Information-Optimal Reflections of View Updates on Relational Database Schemata

Stephen J. Hegner
Umeå University
Department of Computing Science
SE-901 87 Umeå, Sweden
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

Abstract

For the problem of reflecting an update on a database view to the main schema, the constant-complement strategies are precisely those which avoid all update anomalies, and so define the gold standard for well-behaved solutions to the problem. However, the families of view updates which are supported under such strategies are limited, so it is sometimes necessary to go beyond them, albeit in a systematic fashion. In this work, an investigation of such extended strategies is initiated for relational schemata. The approach is to characterize the information content of a database instance, and then require that the optimal reflection of a view update to the main schema embody the least possible change of information. To illustrate the utility of the idea, sufficient conditions for the existence of optimal insertions in the context of families of extended embedded implicational dependencies (XEIDs) are established. It is furthermore established that all such optimal insertions are equivalent up to a renaming of the new constant symbols which were introduced in support of the insertion.

1. Introduction

The problem of reflecting view updates to the main schema of a database system is a difficult one whose solution invariably involves compromise. The constant-complement approach [BS81] is exactly the one which avoids all so-called update anomalies [Heg04], and so is the gold standard for well-behaved strategies. On the other hand, it is also quite conservative regarding the updates which it admits.

Substantial research has been conducted on allowing more general view updates in a systematic fashion. In the classical relational context, much of this work, such as [DB82], [Kel85], [Lan90], [BL97], and [BL98], focuses upon translations via the relational algebra. In this work, a quite different, logic-based approach is undertaken. The fundamental point of departure is that an optimal reflection of a view update is one which minimizes the information change in

the main schema, with the information content of a database measured by the set of sentences in a certain family which it satisfies. An example will help clarify the main ideas.

1.1 Motivating example Let $\mathbf{E}_0$ be the relational schema with relations $R[ABC]$ and $S[CD]$, constrained by the inclusion dependency $R[C] \subseteq S[C]$. Regard a database as a set of ground atoms over the associated logic. For example, $M_{00} = \{R(a_0, b_0, c_0), R(a_1, b_1, c_1), S(c_0, d_0), S(c_1, d_1)\}$ is such a database. Now, let K be a set of constants in the underlying logical language, regarded as domain elements for this schema. The *information content* of a database M relative to K is the set of all positive (*i.e.*, no negation, explicit or implicit), existential, and conjunctive sentences which are implied by M. Using the notation to be introduced in 3.3, this information content is denoted $\mathsf{Info}\langle M, \mathsf{WFF}(\mathbf{E_0}, K, \exists\wedge+)\rangle$. A *basis* for this information content is a subset $\Phi \subseteq \mathsf{Info}\langle M, \mathsf{WFF}(\mathbf{E_0}, K, \exists\wedge+)\rangle$ such that Φ and $\mathsf{Info}\langle M, \mathsf{WFF}(\mathbf{E_0}, K, \exists\wedge+)\rangle$ are logically equivalent. For $K_{00} = \{a_0, a_1, b_0, b_1, c_0, c_1, d_0, d_1\}$, the set of all constant symbols of M_{00}, the set M_{00} itself is clearly a basis for $\mathsf{Info}\langle M_{00}, \mathsf{WFF}(\mathbf{E_0}, K_{00}, \exists\wedge+)\rangle$. On the other hand, with $K'_{00} = \{a_0, a_1, b_0, b_1, c_0, d_0\}$, a basis for $\mathsf{Info}\langle M_{00}, \mathsf{WFF}(\mathbf{E_0}, K'_{00}, \exists\wedge+)\rangle$ is $\{R(a_0, b_0, c_0), S(c_0, d_0), (\exists x)(\exists y)(R(a_1, b_1, x)\wedge S(x, y))\}$. Note that the constants in $K_{00} \setminus K'_{00}$ have been replaced by existentially quantified variables.

To see how this idea is useful in the context of view updates, let $\Pi_{R[AB]} = (R[AB], \pi_{R[AB]})$ be the view of $\mathbf{E}_0$ which projects $R[ABC]$ onto $R[AB]$ and which drops the relation S entirely. Consider M_{00} to be the initial state of schema $\mathbf{E}_0$; its image state in the view is then $N_{00} = \{R(a_0, b_0), R(a_1, b_1)\}$. Now, suppose that the view update $\mathsf{Insert}\langle R(a_2, b_2)\rangle$ is requested, so that $N_{01} = N_{00} \cup \{R(a_2, b_2)\}$ is the desired new view state, and consider $M_{01} = M_{00} \cup \{R(a_2, b_2, c_2), S(c_2, d_2)\}$ as a proposed reflection to the main schema $\mathbf{E}_0$. Relative to its entire set $K_{01} = \{a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2, d_0, d_1, d_2\}$ of constant symbols, a basis for $\mathsf{Info}\langle M_{01}, \mathsf{WFF}(\mathbf{E_0}, K_{01}, \exists\wedge+)\rangle$ is just M_{01} itself. Similarly, for $M_{02} = M_{00} \cup \{R(a_2, b_2, c_3), S(c_3, d_3)\}$ with $K_{02} = \{a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_3, d_0, d_1, d_3\}$ a basis for $\mathsf{Info}\langle M_{02}, \mathsf{WFF}(\mathbf{E_0}, K_{02}, \exists\wedge+)\rangle$ is just M_{02} itself. Observe that the proposed updates M_{01} and M_{02} are identical up to a renaming of the new constants. The utility of information measure is that it provides a means to recapture this idea formally; the information content of each, relative to the set K_{00} of constant symbols of the original state M_{00}, is the same. More precisely, $\mathsf{Info}\langle M_{01}, \mathsf{WFF}(\mathbf{E_0}, K_{00}, \exists\wedge+)\rangle = \mathsf{Info}\langle M_{02}, \mathsf{WFF}(\mathbf{E_0}, K_{00}, \exists\wedge+)\rangle$. A basis for each of these is $I_1 = M_{00} \cup \{(\exists x)(\exists y)(R(a_2, b_2, x)\wedge S(x, y))\}$. In effect, this measure is indifferent to whether c_2 and d_2 or c_3 and d_3 are used.

Now, consider the alternative solution $M_{03} = M_{00} \cup \{R(a_2, b_2, c_3), S(c_3, d_1)\}$ to this view-update problem. A basis for $\mathsf{Info}\langle M_{03}, \mathsf{WFF}(\mathbf{E_0}, K_{00}, \exists\wedge+)\rangle$ is $I_3 = M_{00} \cup \{(\exists x)(R(a_2, b_2, x)\wedge S(x, d_1))\}$, which is strictly stronger than I_1, since $(\exists x)(R(a_2, b_2, x)\wedge S(x, d_1)) \models (\exists x)(\exists y)(R(a_2, b_2, x)\wedge S(x, y)))$, but not conversely. Thus, relative to the information measure defined by K_{00}, M_{03} adds more information to M_{00} than does M_{01} or M_{02}. Similarly, $M_{04} = M_{00} \cup \{R(a_2, b_2, c_0))\}$ adds more information than does M_{01} or M_{02}, since a basis for its information content is just M_{04} itself, which is stronger than I_1, since $R(a_2, b_2, c_0)\wedge S(c_0, d_0) \models (\exists x)(\exists y)(R(a_2, b_2, x)\wedge S(x, y)))$, but not conversely.

