we obtain* $\widetilde{E}([ac, \top]) = \{[(ax_B, x_Bc), x_B = b]\}$*. Indeed:*

- *For* $E = \Pi_{AC}(E')$ *where* $sch(E') = ABC$*,* $\widetilde{E}([(ac), \top]) = \widetilde{E'}([(ax_Bc), \top])$
- *For* $E' = \sigma_{B=b}(E'')$*,* $\widetilde{E'}([(ax_Bc), \top]) = \widetilde{E''}[(ax_Bc), x_B = b]$
- *For* $E'' = (R \bowtie R')$ *where* $sch(R) = AB$ *and* $sch(R') = BC$*,* $\widetilde{E''}([(ax_Bc), x_B = b]) = [(ax_B, x_Bc), x_B = b]$.

4.3 Inversion and Inverse-image

The following theorem states that, for every tuple t_0 and for every relational expression E, the set $Inv(t_0, E)$ is obtained by instantiating every c-k-tuple of $\widetilde{E}[(t_0), \top]$ in such a way that the associated constraint takes the value *true*.

Theorem 4.1 *Let* E *be a relational expression, and let* t *be a tuple defined over* $sch(E)$*. Then:* $Inv(t, E) = \{(N\psi) \mid [N, C] \in \widetilde{E}([(t), \top])$ *and* $C\psi = \top$ *where* ψ *is an instantiation of all variables in* $[N, C]\}$*.* □

Proof. See Theorem 2.1 in [BL97]. ■

Let us illustrate Theorem 4.1 using our Running Example.

Example 4.4 *We recall that in our Running Example the view* v *is defined by the relational expression* $E = \Pi_{Ename,Dname}\ (\sigma_{Pname=P}(Emp \bowtie Dept))$*. If* $t = \langle Mary, CS\rangle$ *is a tuple defined over* $sch(E) = Ename\ Dname$*, then:*
$\widetilde{E}(\langle Mary, CS\rangle, \top) = \{[(\langle Mary, x_{Pname}\rangle, \langle x_{Pname}, CS\rangle), x_{Pname} = P]\}$*. By instantiating the variable* x_{Pname} *by* P*, we obtain* $Inv(t, E) = \{(\langle Mary, P\rangle, \langle P, CS\rangle)\}$*.*

It is important to note that the computation of the set of inverse-images with respect to a given relational expression is computed only once at the definition step, and not at each evaluation of its instances.

5 View Update Translations using Inverse

In this section, we study three important aspects of the notion of translation, for a given relational expression E, a tuple t defined over $sch(E)$ and a view update $\mathcal{U}(t, E)$. In this respect, we have to answer to the following questions:

1. What is the relationship between $Inv(t, E)$ and translations of $\mathcal{U}(t, E)$?
2. How to characterize deterministic view updates $\mathcal{U}(t, E)$?
3. How to use translations in the process of database updating?

All these questions are answered in the following section.

5.1 Relationship between Translations and Inverses

The next two propositions answer the first question above. Proposition 5.1 below states that there is a strong link between insert view update translations and the set of the inverse-images of an updated tuple w.r.t. the relational expression which defines the view.

Proposition 5.1 *Let $E(R_1, \ldots, R_n)$ be a relational expression and $\mathcal{U}(t, E)$ a view update of a tuple t over E such that t is a tuple defined over $sch(E)$. If $\mathcal{U}(t, E)$ is an insertion, then: $T = ins((t_1, R_1), \ldots, (t_n, R_n))$ is a translation of $\mathcal{U}(t, E)$ if and only if $(t_1, \ldots, t_n) \in Inv(t, E)$.* □

Proof. Let $\mathcal{U}(t, E)$ be a view update consisting in inserting a tuple t defined on $sch(E)$ over a relational expression $E(R_1, \ldots, R_n)$, and let $r_1, \ldots, r_n$ be n instances of $R_1, \ldots, R_n$ respectively.

1. Let $T = ins((t_1, R_1), \ldots, (t_n, R_n))$ be a translation of $\mathcal{U}(t, E)$ and let's show that $(t_1, \ldots, t_n) \in Inv(t, E)$. In other words:
   (a) $\mathcal{E}(\{t_1\}, \ldots, \{t_n\}) = \{t\}$.
   Assume that for every instance r_i, $i = 1, \ldots, n$, $r_i = \emptyset$. After applying T, we obtain $r_i = \{t_i\}$, for every r_i. According to Definition 3.2, $t \in \mathcal{E}(\{t_1\}, \ldots, \{t_n\})$. Hence, we have $\mathcal{E}(\{t_1\}, \ldots, \{t_n\}) = \{t\}$.
   (b) for every j such that $t_j \neq \epsilon$, $t \notin \mathcal{E}(\{t_1\}, \ldots, \{t_{j-1}\}, \{\}, \{t_{j+1}\}, \ldots, \{t_n\})$.
   Let $\{t_1\}, \ldots, \{t_n\}$ be n instances of $R_1, \ldots, R_n$ respectively, and assume that $t \in \mathcal{E}(\{t_1\}, \ldots, \{t_n\})$. According to the monotonicity of the insertion, we have: $t \in \mathcal{E}(r_1 \cup \{t_1\}, \ldots, r_i, \ldots, r_n \cup \{t_n\})$ for all instances $r_1, \ldots, r_n$. This implies that T is not a translation. Hence, for every j such that $t_j \neq \epsilon$, $t \notin \mathcal{E}(\{t_1\}, \ldots, \{t_{j-1}\}, \{\}, \{t_{j+1}\}, \ldots, \{t_n\})$.