The first and primary measure of quality of a reflected update is the change of information content which is induces. Under this measure, M_{01} and M_{02} are equivalent, and both are superior to either of M_{03} or M_{04}. However, this is by itself not quite adequate. Rather, there is an additional measure of quality which must be taken into account. To illustrate, consider

the proposed solution $M_{05} = M_{01} \cup M_{02} = M_{00} \cup \{R(a_2, b_2, c_2), R(a_2, b_2, c_3), S(c_2, d_2), S(c_3, d_3)\}$ to this update problem. It has the same information content, I_1, relative to K_{00}, as do M_{01} and M_{02}. The information measure cannot distinguish the insertion of two new tuples with completely new constants from the insertion of just one. However, it is clear that M_{05} should be considered inferior to both M_{01} and M_{02} as a solution to the given update problem, since it is a proper superset of each. Therefore, a second criterion of quality is invoked; namely that no solution whose set of changes is a proper superset of those of another can be considered to be superior. It is important to emphasize that it is a inclusion relationship which applies here, and not simply a counting argument. For example, consider again the proposed solution M_{04}. From a strict counting point of view, M_{04} involves fewer changes than do M_{01} or M_{02}. However, neither M_{01} nor M_{02} is a superset of M_{04}. Thus, the superiority of M_{01} and M_{02} is not contradicted. In other words, only solutions which are *tuple minimal*, in the sense that no proper subset of the changes is also an admissible solution, are permitted.

The main modelling premise of this paper is that the quality of a view update can be measured by the amount of change in information content which it induces, and so an optimal reflection of a view update request is one which is both tuple minimal and which induces the least amount of change of information content. Under this premise, both M_{01} and M_{02} are superior to either of M_{03} or M_{04}. Furthermore, since M_{01} and M_{02} induce the same change in information content, they are equivalent. In Section 3, it is established that, under suitable conditions, all such optimal solutions are equivalent, up to a renaming of the constant symbols. In Section 4, it is established, again under suitable conditions, that for insertions, a minimal solution (in terms of change of information content) must be optimal. These conditions include in particular schemata constrained by XEIDs — the extended embedded implicational dependencies of Fagin [Fag82], which include virtually all other classes of classical database dependencies.

In summary, there are two conditions which must be met for optimality of a proposed update reflection u. First, it must be *tuple minimal*, in that there can be no other solution whose set of changes is a proper subset of those of u. Second, it must be *information least* in terms of a specific set of sentences. This approach applies also to deletions and updates which involve both insertion and deletion, and this generality is incorporated into the formalism which is presented. However, for deletions the two measures will coincide, since no new constants are involved in a deletion.

2. The Relational Model

2.1 Two representations of the traditional relational model In the traditional approach to the relational model [PDGV89] [AHV95], the starting point is a set $\mathcal{A}$ of attributes, a finite nonempty set Rels of of relation symbols, and a function $\mathsf{Ar} : \mathsf{Rels} \to \mathcal{A}$ which assigns to each $R \in \mathsf{Rels}$ a set $\mathsf{Ar}(R) \subseteq \mathcal{A}$, the *attributes* of R. Furthermore, to each $A \in \mathcal{A}$ is associated a (usually countably infinite) *domain* $\mathsf{dom}(\mathcal{A})$. An *$R$-tuple* is then a function $t : \mathsf{Ar}(R) \to \mathsf{dom}(\mathcal{A})$, and a *relational database* over $(\mathcal{A}, \mathsf{dom})$ is a collection of R-tuples for each $R \in \mathcal{A}$.

From a logical point of view, there are two common interpretations of the domain elements. In logic programming, they are usually taken to be constant symbols of the underlying logic. Tuples then become ground atoms, with (extensional) databases finite sets of such atoms [CGT89]. Furthermore, in that context, the set of all constant symbols is usually taken to be

finite, in order to allow first-order axiomatization of domain closure. On the other hand, for model-theoretic constructions, such as those of [Fag82], it is necessary to interpret the relational domain elements as members of the underlying set of a structure [Mon76, Def. 11.1], and to allow these sets to be countably infinite is essential. Both representations of tuples are crucial to this paper, so it is necessary to establish a bijective correspondence between them. To accomplish this, it is first necessary to establish a bijective correspondence between the elements of the structure and the constant symbols. This requires some care, since this condition cannot be stated using finite sentences in first-order logic. The solution employed in this paper is to use the same countable underlying set for all structures, and then to fix a bijection between the constant symbols and the structure elements. This bijection is invariant across all the main schemata and the view to be updated. Once such a bijection of elements and constants is established, a corresponding bijection of tuples and ground atoms, and consequently of databases represented in these two distinct formats, follows directly.

2.2 Relational contexts and constant interpretations A relational context contains the logical information which is shared amongst the distinct schemata and the corresponding database mappings: the attribute names, the variables, and constant symbols. Formally, a relational context $\mathcal{D}$ consists of attribute names $\mathcal{A}_\mathcal{D}$, variables $\mathsf{Vars}(\mathcal{D})$, and for each $A \in \mathcal{A}_\mathcal{D}$, a set $\mathsf{Const}_\mathcal{D}(A)$ of constant symbols. The variables $\mathsf{Vars}(\mathcal{D})$ are further partitioned into two disjoint sets; a countable set $\mathsf{GenVars}(\mathcal{D}) = \{x_0, x_1, x_2, \ldots\}$ of *general variables*, and special $\mathcal{A}_\mathcal{D}$-indexed set $\mathsf{AttrVars}(\mathcal{D}) = \{x_A \mid A \in \mathcal{A}_\mathcal{D}\}$ of *attribute variables*. The latter are used in the definition of interpretation mappings; see 2.5 for details.

A constant interpretation provides a model-theoretic interpretation for the constant symbols, in the sense of [Mon76, Def. 11.1]. It is also fixed over all databases of all schemata. Formally, a *constant interpretation* for the relational context $\mathcal{D}$ is a pair $\mathcal{I} = (\mathsf{Dom}_\mathcal{I}, \mathsf{IntFn}_\mathcal{I})$ in which $\mathsf{Dom}_\mathcal{I}$ is a countably infinite set, called the *domain* of $\mathcal{I}$, and $\mathsf{IntFn}_\mathcal{I} : \mathsf{Const}(\mathcal{D}) \to \mathsf{Dom}_\mathcal{I}$ is a bijective function, called the *interpretation function* of $\mathcal{I}$. Note that the latter effectively stipulates the following two well-known conditions [GN87, p. 120]:

Domain closure: $(\forall x)(\bigvee_{a \in \mathsf{Const}(\mathcal{D})} x = a)$ $(\mathsf{DCA}(\mathcal{D}))$

Unique naming: $(\neg(a = b))$ for distinct $a, b \in \mathsf{Const}(\mathcal{D})$ $(\mathsf{UNA}(\mathcal{D}))$

Since there are countably many constant symbols, the domain closure axiom is not a finite disjunction. This is not a problem however, since it is never used in an otherwise first-order set of constraints. Except for the extended tuple databases of 4.4, in which this constraint is relaxed, the assignment of domain values to constants is fixed, and so it is not necessary to verify that it holds.

For $A \in \mathcal{A}_\mathcal{D}$, define $\mathsf{Dom}_\mathcal{I}(A) = \{z \in \mathsf{Dom}_\mathcal{I} \mid \mathsf{IntFn}_I(z) \in \mathsf{Const}_\mathcal{D}(A)\}$. Thus, $\mathsf{Dom}_\mathcal{I}(A)$ is the set of all domain values which are associated with attribute A.

As a notational convention, from this point on, unless stated otherwise, fix relational context $\mathcal{D}$ and a constant interpretation $\mathcal{I} = (\mathsf{Dom}_\mathcal{I}, \mathsf{IntFn}_\mathcal{I})$ for it.

2.3 Tuples and databases An *unconstrained relational schema* over $(\mathcal{D}, \mathcal{I})$ is a pair $\mathbf{D} = (\mathsf{Rels}(\mathbf{D}), \mathsf{Ar}_\mathbf{D})$ in which $\mathsf{Rels}(\mathbf{D})$ is finite set of relational symbols and $\mathsf{Ar}_\mathbf{D} : \mathsf{Rels}(\mathbf{D}) \to \mathbf{2}^{\mathcal{A}_\mathcal{D}}$ a function which assigns an *arity*, a set of distinct attributes from $\mathcal{A}_\mathcal{D}$, to each $R \in \mathsf{Rels}(\mathbf{D})$.