   items (a) and (b) above imply that $(t_1, \ldots, t_n) \in Inv(t, E)$. (see Definition 4.4).
2. Assume the k-tuple $(t_1, \ldots, t_n) \in Inv(t, E)$, we show that $ins((t_1, R_1), \ldots, (t_n, R_n))$ is a translation of $\mathcal{U}(t, E)$.
   (a) $(t_1, \ldots, t_n) \in Inv(t, E)$ implies that $\mathcal{E}(\{t_1\}, \ldots, \{t_n\}) = \{t\}$. Then $t \in \mathcal{E}(r_1 \cup \{t_1\}, \ldots, r_n \cup \{t_n\})$ for all instances $r_1, \ldots, r_n$.
   (b) $(t_1, \ldots, t_n) \in Inv(t, E)$ implies that, for every $t_j \neq \epsilon$, $\mathcal{E}(\{t_1\}, \ldots, \{t_{j-1}\}, \{\}, \{t_{j+1}\}, \ldots, \{t_n\}) \neq \{t\}$.
   Then, for the instances $r_i = \emptyset$, $i = 1, \ldots, n$, we have $t \notin \mathcal{E}(r_1 \cup \{t_1\}, \ldots, r_{j-1} \cup \{t_{j-1}\}, \{\}, r_{j+1} \cup \{t_{j+1}\}, \ldots, r_n \cup \{t_n\})$.

   items (a) and (b) above imply that $ins(t_1, R_1, \ldots, (t_n, R_n))$ is a translation of $\mathcal{M}(t, E)$. (see Definition 3.2).

■

The following example illustrates Proposition 5.1.

Example 5.1 *Let $\mathcal{D}$ be the database of our Running Example, and let v be the view defined by the relational expression $E = \Pi_{Ename,Dname}(\sigma_{Pname=P}(Emp \bowtie Dept))$.*

Consider the following view update: $insert(\langle Mary, CS\rangle, E)$. According to the definition of a translation, $ins((\langle Mary,\ P\rangle, Emp), (\langle Mary, CS\rangle, Dept))$ is a translation of $Ins(\langle Mary, CS\rangle, E)$. On the other hand, as we have seen in Example 4.4, $(\langle Mary, P\rangle, \langle Mary, CS\rangle) \in Inv(\langle Mary,\ CS\rangle, E)$.

In order to translate the delete updates of tuples over relational expressions, we define for a given set of tuples τ_i, the set Inv_i of k-tuples which belong to Inv and which contain at least one non-empty tuple from τ_i. Formally, the set Inv_i is defined as follows:

Definition 5.1 *Given a relational expression $E(R_1, \ldots, R_n)$ and the sets of tuples $\tau_1, \ldots, \tau_n$ defined respectively on the relation schemes $\mathcal{R}_1, \ldots, \mathcal{R}_n$, we define for each $i = 1, \ldots, n$, the set $Inv_i = \{N \in Inv(t, E) \mid \tau_i \neq \emptyset, N_i \in \tau_i\}$, where N_i is the $i-th$ tuple of N.* $\square$

The next proposition states the relationship between delete view update translations and partitions of the set of inverse-images.

Proposition 5.2 *Let $E(R_1, \ldots, R_n)$ be a relational expression and $\mathcal{U}(t, E)$ a view update of a tuple t defined on $sch(E)$ over E. If $\mathcal{U}(t, E)$ is a deletion , then: $T = del((\tau_1, r_1\rangle, \ldots, (\tau_n, r_n\rangle)$ is a translation of $\mathcal{U}(t, E)$ iff $\{Inv_i \mid i = 1, \ldots, n\}$ is a partition of $Inv(t, E)$.* $\square$

Proof. Let $E(R_1, \ldots, R_n)$ be a relational expression and $\mathcal{U}(t, E)$ an update which consists in deleting the tuple t over E.

1. Let $T = del((\tau_1, R_1), \ldots, (\tau_n, R_n))$ be a translation of $\mathcal{U}(t, E)$. We have to show that $\{Inv_i \mid i = 1, \ldots, n\}$ is a partition of $Inv(t, E)$. In other words:
   (a) for every i, $Inv_i \neq \emptyset$. This is true according to Definition 5.1.
   (b) for every i, j such that $i \neq j$, $Inv_i \cap Inv_j = \emptyset$.
   Assume that there exists i_0 and j_0 such that $N_0 = (q_1, \ldots, q_n) \in Inv_{i_0} \cap Inv_{j_0}$, we show that $T = del((\tau_1, R_1), \ldots, (\tau_n, R_n))$ is not a translation of $\mathcal{U}(t, E)$.
   According to Definition 5.1, we have: $N_0 \in Inv(t, E)$, $q_{i_0} \in \tau_{i_0}$, and $q_{j_0} \in \tau_{j_0}$. Let T' be a translation of $Del(t, E)$ such that $T' = T$ except on i_0. We have then $T' = del((\tau_1, R_1), \ldots, (\tau'_{i_0}, R_{i_0}), \ldots (\tau_n, R_n))$ such that $\tau'_{i_0} = \tau_{i_0} - q_{i_0}$. Moreover, let $I = (\{q_1\}, \ldots, \{q_{i_0}\}, \ldots, \{q_{j_0}\}, \ldots, \{q_n\})$ be n instances of $R_1, \ldots, R_n$ respectively. Since $(q_1, \ldots, q_{i_0}, \ldots, q_{j_0}, \ldots, q_n) = N_0$ and $N_0 \in Inv(t, E)$, we have $\{t\} \in \mathcal{E}(I)$.
   Let I' be the instance after applying T'. T' does not delete q_{i_0} from I but deletes q_{j_0} from I. Hence $I' = (\{q'_1\}, \ldots, \{q'_n\})$ where $q'_i = q_i$ or $q'_i = \epsilon$ if $i \notin \{i_0, j_0\}$, $q'_{i_0} = q_{i_0}$, and $q'_{j_0} = \epsilon$. According to Definition 4.4, $\{t\} \notin \mathcal{E}(I')$. Hence T is not a translation of $\mathcal{U}(t, E)$.
   (c) $\bigcup_i Inv_i = Inv(t, E)$.

i. for every i, if $N \in Inv_i$, then $N \in Inv(t,E)$ according to Definition 5.1. We have then $\bigcup_i Inv_i \subseteq Inv(t,E)$.

ii. Let's show now that $Inv(t,E) \subseteq \bigcup_i Inv_i$.
Assume $N = (q_1, \ldots, q_n) \in Inv(t,E)$, and assume that for every i, $N \notin Inv_i$. We distinguish two cases:
(1) for every i, $\tau_i = \emptyset$. This implies that T is not a translation.
(2) for every $\tau_i \neq \emptyset$, for every $q_i \in N$, $q_i \notin \tau_i$. Then, for the database instance $I = (\{q_1\}, \ldots, \{q_n\})$, $\mathcal{E}(\{q_1\} - \tau_1, \ldots, \{q_n\} - \tau_n)$ $= \mathcal{E}(\{q_1\}, \ldots, \{q_n\}) = \{t\}$. Hence T is not a translation.

2. Let $T = del((\tau_1, R_1), \ldots, (\tau_n, R_n))$ be a transaction over the relation symbols $R_1, \ldots, R_n$, and assume $Inv_i = \{N \in Inv(t,E) \mid N_i \in \tau_i\}$. Suppose that $\{Inv_i \mid i = 1, \ldots, n\}$ is a partition of $Inv(t,E)$, let's show that T is a translation of $\mathcal{U}(t,E)$. In other words:

   (a) For every database instance $r_1, \ldots, r_n$, $t \notin \mathcal{E}(r_1 - \tau_1, \ldots, r_n - \tau_n)$.
   Assume that there exists one database instance $(r'_1, \ldots, r'_n)$ such that $t \in \mathcal{E}(r'_1 - \tau_1, \ldots, r'_n - \tau_n)$. Then, there exists $N = (q_1, \ldots, q_n) \in Inv(t,E)$ such that $q_i \in (r'_i - \tau_i)$ for every i. Hence for every i, $q_i \notin \tau_i$. Hence for every i $N \notin Inv_i$. Since $N \in Inv(t,E)$, $\bigcup Inv_i \neq Inv(t,E)$. This implies that $\{Inv_i \mid i = 1, \ldots, n\}$ is not a partition of $Inv(t,E)$.

   (b) There exists i_0 and an instance $(r'_1, \ldots, r'_n)$ such that:
   $t \in \mathcal{E}(r'_1, \ldots, r'_n)$ implies that $t \in \mathcal{E}(r'_1 - \tau'_1, \ldots, r'_i - \tau'_{i_0}, \ldots, r'_n - \tau'_n)$ with $\tau'_{i_0} \subseteq \tau_{i_0}$ and $\tau'_i = \tau_i$ for every $i \neq i_0$. Assume $q_{i_0} \in \tau_{i_0}$ and $q_{i_0} \notin \tau'_{i_0}$. According to Definition 5.1, we have $q_{i_0} \in \tau_{i_0}$. We have $N \notin Inv_j$ since for every $j \neq i_0$, $q_j \notin \tau_j$ ($\{Inv_i \mid i = 1, \ldots, n\}$ is a partition of $Inv(t,E)$). Hence, if $r'_i = \{q_i\}$ for every $i \neq i_0$, we have $q_i \in r'_i - \tau_i$ and $q_{i_0} \in \tau'_{i_0}$. Hence $t \in \mathcal{E}(\{q_1\} - \tau_1, \ldots, \{q_{i_0}\} - \tau'_{i_0}, \ldots, \{q_n\} - \tau_n)$.

■

The following example illustrates Proposition 5.2.

Example 5.2 *Let $\mathcal{D}$ the database as defined in the Running Example, and v the view defined over $\mathcal{D}$ by the relational expression $E = \Pi_{Ename,Dname}(\sigma_{Pname=P}(Emp \bowtie Dept))$.
Consider the following view update: $Del(\langle John, M\rangle, E)$. Clearly, the two translations of $delete(\ (\langle John, M\rangle, E)$ are $del(\langle John, P\rangle, Emp)$ and $del(\langle John, M\rangle, Dept)$.
Moreover, $Inv(\langle John,\ M\rangle, E) = \{(\langle John, P\rangle, \langle P, M\rangle)\}$. Then we have:*

1. *For $del((\langle John, P\rangle, Emp), (\emptyset, Dept))$, $Inv_1 = \{(\langle John, P\rangle, \langle P, M\rangle)\}$.*
   Clearly, $\{Inv_1\}$ is a partition of Inv.