It is now possible to address the problem of modelling databases in the two distinct ways identified in 2.1 above. For $R \in \mathsf{Rels}(\mathbf{D})$, an *$R$-tuple* is a function t on $\mathsf{Ar}_{\mathbf{D}}(R)$ with the property that $t[A] \in \mathsf{Dom}_I(A)$ for every $A \in \mathsf{Ar}_{\mathbf{D}}$. Similarly, an *$R$-atom* is such a function with the property that $t[A] \in \mathsf{Const}_{\mathcal{D}}(A) \cup \mathsf{GenVars}(\mathcal{D}) \cup \{x_A\}$. A *ground R-atom* contains no variables, so $t[A] \in \mathsf{Const}_{\mathcal{D}}(A)$. The set of all R-tuples (resp. R-atoms, resp. ground R-atoms) is denoted $\mathsf{Tuples}(\mathbf{D})$, (resp. $\mathsf{Atoms}(\mathbf{D})$, resp. $\mathsf{GrAtoms}(\mathbf{D})$). In view of 2.2 above, it is easy to see that there is a bijective correspondence between $\mathsf{GrAtoms}(\mathbf{D})$ and $\mathsf{Tuples}(\mathbf{D})$ given by $t(a_1, a_2, \ldots, a_n) \mapsto t(\mathsf{IntFn}_I(a_1), \mathsf{IntFn}_I(a_2), \ldots \mathsf{IntFn}_I(a_n))$.

It will be necessary to work with sets of R-tuples and sets of R-atoms, with R ranging over distinct relation symbols. A *$\mathbf{D}$-tuple* is an R-tuple for some $R \in \mathsf{Rels}(\mathbf{D})$, with the set of all $\mathbf{D}$-tuples denoted $\mathsf{Tuples}(\mathbf{D})$. A *tuple database for* $\mathbf{D}$ is a finite subset of $\mathsf{Tuples}(\mathbf{D})$, with the set of all tuple databases for $\mathbf{D}$ denoted $\mathsf{TDB}(\mathbf{D})$. The $\mathbf{D}$-atoms and ground $\mathbf{D}$-atoms are defined analogously, with the corresponding sets denoted $\mathsf{Atoms}(\mathbf{D})$ and $\mathsf{GrAtoms}(\mathbf{D})$, respectively. An *atom database for* $\mathbf{D}$ is a finite subset of $\mathsf{GrAtoms}(\mathbf{D})$; the set of all atom databases for $\mathbf{D}$ is denoted $\mathsf{DB}(\mathbf{D})$.

In the above definitions, it is necessary to be able to recover the associated relation from a tuple, and so *tagging* is employed, in which tuples are marked with the associated relation. Formally, this is accomplished by introducing a new attribute $\mathsf{RName} \notin \mathcal{A}_{\mathcal{D}}$, and then regarding an R-tuple not as a function t just on $\mathsf{Ar}_{\mathbf{D}}(R)$ but rather as one on on $\{\mathsf{RName}\} \cup \mathsf{Ar}_{\mathbf{D}}(R)$ with the property that $t[\mathsf{RName}] = R$. Tagging of R-atoms is defined analogously; both will be used from this point on throughout the paper. Nevertheless, in writing tuples, the more conventional notation $R(\tau_1, \tau_2, \ldots, \tau_n)$ will be used in lieu of the technically more correct $(R, \tau_1, \tau_2, \ldots, \tau_n)$, although tags will be used in formal constructions.

For the product construction of 4.5, it is necessary to restrict attention to nonempty databases. To this end, call $M \in \mathsf{TDB}(\mathbf{D})$ (resp. $M \in \mathsf{DB}(\mathbf{D})$) *relationwise nonempty* if for each $R \in \mathsf{Rels}(\mathbf{D})$, there is at least one R-tuple (resp. R-atom) in M, and define $\mathsf{RNeTDB}(\mathbf{D})$ (resp. $\mathsf{RNeDB}(\mathbf{D})$) to be the set of all relationwise nonempty members of $\mathsf{TDB}(\mathbf{D})$ (resp. $\mathsf{DB}(\mathbf{D})$).

The first-order language associated with the relational schema $\mathbf{D}$ is defined in the natural way; however, it is useful to introduce some notation which identifies particular sets of formulas. Define $\mathsf{WFF}(\mathbf{D})$ to be the set of all well-formed first-order formulas with equality, in the language whose set of relational symbols is $\mathsf{Rels}(\mathbf{D})$, whose set of constant symbols is $\mathsf{Const}(\mathcal{D})$, and which contains no non-nullary function symbols. The variables are those of $\mathcal{D}$. Additional arguments may be given to restrict this set. If $S \subseteq \mathsf{Const}(\mathcal{D})$, then $\mathsf{WFF}(\mathbf{D}, S)$ denotes the formulas in $\mathsf{WFF}(\mathbf{D})$ which involve only constant symbols from S. In particular, $\mathsf{WFF}(\mathbf{D}, \emptyset)$ denotes the set of formulas which do not contain constant symbols. Arguments are also used to limit the logical connectives. $\mathsf{WFF}(\mathbf{D}, \exists+)$ identifies those formulas which are built up from the connectives $\wedge$ and $\vee$, using at most existential quantifiers. $\mathsf{WFF}(\mathbf{D}, \exists\wedge+)$ enforces the further restriction that disjunction is not allowed. It will be furthermore assumed, in $\mathsf{WFF}(\mathbf{D}, \exists\wedge+)$, that equality atoms (*i.e.*, atoms of the form $x_i = x_j$ and $x_i = a$) are not allowed. This is not an essential limitation; such equality can always be represented by setting the terms to be equal in the atoms in which they are used. These notations may be combined, with the obvious semantics. For example, $\mathsf{WFF}(\mathbf{D}, \emptyset, \exists\wedge+)$ denotes the members of $\mathsf{WFF}(\mathbf{D}, \exists\wedge+)$ which do not involve constant symbols.

$\mathsf{WFS}(\mathbf{D})$ denotes the subset of $\mathsf{WFF}(\mathbf{D})$ consisting of sentences; that is, formulas with no free variables. The conventions regarding additional arguments applies to sets of sentences as

well. For example, $\mathsf{WFS}(\mathbf{D}, \emptyset, \exists\wedge+)$ is the subset of $\mathsf{WFF}(\mathbf{D}, \emptyset, \exists\wedge+)$ consisting of sentences.

$M \in \mathsf{TDB}(\mathbf{D})$ is an $\mathcal{I}$-*model* of $\varphi \in \mathsf{WFS}(\mathbf{D})$ if it is a model of φ in the ordinary sense which furthermore interprets the constant symbols according to $\mathcal{I}$. The set of all $\mathcal{I}$-models of φ is denoted $\mathsf{Mod}_{\mathcal{I}}(\varphi)$. In view of the bijection $\mathsf{GrAtoms}(\mathbf{D}) \to \mathsf{Tuples}(\mathbf{D})$ identified in 2.3 above, it is possible to identify $\mathcal{I}$-models with finite sets of ground atoms. More precisely, define the *atomic* $\mathcal{I}$-*models* of φ to be $\mathsf{AtMod}_{\mathcal{I}}(\varphi) = \{M \in \mathsf{DB}(\mathbf{D}) \mid M \cup \{\varphi\} \cup \mathsf{UNA}(\mathcal{D}) \text{ is consistent}\}$. Clearly, $\mathsf{IntFn}_{\mathcal{I}}(M) \in \mathsf{Mod}_{\mathcal{I}}(\varphi)$ iff $M \in \mathsf{AtMod}_{\mathcal{I}}(\varphi)$. The relationally nonempty versions $\mathsf{RNeMod}_{\mathcal{I}}(\varphi)$ and $\mathsf{RNeAtMod}_{\mathcal{I}}(\varphi)$ are defined analogously. Furthermore, all of these definitions of model extend to sets Φ of sentences in the obvious way; notation such as $\mathsf{Mod}_{\mathcal{I}}(\Phi)$, $\mathsf{RNeMod}_{\mathcal{I}}(\Phi)$, $\mathsf{AtMod}_{\mathcal{I}}(\Phi)$, and $\mathsf{RNeAtMod}_{\mathcal{I}}(\Phi)$ will be used throughout.

2.4 Schemata with constraints and constrained databases A *relational schema* over $(\mathcal{D}, \mathcal{I})$ is a triple $\mathbf{D} = (\mathsf{Rels}(\mathbf{D}), \mathsf{Ar}_{\mathbf{D}}, \mathsf{Constr}(\mathbf{D}))$ in which $(\mathsf{Rels}(\mathbf{D}), \mathsf{Ar}_{\mathbf{D}})$ is an unconstrained relational schema over $(\mathcal{D}, \mathcal{I})$ and $\mathsf{Constr}(\mathbf{D}) \subseteq \mathsf{WFS}(\mathbf{D}, \emptyset)$ is the set of *dependencies* or *constraints* of $\mathbf{D}$. Note that constant symbols are not allowed in the constraints.

In representing a database as a set of $\mathbf{D}$-atoms, the closed-world assumption is implicit. On the other hand, to express what it means for such a representation to satisfy a set of constraints, it is necessary to state explicitly which atoms are not true as well. Formally, for $M \in \mathsf{DB}(\mathbf{D})$, define the *diagram* of M to be $\mathsf{Diagram}_{\mathbf{D}}(M) = M \cup \{\neg t \mid t \in \mathsf{GrAtoms}(\mathbf{D}) \setminus M\}$. Define the *legal* (or *constrained*) *databases* of $\mathbf{D}$ to be $\mathsf{LDB}(\mathbf{D}) = \{M \in \mathsf{DB}(\mathbf{D}) \mid \mathsf{Diagram}_{\mathbf{D}}(M) \cup \mathsf{Constr}(\mathbf{D}) \text{ has an } \mathcal{I}\text{-model}\}$ and the *nonempty legal databases* to be $\mathsf{RNeLDB}(\mathbf{D}) = \mathsf{LDB}(\mathbf{D}) \cap \mathsf{RNeDB}(\mathbf{D})$.

2.5 Database morphisms and views Database morphisms are defined using expressions in the relational calculus; more formally, they are interpretations of the theory of the view into the theory of the main schema [JAK82]. Let $\mathbf{D}_1$ and $\mathbf{D}_2$ be relational schemata over $(\mathcal{D}, \mathcal{I})$. Given $R \in \mathsf{Rels}(\mathbf{D}_2)$, an *interpretation* for R into $\mathbf{D}_1$ is a $\varphi \in \mathsf{WFF}(\mathbf{D})$ in which precisely the variables $\{x_A \mid A \in \mathsf{Ar}_{\mathbf{D}}(R)\}$ are free, with x_A is used to mark the position in the formula which is bound to attribute A. The set of all interpretations of R into $\mathbf{D}_1$ is denoted $\mathsf{Interp}(R, \mathbf{D}_1)$. A *syntactic morphism* $f : \mathbf{D}_1 \to \mathbf{D}_2$ is a family $f = \{f^R \mid R \in \mathsf{Rels}(\mathbf{D}_2) \text{ and } f^R \in \mathsf{Interp}(R, \mathbf{D}_1)\}$.

Let $t \in \mathsf{GrAtoms}(R, \mathbf{D}_2)$. The *substitution* of t into f, denoted $\mathsf{Subst}\langle f, t\rangle$, is the sentence in $\mathsf{WFS}(\mathbf{D}_1)$ obtained by substituting $t[A]$ for x_A, for each $A \in \mathsf{Ar}_{\mathbf{D}}(R)$. For $M \in \mathsf{DB}(\mathbf{D}_1)$, define $f(M) = \{t \in \mathsf{Atoms}(\mathbf{D}_2) \mid \mathsf{Subst}\langle f, t\rangle \cup \mathsf{Diagram}_{\mathbf{D}_1}(M) \text{ has an } \mathcal{I}\text{-model}\}$. f is called a *semantic morphism* if it maps legal databases to legal databases; formally, $f(M) \in \mathsf{LDB}(\mathbf{D}_2)$ for each $M \in \mathsf{LDB}(\mathbf{D}_1)$.

Say that f is of class $\exists+$ (resp. $\exists\wedge+$) if each $f^R \in \mathsf{WFF}(\mathbf{D}_1, \exists+)$ (resp. $f^R \in \mathsf{WFF}(\mathbf{D}_1, \exists\wedge+)$). It is easy to see that if f is of class $\exists+$ (resp. $\exists\wedge+$), then for each $t \in \mathsf{Atoms}(\mathbf{D}_2)$, $\mathsf{Subst}\langle f, t\rangle \in \mathsf{WFS}(\mathbf{D}_1, \exists+)$ (resp. $\mathsf{Subst}\langle f, t\rangle \in \mathsf{WFS}(\mathbf{D}_1, \exists\wedge+)$).

Let $\mathbf{D}$ be a relational schema over $(\mathcal{D}, \mathcal{I})$. A *(relational) view* of $\mathbf{D}$ is a pair $\Gamma = (\mathbf{V}, \gamma)$ in which $\mathbf{V}$ is a relational schema over $(\mathcal{D}, \mathcal{I})$ and $\gamma : \mathbf{D} \to \mathbf{V}$ is a semantic morphism which is furthermore *semantically surjective* in the sense that for every $N \in \mathsf{LDB}(\mathbf{V})$, there is an $M \in \mathsf{LDB}(\mathbf{D})$ with $f(M) = N$. Γ is of class $\exists+$ (resp. class $\exists\wedge+$) precisely in the case that γ is of that same class.

3. The General Theory of Updates

In this section, the general ideas concerning the information content of a database state, and the ideas of optimizing an update relative to such content, are developed. It is furthermore established that for a wide class of schemata and views, all optimal updates are isomorphic in a natural way.

3.1 Notational convention Throughout the rest of this paper, unless stated specifically to the contrary, take $\mathbf{D}$ to be a relational schema over $(\mathcal{D}, \mathcal{I})$ and $\Gamma = (\mathbf{V}, \gamma)$ to be a (relational) view of $\mathbf{D}$.

For X an entity (for example, an atom, a formula, a database, etc.), $\mathsf{ConstSym}(X)$ denotes the set of all $a \in \mathsf{Const}(\mathcal{D})$ which occur in X. Similarly, $\mathsf{Vars}(X)$ denotes the set of all variables which occur in X. This will not be formalized further, but the meaning should always be unambiguous.

3.2 Updates and reflections An *update* on $\mathbf{D}$ is a pair $(M_1, M_2) \in \mathsf{LDB}(\mathbf{D}) \times \mathsf{LDB}(\mathbf{D})$. M_1 is the current state, and M_2 the new state. It is an *insertion* if $M_1 \subseteq M_2$, and a *deletion* if $M_2 \subseteq M_1$.

To describe the situation surrounding an update request on Γ, it is sufficient to specify the current state M_1 of the main schema and the desired new state N_2 of the view schema $\mathbf{V}$. The current state of the view can be computed as $\gamma(M_1)$; it is only the new state M_2 of the main schema (subject to $N_2 = \gamma(M_2)$) which must be obtained from an update strategy. Formally, an *update request* from Γ to $\mathbf{D}$ is a pair (M_1, N_2) in which $M_1 \in \mathsf{LDB}(\mathbf{D})$ (the old state of the main schema) and $N_2 \in \mathsf{LDB}(\mathbf{V})$ (the new state of the view schema). If $\gamma(M_1) \subseteq N_2$, it is called an *insertion request*, and if $N_2 \subseteq \gamma(M_1)$, it is called a *deletion request*. Collectively, insertion requests and deletion requests are termed *unidirectional update requests*. A *realization* of (M_1, N_2) along Γ is an update (M_1, M_2) on $\mathbf{D}$ with the property that $\gamma(M_2) = N_2$. The update (M_1, M_2) is called a *reflection* (or *translation*) of the view update $(\gamma(M_1), N_2)$. The set of all realizations of (M_1, N_2) along Γ is denoted $\mathsf{UpdRealiz}\langle M_1, N_2, \Gamma\rangle$. The subset of $\mathsf{UpdRealiz}\langle M_1, N_2, \Gamma\rangle$ consisting of insertions (resp. deletions) is denoted $\mathsf{InsRealiz}\langle M_1, N_2, \Gamma\rangle$ (resp. $\mathsf{DelRealiz}\langle M_1, N_2, \Gamma\rangle$.