2. *For $del(\langle \emptyset, Emp\rangle, \langle\langle John, M\rangle, Dept\rangle)$, $Inv_2 = \{(\langle John, P\rangle, \langle P, M\rangle)\}$.*
   Clearly, $\{Inv_2\}$ is a partition of Inv.

Remark 5.1 *Given a relational expression $E(R_1, \ldots, R_n)$ and a tuple t defined on $sch(E)$, a partition of $Inv(t,E)$ does not always correspond to a translation of $Del(t,E)$.* $\square$

We illustrate the above remark by the following example.

Example 5.3 *Let $E = \Pi_{AC}(\sigma_{B\geq 5}(R \bowtie R'))$ be a relational expression such that r and r' are two instances of R and R', respectively defined over the schemes AB and BC and let $dom_B = \{1,2,3,4,5,6,7\}$. Let ac be a tuple defined over $sch(E)$. Then we have: $Inv(ac, E)$ $= \{(a5,5c),(a6,6c),(a7,7c)\}$. It can be seen that $\{\{(a5,5c)\},\{(a6,6c)\}, \{(a7,7c)\}\}$ is a partition of $Inv(ac,E)$, but this partition does not correspond to any translation of $Del(ac,E)$.*

5.2 Computing Translations from Inverses

Given a relational expression $E(R_1, \ldots, R_n)$, a tuple t defined over $sch(E)$ and a view update $\mathcal{U}(t,E)$, a translation of $\mathcal{U}(t,E)$ is built from the set $Inv(t,E)$ in the following way:

(1) If $\mathcal{U}(t,E)$ is an insert view update, based on Proposition 5.1, the computation of $Inv(t,E)$ allows for obtaining all translations of $\mathcal{U}(t,E)$. Indeed, Proposition 5.1 states that, to every inverse-image in $Inv(t,E)$, corresponds one translation of $\mathcal{U}(t,E)$.

(2) If $\mathcal{U}(t,E)$ is a delete view update, based on Proposition 5.2, every partition of $Inv(t,E)$ of the form $\{Inv_1, \ldots, Inv_n\}$ corresponds to a translation of the form $del((\tau_1, R_1), \ldots, (\tau_n, R_n))$. Moreover, the sets τ_i can be built up fro the set$Inv(t,E)$ based on the following proposition:

Proposition 5.3 *Let $E(R_1, \ldots, R_n)$ be a relational expression, t a tuple defined on $sch(E)$ and $\mathcal{U}(t,E)$ a view update consisting in deleting the tuple t over E.*
$del((\tau_1, R_1), \ldots, \tau_n, R_n))$ is a translation of $\mathcal{U}(t,E)$ if $\tau_1, \ldots, \tau_n$ are defined as follows:
Let $Inv(t,E)$ be the current inverse-images set at step i, and assume that the sets τ_1, ..., τ_{i-1} are already defined during steps j, $j = 1, i-1$. We have then:

1. $Inv_{i-1} := \{N \in Inv(t,E) \mid N_{i-1} \in \tau_{i-1}\}$
2. $Inv(t,E) := Inv(t,E) - Inv_{i-1}$
3. $\tau_i := \{N_i \mid N \in Inv(t,E),$ *and* $N_i \neq \epsilon\}$

Proof. Let $E(R_1, \ldots, R_n)$ be a relational expression and $\mathcal{U}(t,E)$ be a view update which consists in deleting a tuple t over E. We have to show that $del((\tau_1, R_1), \ldots, (\tau_n, R_n))$ is a translation of $\mathcal{U}(t,E)$ if $\{Inv_1, \ldots, Inv_n\}$ is such that, for every i, Inv_i is defined as described in item 1 above.
From items 1 and 2 above, it is obvious that, for every $i = 1, \ldots, n$, $Inv_i \neq \emptyset$, and for every $i \neq j$, $Inv_i \cap Inv_j = \emptyset$. Moreover, it is obvious that $\bigcup_{i=1}^{n} Inv_i \subseteq Inv(t,E)$. The remaining part to prove is $Inv(t,E) \subseteq \bigcup_{i=1}^{n} Inv_i$. Since $Inv(t,E)$ does not contain k-tuples of the form $(\epsilon, \ldots, \epsilon)$ according to Definition 5.1, assume that there exists $N = (\ldots, t_i, \ldots) \in Inv(t,E)$ where $t_i \neq \epsilon$ and $N \notin \bigcup_{i=1}^{n} Inv_i$. But, in step i, $t_i \in Inv_i$. This contradicts the hypothesis.

Moreover, the recurrence process of Proposition 5.3 terminates. Indeed, the sequence of the sets $Inv^1(t, E)$, $Inv^2(t, E)$, ..., $Inv^n(t, E)$ corresponding to steps $1, \ldots, n$ respectively, is a descending sequence, and has as lower bound the empty set. ∎

Remark 5.2 *Sets τ_i of Proposition 5.3 can be defined in any order. Indeed, This is possible since it is sufficient to define a permutation of the sets $\tau_1, \ldots, \tau_n$ to the set of steps $\{1, \ldots, n\}$.*

The following example illustrates Proposition 5.3 above.

Example 5.4 *Let $E = \Pi_{AC}\sigma_{B\geq b}(R_1 \bowtie R_2) \cup R_3$ be a relational expression such that r_1, r_2 and r_3 are three instances of R_1, R_2 and R_3 defined on the relation schemes AB, BC and AC respectively and $dom_B = \{b, b'\}$ with $b' \geq b$. Let $t = ac$ be a tuple defined on $sch(E) = AC$.Then, we have $Inv(ac, E) = \{(ab, bc, \epsilon), (ab', b'c, \epsilon), (\epsilon, \epsilon, ac)\}$.*

In order to built translations of $Del(ac, E)$ from $Inv(ac, E)$, and since $Del(ac, E)$ has more than one translation, we will choose the following two translations of the form $del((\tau_1, r_1)$, (τ_2, r_2), $(\tau_3, r_3))$ in the following way:

1. *First translation:*

   Choose the second tuple $b'c$ from $(ab', b'c, \epsilon)$ and put it in τ_2. Hence, $b'c \in \tau_2$.

   (a) $\tau_2 = \{b'c\}$, *and* $Inv_2 = \{(ab', b'c, \epsilon)\}$.
   $Inv(t, E) = \{(ab, bc, \epsilon), (\epsilon, \epsilon, ac)\}$.

   (b) $\tau_3 = \{ac\}$, *and* $Inv_3 = \{(\epsilon, \epsilon, ac)\}$.
   $Inv(t, E) = \{(ab, bc, \epsilon)\}$.

   (c) $\tau_1 = \{ab\}$, *and* $Inv_3 = \{(ab, bc, \epsilon)\}$.
   $Inv(t, E) = \emptyset$.

   Hence, $T = del((\{ab\}, r_1), (\{b'c\}, r_2), (\{ac\}, r_3))$ is a translation of $Del(ac, E)$.

2. *Second translation:*

   Choose the first tuple of (ab, bc, ϵ) and put it in τ_1. Hence, $ab \in \tau_1$.

   (a) $\tau_1 = \{ab\}$, *and* $Inv_1 = \{(ab, bc, \epsilon)\}$.
   $Inv(t, E) = \{(ab', b'c, \epsilon), (\epsilon, \epsilon, ac)\}$.

   (b) $\tau_1 = \{ab, ab'\}$, *and* $Inv_1 = \{(ab, bc, \epsilon), (ab', b'c, \epsilon)\}$.
   $Inv(t, E) = \{(\epsilon, \epsilon, ac)\}$.

   (c) $\tau_3 = \{ac\}$, *and* $Inv_3 = \{(\epsilon, \epsilon, ac)\}$. $Inv(t, E) = \emptyset$.

   Hence, $T = del((\{ab, ab'\}, r_1), (\emptyset, r_2), (\{ac\}, r_3))$ is a translation of $Del(ac, E)$.

5.3 View Update Determinism

Non determinism is one of the most important problems in database updating through views. Indeed, we have seen in the previous section that, for every view update $\mathcal{U}$, there exists in general more than one database translation which translates $\mathcal{U}$. In this section, we characterize two kinds of deterministic updates based on translations: (1) deterministic updates regardless to the database state, called strongly deterministic updates, and (2) deterministic updates according to some database state, called weakly deterministic updates.

5.3.1 Strongly Deterministic View Updates

Intuitively, a view update $\mathcal{U}(t, E)$ is *strongly* deterministic if it has a single translation. However, in our formalism, this is not always true in few particular cases, as shown below.

(1) If $E = R \cup R$, and if t is a tuple over $sch(E)$, then $Ins(t, E)$ has two translations, i.e. $T_1 = ins((\epsilon, t), (t, \epsilon))$ and $T_2 = ins((t, \epsilon), (\epsilon, t))$.

(2) Similarly, If $E = R \cap R$, and if t is a tuple over $sch(E)$, then $Del(t, E)$ has two translations, i.e. $T_1 = del((\epsilon, t), (t, \epsilon))$ and $T_2 = del((t, \epsilon), (\epsilon, t))$.

It is easily seen that, in both cases (1) and (2), the translations T_1 and T_2 have the same effect on any database state.

This led us to define a view update to be strongly deterministic if *all* its translations have the same effect on any given database state (see Definition 5.2 below). Moreover, in order to characterize strong determinism in terms of Inverses, we restrict relational expressions so that they do *not* contain several occurrences of the same relation symbol.

Definition 5.2 *Let $E(R_1, \ldots, R_n)$ be a relational expression, t a tuple defined on $sch(E)$, and $\mathcal{U}(t, E)$ a view update over E. $\mathcal{U}(t, E)$ is strongly deterministic, if for all instances $r_1, \ldots, r_n$ of $R_1, \ldots, R_n$ respectively, and for all translations T and T' of $\mathcal{U}(t, E)$, the composition of the effects of $eff(T, R_i)$ is equal to the composition of the effects of $eff(T', R_i)$, $i = 1, \ldots, n$.* □

Results in this section are stated under the following restrictions: relational expressions are positive and, moreover, contain no more than one occurrence of any relation symbol.

Then, based on Proposition 5.1, we have the following:

Proposition 5.4 *Let $E(R_1, \ldots, R_n)$ be a relational expression and t be a tuple defined on $sch(E)$. If for all i and j such that $i \neq j$, $R_i \neq R_j$, then: $Ins(t, E)$ is strongly deterministic iff $|Inv(t, E)| = 1$.*

Proof. Obvious. Proposition 5.4 is a consequence of Proposition 5.1. ■

Proposition 5.4 does not always hold if a relational expression E contains more than one occurrence of any relation symbol. If $E = R \cup R$, and t a tuple defined over $sch(E)$, we have $Inv(t, E) = \{(t, \epsilon), (\epsilon, t)\}$. On the other hand, the update $Ins(t, E)$ is strongly deterministic according to E since $T = ins((t,R),(\epsilon,R))$ and $T' = ins((\epsilon,R),(t,R))$ have the same effect on E.