3.3 Information content and Φ-equivalence Let $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ and let $M \in \mathsf{DB}(\mathbf{D})$. The *information content* of M relative to Φ is the set of all sentences in Φ which are true for M. More precisely, $\mathsf{Info}\langle M, \Phi\rangle = \{\varphi \in \Phi \mid M \in \mathsf{AtMod}_{\mathcal{I}}(\varphi)\}$. For $\varphi \in \mathsf{WFS}(\mathbf{D})$, $\mathsf{Info}\langle M, \varphi\rangle$ denotes $\mathsf{Info}\langle M, \{\varphi\}\rangle$. M_1 and M_2 are Φ-*equivalent* if they have the same information content relative to Φ; *i.e.*, $\mathsf{Info}\langle M_1, \Phi\rangle = \mathsf{Info}\langle M_2, \Phi\rangle$.

3.4 Update difference and optimal reflections The update difference of an update (M_1, M_2) on $\mathbf{D}$ with respect to a set $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ is a measure of how much M_2 differs from M_1 in terms of satisfaction of the sentences of Φ. Formally, the *positive* (Δ^+), *negative* (Δ^-),

and total (Δ) update differences of (M_1, M_2) with respect to Φ are defined as follows:

$$\Delta^+\langle(M_1, M_2), \Phi\rangle = \mathsf{Info}\langle M_2, \Phi\rangle \setminus \mathsf{Info}\langle M_1, \Phi\rangle$$
$$\Delta^-\langle(M_1, M_2), \Phi\rangle = \mathsf{Info}\langle M_1, \Phi\rangle \setminus \mathsf{Info}\langle M_2, \Phi\rangle$$
$$\Delta\langle(M_1, M_2), \Phi\rangle = \Delta^+\langle(M_1, M_2), \Phi\rangle \cup \Delta^-\langle(M_1, M_2), \Phi\rangle$$

Note that, given $\varphi \in \Delta\langle(M_1, M_2), \Phi\rangle$, it is always possible to determine whether $\varphi \in \Delta^+\langle(M_1, M_2), \Phi\rangle$ or $\varphi \in \Delta^-\langle(M_1, M_2), \Phi\rangle$ by checking whether or not $M_1 \in \mathsf{AtMod}_{\mathcal{I}}(\varphi)$. Given an update request (M_1, N_2), the quality of a realization (M_1, M_2) is measured by its update difference. Formally, let $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$, let (M_1, N_2) be an update request from Γ to $\mathbf{D}$, let $T \subseteq \mathsf{UpdRealiz}\langle M_1, N_2, \Gamma\rangle$, and let $(M_1, M_2) \in T$.

(a) (M_1, M_2) is *minimal* in T with respect to Φ if for any $(M_1, M_2') \in T$, if $\Delta\langle(M_1, M_2'), \Phi\rangle \subseteq \Delta\langle(M_1, M_2), \Phi\rangle$, then $\Delta\langle(M_1, M_2'), \Phi\rangle = \Delta\langle(M_1, M_2), \Phi\rangle$.

(b) (M_1, M_2) is *least* in T with respect to Φ if for all $(M_1, M_2') \in T$, $\Delta\langle(M_1, M_2), \Phi\rangle \subseteq \Delta\langle(M_1, M_2'), \Phi\rangle$.

3.5 Information-monotone sentences and update classifiers For the above definitions of minimal and least to be useful, it is necessary to place certain restrictions on the nature of Φ. As a concrete example of the problems, define $\mathsf{GrAtoms}^\neg(\mathbf{D}) = \{\neg t \mid t \in \mathsf{GrAtoms}(\mathbf{D})\}$, with $\mathsf{GrAtoms}^\pm(\mathbf{D}) = \mathsf{GrAtoms}(\mathbf{D}) \cup \mathsf{GrAtoms}^\neg(\mathbf{D})$. In the context of 3.4 above, it is easy to see that every reflection (M_1, M_2) is minimal with respect to $\mathsf{GrAtoms}^\pm(\mathbf{D})$, while only identity updates (with $M_1 = M_2$) are least. Any $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ with the property that $\mathsf{GrAtoms}^\pm(\mathbf{D}) \subseteq \Phi$ will have this same property. The problem is that the sentences in $\mathsf{GrAtoms}^\neg(\mathbf{D})$ are not information monotone; adding new tuples reduces the information content. The sentence $\varphi \in \mathsf{WFS}(\mathbf{D})$ is *information monotone* if for any $M_1, M_2 \in \mathsf{DB}(\mathbf{D})$ if $M_1 \subseteq M_2$, then $\mathsf{Info}\langle M_1, \varphi\rangle \subseteq \mathsf{Info}\langle M_2, \varphi\rangle$. The set $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ is *information monotone* if each $\varphi \in \Phi$ has this property. Any $\varphi \in \mathsf{WFS}(\mathbf{D})$ which does not involve negation, either explicitly or implicitly (via implication, for example), is information monotone. Thus, in particular, for any $S \subseteq \mathsf{Const}(\mathcal{D})$, $\mathsf{WFS}(\mathbf{D}, S, \exists+)$, $\mathsf{WFS}(\mathbf{D}, S, \exists\wedge+)$, and $\mathsf{GrAtoms}(\mathbf{D})$ all consist of information-monotone sentences. The total absence of negation is not necessary, however. Sentences which allow negation of equality terms (*e.g.*, $\neg(x_i = x_j)$) but only existential quantification are also information monotone.

An *update classifier* for $\mathbf{D}$ is simply a set Σ of information-monotone sentences. The idea is simple: updates which involve less change of information are to be preferred to those which involve more. However, as illustrated in the example of 1.1, there are two distinct measures of optimality. On the one hand, an optimal realization (M_1, M_2) of an update request (M_1, N_2) must be least with respect to the update classifier, which in that example is $\mathsf{WFS}(\mathbf{D}, \mathsf{ConstSym}(M_{00}), \exists\wedge+)$. Unfortunately, this measure cannot always eliminate solutions which contain two "isomorphic" copies of the same update, such as M_{05} of that example. To remedy this, the update must also be minimal with respect to $\mathsf{Atoms}(\mathbf{D})$; or, equivalently, with respect to the symmetric difference $M_1 \triangle M_2 = (M_1 \setminus M_2) \cup (M_2 \setminus M_1)$. Formally, let (M_1, N_2) be an update request from Γ to $\mathbf{D}$, let $T \subseteq \mathsf{UpdRealiz}\langle M_1, N_2, \Gamma\rangle$, and let $(M_1, M_2) \in T$.

(a) (M_1, M_2) is $\langle\Sigma, T\rangle$-*admissible* if it is minimal in T with respect to both Σ and $\mathsf{Atoms}(\mathbf{D})$.

(b) (M_1, M_2) is $\langle\Sigma, T\rangle$-*optimal* if it is $\langle\Sigma, T\rangle$-admissible and least in T with respect to Σ.

Roughly, (M_1, M_2) is admissible if no other realization is better, and it is optimal if it is better than all others, up to the equivalence defined by Σ. Observe that if some update request is $\langle \Sigma, T \rangle$-optimal, then all $\langle \Sigma, T \rangle$-admissible update requests are $\langle \Sigma, T \rangle$-optimal.