When the view updates are deletions, the criterion of strong determinism of view updates depends on the form of the k-tuples belonging to the set of the inverse-images of the tuple to be deleted.

Before to state this result, we present the following lemma which states that, if E is a relational expression and t a tuple defined on $sch(E)$, for every non empty tuple t_0 belonging to one inverse-image of $Inv(t,E)$, there exists at least one translation of $Del(t,E)$ which containing it.

Lemma 5.1 *Let $E(R_1,\ldots,R_n)$ be a relational expression and t be a tuple defined on $sch(E)$ such that for every $i \neq j$, $R_i \neq R_j$, $i = 1,\ldots,n$, $j = 1,\ldots,n$.*
If $t_i \in N$ is such that $t_i \neq \epsilon$ with $N \in Inv(t,E)$, then there exists a translation $T = del((\tau_1,r_1),\ldots,(\tau_n,r_n))$ of $Del(t,E)$ such that $t_i \in \tau_i$.

Proof. Lemma 5.1 is a direct consequence of Proposition 5.3. Indeed, let $E(R_1,\ldots,R_n)$ be a relational expression, t be a tuple defined over $sch(E)$, and t_{i_0} a non empty tuple such that $(t_1,\ldots,t_{i_0},\ldots,t_n) \in Inv(t,E)$. In order built one translation $del((\tau_1,R_1),\ldots,(\tau_n,R_n))$ which contains t_{i_0}, it is sufficient to perform the following two operations:

1. First, choose the tuple $t_{i_0} \neq \epsilon$ ($t_{i_0} \in N$ and $N \in Inv(t,E)$) and put it in the set τ_{i_0}.
2. Built τ_{i_0} in step 1, then built the other sets τ_i (see Proposition 5.2).

■

We have then the following proposition:

Proposition 5.5 *Let $E(R_1,\ldots,R_n)$ be a relational expression and t a tuple defined on $sch(E)$. If for all i and j such that $i \neq j$, $R_i \neq R_j$, then: $Del(t,E)$ is strongly deterministic iff every N in $Inv(t,E)$ is unary.*

Proof. Let $E(R_1,\ldots,R_n)$ be a relational expression and t be a tuple defined on $sch(E)$.

1. Assume that $del((\tau_1,r_1),\ldots,(\tau_n,r_n))$ is the unique translation of the view update $Del(t,E)$ and show that every k-tuple N such that $N \in Inv(t,E)$, N is unary.

   Let $N \in Inv(t,E)$ not be a unary k-tuple. Hence, $N = (t_1,\ldots,t_i,\ldots,t_j,\ldots,t_n)$ with $t_i \neq \epsilon$, $t_j \neq \epsilon$ and $t_i \neq t_j$. Since t_i and t_j are non empty tuples and belonging to N, according to Lemma 5.1, there exists a translation T^1 such that $t_i \in \tau_i^1$ and there exists a translation T^2 such that $t_j \in \tau_j^2$. If $T^1 = T^2$, it implies that $N \in Inv_i^1 \cap Inv_j^2$, hence $T^1 \neq T^2$.

2. Assume that every k-tuple N such that $N \in Inv(t,E)$, N is unary and let's show that $Del(t,E)$ has one translation $del((\tau_1,r_1),\ldots,(\tau_n,r_n))$.

   Assume that $Del(t,E)$ has two translations $T = del((\tau_1,r_1),\ldots,(\tau_n,r_n))$ and $T' = del((\tau'_1,r_1),\ldots,(\tau'_n,r_n))$. Hence, there exists i such that $\tau_i \neq \tau'_i$. This implies that there exists one tuple t_i such that $t_i \in \tau_i$ and $t_i \notin \tau'_i$. Since $t_i \in \tau_i$, there exists $N_0 = (t_1,\ldots,t_i,\ldots,t_n) \in Inv(t,E)$ such that $N_0 \in Inv_i$ and $N_0 \notin Inv'_i$. Since $\bigcup Inv'_i = Inv(t,E)$ (because $\{Inv'_1,\ldots,Inv'_n\}$ is a partition of $Inv(t,E)$), there exists k such that $N_0 \in Inv'_k$. Then we have $N_{0_k} = t_k$ and $t_k \neq \epsilon$. Hence, the k-tuple $N_0 = (t_1,\ldots,t_i,\ldots,t_k,\ldots,t_n)$ contains two non empty tuples t_i and t_k.

■

We give now an example of a relational expression with more than one occurrence of a relation symbol showing that the above proposition does not hold in general.

Example 5.5 *Let $E = R \cap R$ be a relational expression and t be a tuple over $sch(E)$. $Inv(t,E) = \{(t,t)\}$ and thus, no k-tuple of $Inv(t,E)$ is unary. However, since $T = del((t,R),(\epsilon,R))$ and $T' = del((\epsilon,R),(t,R))$ are the two translations of $Del(t,E)$, according to Definition 5.2, $Del(t,E)$ is strongly deterministic.*

5.3.2 Weakly deterministic View Updates

We introduce the notion of quantity of change in a given database which is, intuitively, the number of tuples that are actually to be inserted in or deleted, when the translation of a given view update is processed against the current instance of the database.