As a notational shorthand, if $T = \mathsf{InsRealiz}\langle M_1, N_2, \Gamma \rangle$ (resp. $T = \mathsf{DelRealiz}\langle M_1, N_2, \Gamma \rangle$), that is, if T is the set of all possible insertions (resp. deletions) which realize (M_1, N_2), then $\langle \Sigma, T \rangle$-admissible and $\langle \Sigma, T \rangle$-optimal will be abbreviated to $\langle \Sigma, \uparrow \rangle$-admissible and $\langle \Sigma, \uparrow \rangle$-optimal (resp. $\langle \Sigma, \downarrow \rangle$-admissible and $\langle \Sigma, \downarrow \rangle$-optimal).

3.6 Examples of update classifiers For $M_1 \in \mathsf{LDB}(\mathbf{D})$, the *standard M_1-based update classifier* is $\mathsf{StdUCP}(\mathbf{D}, M_1) = \mathsf{WFS}(\mathbf{D}, \mathsf{ConstSym}(M_1), \exists\wedge+)$. As illustrated in 1.1, this classifier is appropriate for characterizing optimal insertions. Because it "hides" new constants, optimal solutions which are unique up to constant renaming are easily recaptured.

A much simpler example is $\mathsf{GrAtoms}(\mathbf{D})$. It yields optimal solutions only in the case that such solutions are truly unique. For deletions, this equivalence is adequate. In fact, for deletions, $\mathsf{StdUCP}(\mathbf{D}, M_1)$ and $\mathsf{GrAtoms}(\mathbf{D})$ always identify the same optimal solutions.

There are other possibilities which provide different notions of optimality. Let $\mathbf{E}_1$ be the schema which is identical to $\mathbf{E}_0$ of 1.1, save that it includes an additional relation symbol $S'[CD]$, and the inclusion dependency $R[C] \subseteq S[C]$ is replaced with $R[C] \subseteq S[C] \cup S'[C]$. Let M'_{00} be the state of $\mathbf{E}_1$ which is the extension of M_{00} in which the relation of S' is empty. The view $\Pi_{R[AB]} = (R[AB], \pi_{R[AB]})$ is unchanged. Under the update classifier $\mathsf{WFS}(\mathbf{E}_1, \mathsf{ConstSym}(M'_{00}), \exists\wedge+)$, the update request (M'_{00}, N_{01}) (using N_{01} from 1.1) no longer has an optimal solution, since a minimal solution involves adding a tuple either to S or to S' but not to both. However, optimality can be recovered formally via an alternative update classifier. Let Ξ_1 denote the subset of $\mathsf{WFS}(\mathbf{E}_1, \exists+)$ obtained from $\mathsf{WFS}(\mathbf{E}_0, \exists\wedge+)$ by replacing each occurrence of the form $S(\tau_1, \tau_2)$ by $(S(\tau_1, \tau_2) \vee S'(\tau_1, \tau_2))$. Here τ_1 and τ_2 are arbitrary terms (*i.e.*, variables or constants). In effect, the sentences of Ξ_1 cannot distinguish a given tuple in S from an identical one in S'. It is easy to see that Ξ_1 is information monotone (since it is a subset of $\mathsf{WFF}(\mathbf{E}_1, \exists+)$). Furthermore, both of the solutions $M'_{01} = M'_{00} \cup \{R(a_2, b_2, c_2), S(c_2, d_2)\}$ and $M''_{01} = M'_{00} \cup \{R(a_2, b_2, c_2), S'(c_2, d_2)\}$ are optimal under this measure.

By choosing a suitable update classifier, rather broad notions of equivalence are hence achievable, so there is a tradeoff between the generality of the update classifier and how "equivalent" the various optimal solutions really are. In the example sketched above, the solutions are not isomorphic in any reasonable sense. On the other hand, for $\mathsf{StdUCP}(\mathbf{E}_0, M_{00})$, all optimal solutions are naturally isomorphic, a nontrivial result which requires some work to establish; the rest of this section is devoted to that task.

3.7 Constant endomorphisms An *endomorphism* on $\mathcal{D}$ is a function $h : \mathsf{Const}(\mathcal{D}) \to \mathsf{Const}(\mathcal{D})$ which preserves attribute types, in the precise sense that for each $A \in \mathcal{A}_\mathcal{D}$ and each $a \in \mathsf{Const}_\mathcal{D}(A)$, $h(a) \in \mathsf{Const}_\mathcal{D}(A)$. If h is additionally a bijection, then it is called an *automorphism* of $\mathcal{D}$. For $S \subseteq \mathsf{Const}(\mathcal{D})$, call h S-invariant if $h(a) = a$ for all $a \in S$.

Given a database schema $\mathbf{D}$, an endomorphism on $\mathcal{D}$ induces a mapping from $\mathsf{GrAtoms}(\mathbf{D})$ to itself given by sending $t \in \mathsf{GrAtoms}(\mathbf{D})$ to the tuple t' with $t'[\mathsf{RName}] = t[\mathsf{RName}]$ and $t'[A] = t[h(A)]$ for all $A \in \mathsf{Ar}_{\mathbf{t}[\mathsf{RName}]}$. This mapping on atoms will also be represented by h, as will the induced mapping from $\mathsf{DB}(\mathbf{D})$ to itself given by $M \mapsto \{h(t) \mid t \in M\}$.

3.8 Armstrong models in an information-monotone context Let $\Psi \subseteq \mathsf{WFS}(\mathbf{D})$ and let $\Phi \subseteq \Psi$. Informally, an Armstrong model for Φ relative to Ψ is a model of Φ which satisfies only those constraints of Ψ which are implied by Φ. More formally, an *Armstrong model* for Φ relative to Ψ is an $M \in \mathsf{Mod}_{\mathcal{I}}(\Phi)$ with the property that for any $\psi \in \Psi$, if $M \in \mathsf{Mod}_{\mathcal{I}}(\psi)$, then $\mathsf{Mod}_{\mathcal{I}}(\Phi) \subseteq \mathsf{Mod}_{\mathcal{I}}(\psi)$. Armstrong models have been studied extensively for database dependencies; see, for example, [Fag82] and [FV83]. In the current context, it will be shown that if (M_1, M_2) is a $\mathsf{StdUCP}(\mathbf{D}, M_1)$-optimal reflection of the update request (M_1, N_2), then M_2 is a minimal Armstrong model with respect to $\mathsf{StdUCP}(\mathbf{D}, M_1)$. It will furthermore be shown that if (M_1, M_2') is another such optimal reflection, there is an an automorphism h which is constant on $\mathsf{Const}_{\mathcal{D}}(M)$ with $M_2' = h(M_2)$.

3.9 Representation of $\exists\wedge+$-sentences as sets of D-atoms There is an alternative syntactic representation for formulas in $\mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ which will be used in that which follows. Specifically, for $\varphi \in \mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ define $\mathsf{AtRep}(\varphi)$ to be the set of all atoms which occur as conjuncts in φ. For example, if $\varphi = (\exists x_1)(\exists x_2)(\exists x_3)(R(x_1, a) \wedge R(x_1, b) \wedge S(x_2, a) \wedge T(x_2, x_3))$ then $\mathsf{AtRep}(\varphi) = \{R(x_1, a), R(x_1, b), S(x_2, a), T(x_2, x_3)\}$.

This representation is dual to that used in theorem-proving contexts in classical artificial intelligence [GN87, 4.1]. Here the variables are existentially quantified and the atoms are conjuncts of one another; in the AI setting the atoms are disjuncts of one another and the variables are universally quantified.

3.10 Substitutions Let $V = \{v_1, v_2, \ldots, v_n\} \subseteq \mathsf{GenVars}(\mathcal{D})$. A *(constant) substitution* for V (in $\mathcal{D}$) is a function $s : V \to \mathsf{Const}(\mathcal{D})$. If $s(x_i) = a_i$ for $i \in \{1, 2, \ldots, n\}$, following (somewhat) standard notation this substitution is often written $\{a_1/x_1, a_2/x_2, \ldots, a_n/x_n\}$ (although some authors [GN87, 4.2] write $\{x_1/a_1, x_2/a_2, \ldots, x_n/a_n\}$ instead).