Definition 5.3 *Let $E(R_1,\ldots,R_n)$ be a relational expression, t a tuple defined on $sch(E)$, and $T = upd((\tau_1,R_1),\ldots,(\tau_n,R_n))$ a translation of $\mathcal{U}(t,E)$.*
Given n instances $r_1,\ldots,r_n$ of $R_1,\ldots,R_n$ respectively, the measure of T is defined by:

- $\phi(T) = |\tau_1 - r_1| + \ldots + |\tau_n - r_n|$ *if T is an insert-transaction.*
- $\phi(T) = |\tau_1 \cap r_1| + \ldots + |\tau_n \cap r_n|$ *if T is a delete-transaction.* □

Definition 5.4 *Let $E(R_1,\ldots,R_n)$ be a relational expression, t be a tuple defined on $sch(E)$, $I = (r_1,\ldots,r_n)$ be an instance of $(R_1,\ldots,R_n)$ and $\mathcal{U}(t,E)$ be a view update of t over E.*
If $\mathcal{T}$ is the set of translations of $\mathcal{U}(t,E)$ then a translation T_0 of $\mathcal{T}$ guarantees the minimal change on I if, for every T of $\mathcal{T}$, we have: $\phi(T_0) \leq \phi(T)$. □

The following example illustrates the above definition.

Example 5.6 *Let $E = \Pi_{AC}(\sigma_{B\geq b}(R_1 \bowtie R_2))$ be a relational expression such that r_1 and r_2 are two instances of the relation symbols R_1 and R_2 defined respectively on the schemes AB and BC, and let ac be a tuple defined on $sch(E)$.*
Assume $dom_B = \{b,b'\}$ such that $b' \geq b$. Then we have $Inv(t,E) = \{(ab,bc),\ (ab',b'c)\}$.

1. *Thus* $T_1 = ins((ab, R_1), (bc, R_2))$ *and* $T_2 = ins((ab', R_1), (b'c, R_2))$ *are the only two translations of* $Ins(ac, E)$. *Assuming that* $r_1 = \{ab\}$ *and* $r_2 = \emptyset$, *we have:*

   $\phi(T_1) = |\{ab\} - r_1| + |\{bc\} - r_2| = 1$, *and* $\phi(T_2) = |\{ab'\} - r_1| + |\{b'c\} - r_2| = 2$.

   As $\phi(T_1) < \phi(T_2)$, T_1 *is the translation which guarantees the minimal change on the instance* $(\{ab\}, \emptyset)$.

2. *The four translations of* $Del(ac, E)$ *and their corresponding measures are:*

translation	*corresponding measure*
$T_1 = del((\{ab, ab'\}, R_1), (\emptyset, R_2))$	$\phi(T_1) = \|\{ab, ab'\} \cap r_1\| + \|\emptyset \cap r_2\| = 1$
$T_2 = del((\{ab\}, R_1), (\{b'c\}, R_2))$	$\phi(T_2) = \|\{ab\} \cap r_1\| + \|\{b'c\} \cap r_2\| = 1$
$T_3 = del((\{ab'\}, R_1), (\{bc\}, R_2))$	$\phi(T_3) = \|\{ab'\} \cap r_1\| + \|\{bc\} \cap r_2\| = 0$
$T_4 = del((\emptyset, R_1), (\{bc, b'c\}, R_2))$	$\phi(T_4) = \|\emptyset \cap r_1\| + \|\{bc, b'c\} \cap r_2\| = 0$

   Clearly, in this example the two translations T_3 *and* T_4 *guarantee the minimal change on the instance* $I = (r_1, r_2)$ *since they have the least measure.*

Given a relational expression and a view update $\mathcal{U}(t, E)$ of a tuple t over E, obviously, we see that, for each database instance I, there exists a translation of $\mathcal{U}(t, E)$ which guarantees the minimal change of I. So, we introduce the notion of *weak determinism* of updates according to a given database instance:

Definition 5.5 *Let* $E(R_1, \ldots, R_n)$ *be a relational expression defined on the database* $\mathcal{D}$ *and* $\mathcal{U}(t, E)$ *an update of the tuple* t *over* E. $\mathcal{U}(t, E)$ *is weakly deterministic according to an instance* $I = (r_1, \ldots, r_n)$ *iff for all translations* T *and* T' *of* $\mathcal{U}(t, E)$ *which guarantee the minimal change on* I, $T = T'$. □

5.4 Performing View Updates: an Overview

In the traditional relational model, where the views are not stored, when the view update has more than one translation, the view update is simply rejected. In our approach, given a relational expression $E(R_1, \ldots, R_n)$ over a database $\mathcal{D}$, and a tuple t over $sch(E)$, we propose a method to perform a view update $\mathcal{U}(t, E)$ under the restriction that $Inv(t, E)$ is a finite set. To this end, we consider the following three steps:

1. Test of the strong determinism of the view update

   - $\mathcal{U}(t, E) = Ins(t, E)$:
     If $|Inv(t, E)| = 1$, i.e. $Inv(t, E) = \{(t_1, \ldots, t_n)\}$, (see Proposition 5.4), then perform the database updates $ins((t_1, R_1), \ldots, (t_n, R_n))$.
   - $\mathcal{U}(t, E) = Del(t, E)$:
     If every k-tuple in $Inv(t, E)$ is unary (see Proposition 5.5), then build the unique translation from $Inv(t, E)$ using the algorithm described in Section 5.2 and perform the database updates $del((\tau_1, R_1), \ldots, (\tau_n, R_n))$.

2. If the test in item 1 above has failed, then test the weak determinism of the view update according to Definition 5.5.

   If there exists a unique view update translation T which guarantees minimal change of the database instance $r_1, \ldots, r_n$, then execute T on the database.

3. If steps 1 and 2 fail, then apply the method of [BL97] where the tuple to be inserted or deleted is explicitly stored in the view. We note that this step is closely related to the problem of view maintenance in the context of *materialized* views [BLT86, CW91].

6 Conclusion

We have presented a method for translating view updates into database updates using the inversion of relational algebra. We have proposed a way to compute the set of inverses of a given relational expression (excluding the difference), and we have shown that this set, coincides with the corresponding set of inverse-images. Moreover, when a relation occurs at most once in a given expression, we have characterized the notion of a view update translation and shown that there is a strong link between the set of translations and the set of inverses of the relational expression defining the view. We are currently studying the generalization of our results to the case of any relational expression, possibly containing set difference and several occurrences of the same relation.

Moreover, we have characterized two kinds of updates: (1) deterministic updates regardless to the database state, called strongly deterministic updates, and (2) deterministic updates according to some database state, called weakly deterministic updates.

Another research direction envisaged is to provide a deterministic method to performing the database updates which are view updates translations over the underlying database in the context of materialized views. We think that our results about strongly deterministic and weakly deterministic updates could certainly contribute to this research direction.

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* ADDISON-WESLEY, 1995.

[AT92a] P. Atzeni and R. Torlone. Updating Intensional Predicates in Datalog. *Data and Knowledge Engineering*, 8, 1992.

[AT92b] P. Atzeni and R. Torlone. Updating Relational Databases through Weak Instance Interfaces. *ACM TODS*, 17:718–745, 1992.

[BL97] Fadila Bentayeb and Dominique Laurent. Inversion de l'Algèbre Relationnelle et Mises à jour. Technical Report 09, LIFO, Orléans, France, Juin 1997.

[BLT86] José A. Blakeley, Per-Ake Larson, and Frank Wm. Tompa. Efficiently Updating Materialized Views. *ACM SIGMOD*, 1986.

[BS81] F. Bancilhon and N. Spyratos. Update Semantics of Relational Views. *ACM Trans. on Database Syst.*, 6(4):557–575, December 1981.

[CW91] Stefano Ceri and Jennifer Widom. Deriving Production Rules for Incremental View Maintenance. In *Proceedings of the 17th International conference on Very Large Data Bases*, Barcelona, September 1991.

[IL83] Tomasz Imielinski and Witold Lipski. Inverting Relational expressions - A Uniform and Natural Technique For Various Database Problems. *ACM SIGACT-SIGMOD*, pages 305–311, 1983.

[LPS92] D. Laurent, V. Phan Luong, and N. Spyratos. Deleted Tuples are Useful when Updating through Universal Scheme Interfaces. In *8th IEEE ICDE*, USA, February 1992.

[LS94] D. Laurent and N. Spyratos. A Partition Model Approach to Updating Universal Scheme Interfaces. *IEEE TKDE*, 6(2):316–330, Avril 1994.

[Ull89] J.D. Ullman. *Principles of Database and Kowledge-Base Systems*, volume 2. Computer Science Press, 1989.

[Vos90] G. Vossen. *Data Models, Database Languages and Database Management Systems.* ADDISON-WESLEY, 1990.

Publications récentes du LIFO
LIFO's recent publications

97-7 E. MELIN, B. RAFFIN, X. REBEUF, B. VIROT
A Unifying Approach of Data and Task Parallelism Based on Structural Clock Communication Paradigm (extended version)

97-8 S. ANANTHARAMAN, Editor
Proceedings of UNIF'97, International Workshop on Unification

97-9 F. BENTAYEB, D. LAURENT
Inversion de l'algrèbre relationnelle et mises à jour

97-10 I. PARFENOFF
An efficient parallel algorithm for maximum matching in P_4-tidy graphs

97-11 F. ROUSSEL, I. RUSU
An O(m2+mn) algorithm to recognize Meyniel graphs

97-12 M. BAHMA, G. HAINS
Prevision des performances de la θ-jointure et de la multi-jointure pour des machines SN

97-13 F. LOULERGUE, G. HAINS
Design of an event-oriented functionnal parallel language : the finite case

97-14 S. LIMET, F. SAUBION
A proof method for logic programming

97-15 E. MELIN, B. RAFFIN, X. REBEUF, B. VIROT
An Automatic Distribution of Sequential Code Fitting Irregular Data Accesses

Pour récupérer les rapports de recherche du LIFO / To get LIFO's research reports :

- Par WWW : http://www.univ-orleans.fr/SCIENCES/LIFO/Recherche/Publis/RR/index.html
- Par FTP : ftp://ftp-lifo.univ-orleans.fr/pub/RR
- Par courrier papier adressé à / By surface mail adressed to :
  Arnaud Lallouet, LIFO, Université d'Orléans
  4, rue Léonard de Vinci, BP 6759
  F-45067 Orléans cedex 2, FRANCE
  en mentionnant votre adresse.
- Par courrier électronique à / By e-mail to : Arnaud.Lallouet@lifo.univ-orleans.fr