Let $\varphi \in \mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ with $\mathsf{Vars}(\varphi) \subseteq V$. Call s *correctly typed* for φ if for each $t \in \mathsf{AtRep}(\varphi)$ and each $A \in \mathsf{Ar}_{\mathbf{D}}(t[\mathsf{RName}])$, if $t[A] \in \mathsf{Vars}(\mathbf{D})$ then $s(t[A]) \in \mathsf{Const}_{\mathcal{D}}(A)$. Define $\mathsf{Subst}(\varphi, s)$ to be the set of ground atoms obtained by substituting $s(x_i)$ for x_i in $\mathsf{AtRep}(\varphi)$. For example, with $s = \{a_1/x_1, a_2/x_2, a_3/x_3\}$ and $\mathsf{AtRep}(\varphi) = \{R(x_1, a), R(x_1, b), S(x_2, a), T(x_2, x_3)\}$, $\mathsf{Subst}(\varphi, s) = \{R(a_1, a), R(a_1, b), S(a_2, a), T(a_2, a_3)\}$.

Now let $\Phi \subseteq \mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ be a finite set. A *substitution set* for Φ is a Φ-indexed set $S = \{s_\varphi \mid \varphi \in \Phi\}$ of substitutions, with s_φ a substitution for $\mathsf{Vars}(\varphi)$. S is *free for* Φ if each s_φ is correctly typed for φ, injective, and, furthermore, for any distinct $\varphi_1, \varphi_2 \in \Phi$, $s_{\varphi_1}(\mathsf{Vars}(\varphi_1)) \cap s_{\varphi_2}(\mathsf{Vars}(\varphi_2)) = \emptyset$.

For a free substitution S, the *canonical model* defined by (Φ, S) is obtained by applying the substitution s_φ to φ for each $\varphi \in \Phi$. Formally, $\mathsf{CanMod}\langle\Phi, S\rangle = \bigcup\{\mathsf{Subst}(\varphi, s_\varphi) \mid \varphi \in \Phi\}$.

For an illustrative example, let $\varphi_1 = (\exists x_1)(\exists x_2)(\exists x_3)(R(x_1, x_2) \wedge R(x_2, x_3))$, $\varphi_2 = (\exists x_1)(\exists x_2)(\exists x_3)(R(a_1, x_1) \wedge S(x_1, a_2) \wedge T(x_2, x_3) \wedge T(x_2, a_3))$, $\varphi_3 = R(a_1, a_2)$, and $\varphi_4 = S(a_2, a_3)$, with the a_i's all distinct constants, and let $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$. Put $s_{\varphi_1} = \{b_{11}/x_1, b_{12}/x_2, b_{13}/x_3\}$ $s_{\varphi_2} = \{b_{21}/x_1, b_{22}/x_2, b_{23}/x_3\}$, and $s_{\varphi_3} = s_{\varphi_4} = \emptyset$, with all of the b_{ij}'s distinct constants. Then $S = \{s_{\varphi_1}, s_{\varphi_2}, s_{\varphi_3}, s_{\varphi_4}\}$ is free for Φ, with $\mathsf{Subst}(\varphi_1, s_{\varphi_1}) = \{R(b_{11}, b_{12}), S(b_{12}, b_{13})\}$, $\mathsf{Subst}(\varphi_2, s_{\varphi_2}) = \{R(a_1, b_{21}), S(b_{21}, a_2), T(b_{22}, b_{23}), T(b_{22}, a_3)\}$, $\mathsf{Subst}(\varphi_3, s_{\varphi_3}) = \{R(a_1, a_2)\}$, and $\mathsf{Subst}(\varphi_4, s_{\varphi_4}) = \{S(a_2, a_3)\}$. Unfortunately, while $\mathsf{CanMod}\langle\Phi, S\rangle = \{\mathsf{Subst}(\varphi_i, s_{\varphi_i}) \mid 1 \leq i \leq 4\}$ is an Armstrong model for $\{\varphi_i \mid 1 \leq i \leq 4\}$with respect to $\mathsf{WFF}(\mathbf{D}, \exists\wedge+)$, it is not not minimal. The problem is that there is redundancy within Φ, which results in redundancy in

the canonical model. For example, it is easy to see that φ_1 is a logical consequence of φ_2, and so $\mathsf{Subst}(\varphi_1, s_{\varphi_1})$ can be removed from $\mathsf{CanMod}\langle\Phi, S\rangle$ entirely, with the result still an Armstrong model. While this problem could be resolved by choosing the substitutions more cleverly, it is more straightforward to normalize the the set of sentences before applying the construction of the canonical model, as developed next.

3.11 Reduction steps To construct a minimal Armstrong model from a set $\Phi \subseteq \mathsf{WFS}(\mathbf{D}, \exists\wedge+)$, it is first necessary to normalize Φ by applying three simple reduction rules, defined as follows.

Decomposition: For $\varphi \in \Phi$, if $\{X_1, X_2\}$ partitions $\mathsf{AtRep}(\varphi)$ into disjoint sets, and $\mathsf{Mod}_{\mathcal{I}}(\mathsf{AtRep}^{-1}(X_1)) \cap \mathsf{Mod}_{\mathcal{I}}(\mathsf{AtRep}^{-1}(X_2)) = \mathsf{Mod}_{\mathcal{I}}(\varphi)$, then remove φ from Φ and add both $\mathsf{AtRep}^{-1}(X_1)$ and $\mathsf{AtRep}^{-1}(X_2)$.

Collapsing: For $\varphi \in \Phi$, if $\{X_1, X_2\}$ partitions $\mathsf{AtRep}(\varphi)$ into disjoint sets, and $\mathsf{Mod}_{\mathcal{I}}(\mathsf{AtRep}^{-1}(X_1)) \subseteq \mathsf{Mod}_{\mathcal{I}}(\mathsf{AtRep}^{-1}(X_2))$, then remove φ from Φ and add $\mathsf{AtRep}^{-1}(X_1)$.

Minimization: If $\mathsf{Mod}_{\mathcal{I}}(\Phi \setminus \{\varphi\}) = \mathsf{Mod}_{\mathcal{I}}(\Phi)$, then remove φ from Φ.

It is clear that each of these steps preserves $\mathsf{Mod}_{\mathcal{I}}(\Phi)$, and that they may only be applied a finite number of times before none is applicable. Call Φ *reduced* if none of these steps is applicable.

A simple example will help illustrate how these rules work. Let Φ be as in 3.10 above. Using the decomposition rule, $\mathsf{AtRep}(\varphi_2) = \{R(a_1, x_1), R(x_1, a_2), T(x_2, x_3), T(x_2, a_3)\}$ may be replaced with $\{R(a_1, x_1), S(x_1, a_2)\}$ and $\{T(x_2, x_3), T(x_2, a_3)\}$, since these two sets have no variables in common. Next, $\{T(x_2, x_3), T(x_2, a_3)\}$ may be replaced with $\{T(x_2, a_3)\}$ using the collapsing rule. Finally, φ_1 may be removed using the minimization rule, since it is a consequence of $\varphi_3 \wedge \varphi_4$. The final reduced version of Φ is thus $\{(\exists x_1)(R(a_1, x_1) \wedge S(x_1, a_2)), (\exists x_2)(T(x_2, a_3)), R(a_1, a_2), S(a_2, a_3)\}$. Note that $(\exists x_1)(R(a_1, x_1) \wedge S(x_1, a_2))$ is not a consequence of $R(a_1, a_2)$ and $S(a_2, a_3)$, and so it cannot be removed by this procedure. A minimal Armstrong model is obtained by substituting a distinct new constant for each variable: $\{R(a_1, b_1), R(b_1, a_2), T(b_2, a_3), R(a_1, a_2), S(a_2, a_3)\}$. Furthermore, this model is obtained from the one of 3.10 above via the endomorphism which maps $b_{11} \mapsto a_1$, $b_{12} \mapsto a_2$, $b_{13} \mapsto a_3$, $b_{21} \mapsto b_1$, $b_{22} \mapsto b_2$, $b_{23} \mapsto a_3$, and is the identity on everything else. To establish this result in a completely formal fashion requires a bit of work, and is presented below.

3.12 Theorem — Characterization of minimal Armstrong models *Let $\Phi \subseteq \mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ be a finite set of constraints, and assume furthermore that Φ is reduced in the sense of 3.11 above. Let S be a substitution set which is free for Φ. Then the following hold.*

(a) $\mathsf{CanMod}\langle\Phi, S\rangle$ *is a minimal Armstrong model for* Φ *relative to* $\mathsf{WFS}(\mathbf{D}, \mathsf{ConstSym}(\Phi), \exists\wedge+)$.

(b) *For any* $M \in \mathsf{Mod}_{\mathcal{I}}(\Phi)$, *there is a* $\mathsf{ConstSym}(\Phi)$*-invariant endomorphism* h *on* $\mathcal{D}$ *with* $h(\mathsf{CanMod}\langle\Phi, S\rangle) \subseteq M$.

(c) *If* M *is any other minimal Armstrong model for* Φ *relative to* $\mathsf{WFS}(\mathbf{D}, \mathsf{ConstSym}(\Phi), \exists\wedge+)$, *then there is a* $\mathsf{ConstSym}(\Phi)$*-invariant automorphism* h *on* $\mathcal{D}$ *with* $h(\mathsf{CanMod}\langle\Phi, S\rangle) = M$.

PROOF: It is immediate that $\mathsf{CanMod}\langle\Phi, S\rangle$ is a model of Φ. It is furthermore easy to see that it is minimal; if any tuple is deleted, the $\varphi \in \Phi$ associated with the tuple in $\mathsf{CanMod}\langle\Phi, S\rangle$ is no longer satisfied, since Φ is assumed to be minimized, as defined in 3.11 above. To show that it is an Armstrong model, let $\psi \in \mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ for which $\mathsf{Mod}_{\mathcal{I}}(\Phi) \subseteq \mathsf{Mod}_{\mathcal{I}}(\psi)$ does not hold, and let S' be a substitution set, free for $\Phi \cup \{\psi\}$, which is built from S by adding a substitution associated with ψ. Let the resulting set of constraints by the reduction steps of 3.11 from $\Phi \cup \{\psi\}$ be denoted by Φ'. For the reduction steps of 3.11, it suffices to note that ψ cannot be removed by minimization. Hence $\mathsf{CanMod}\langle\Phi', S'\rangle \subseteq \mathsf{CanMod}\langle\Phi, S\rangle$ cannot hold, and so $\mathsf{CanMod}\langle\Phi, S\rangle$ cannot be a model of ψ, whence $\mathsf{CanMod}\langle\Phi, S\rangle$ is an Armstrong model of Φ.

To establish (b), let $M \in \mathsf{Mod}_{\mathcal{I}}(\Phi)$, and for each $\varphi \in \Phi$, let M_φ be a minimal subset of M with $M_\varphi \in \mathsf{Mod}_{\mathcal{I}}(\varphi)$. Let V_φ denote the set of variables of $s_\varphi \in S$. It is easy to see that there must be a substitution s'' with $\mathsf{Vars}(s'') = V_\varphi$ and $\mathsf{Subst}(\varphi, s'') = M_\varphi$. Indeed, there is trivially a substitution with $\mathsf{Subst}(\varphi, s'') \subseteq M_\varphi$, but if the subset inclusion were proper, M_φ would not be minimal.

Now define $h : s_\varphi(V_\varphi) \to s''(V_\varphi)$ by $a \mapsto s''(s_\varphi^{-1}(a))$. Since s_φ is injective, h is well defined. Since $s_{\varphi_1}(\mathsf{Vars}(\varphi_1)) \cap s_{\varphi_2}(\mathsf{Vars}(\varphi_2)) = \emptyset$ for distinct $\varphi_1, \varphi_2 \in \Phi$, there are no conflicts in this definition of h. Finally, extend h to be the identity on all $a \in \mathsf{Const}(\mathcal{D})$ which are not covered by the above definition. The result is a endomorphism on $\mathcal{D}$ which satisfies $h(\mathsf{CanMod}\langle\Phi, S\rangle) \subseteq M$.

To show (c), let M be any other minimal Armstrong model for Φ relative to $\mathsf{WFS}(\mathbf{D}, \mathsf{ConstSym}(\Phi))\exists\wedge+$. In the above construction for the proof of (b), the resulting h must be surjective (else M would not be minimal), and it must be injective (since there must also be an endomorphism in the opposite direction, and both $\mathsf{CanMod}\langle\Phi, S\rangle$ and M are finite, by assumption). Hence, h is an automorphism. □

The desired result, that any two optimal realizations are isomorphic up to a renaming via an automorphism, follows directly as a corollary.

3.13 Corollary — Optimal updates are unique up to constant automorphism *Let (M_1, N_2) be an update request from Γ to $\mathbf{D}$, and let (M_1, M_2) and (M_1, M_2') be $\langle\mathsf{StdUCP}(\mathbf{D}, M_1), \mathsf{UpdRealiz}\langle M_1, N_2, \Gamma\rangle\rangle$-optimal realizations of (M_1, N_2). Then there is a $\mathsf{ConstSym}^+_{\mathcal{D}}(M)$-invariant automorphism h on $\mathcal{D}$ with $M_2' = h(M_2)$.* □

In some ways, the construction given above is similar to the construction of the universal solutions of [FKMP05, Def. 2.4], in that both are based upon similar notions of endomorphism (there termed *homomorphism*). However, those universal solutions are not required to be minimal. On the other hand, they are not limited to positive sentences, but rather apply to XEIDs, as developed in the next section.

4. Optimal Insertion in the Context of XEIDs

In this section, it is shown that in the context of database constraints which are extended embedded implicational dependencies (XEIDs), and views which are of class $\exists\wedge+$, all admissible realizations of an insertion request are optimal. In other words, there cannot be non-isomorphic minimal realizations of an update request which is an insertion. To establish this isomorphism, it is necessary to rule out the kind of non-isomorphic alternatives which are illustrated in

3.6. The logical formulation which formalizes this idea is splitting of disjunctions. Informally, disjunction splitting [Fag82, Thm. 3.1(c)] stipulates that nondeterminism in logical implication cannot occur. If a set Ψ_1 of sentences implies the disjunction of all sentences in Ψ_2, then it must in fact imply some sentence in Ψ_2. Since Ψ_2 may be infinite, the notion of disjunction must be formulated carefully.

4.1 Notational convention Throughout this section, unless stated explicitly to the contrary, take Σ to be an update classifier for $\mathbf{D}$.

4.2 Splitting of disjunctions over finite databases The family $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ *splits disjunctions over finite databases* if whenever $\Psi_1, \Psi_2 \subseteq \Phi$ with Ψ_2 nonempty have the property that $\mathsf{RNeMod}_{\mathcal{I}}(\Psi_1) \subseteq \bigcup\{\mathsf{RNeMod}_{\mathcal{I}}(\psi') \mid \psi' \in \Psi_2\}$, then there is a $\psi \in \Psi_2$ with $\mathsf{RNeMod}_{\mathcal{I}}(\Psi_1) \subseteq \mathsf{RNeMod}_{\mathcal{I}}(\psi)$. The limitation to relationwise-nonempty databases is a technical one which will ultimately be necessary. The definition can, of course, be made without this restriction, and even 4.3 below is true without it, but the critical result 4.9 would fail. Since requiring databases to be relationwise nonempty is not much of restriction, it is easiest to require it throughout.

Define $\mathsf{SDConstr}\langle \mathbf{D}, \Gamma, \Sigma\rangle = \mathsf{Constr}(\mathbf{D}) \cup \mathsf{GrAtoms}(\mathbf{D}) \cup \{\mathsf{Subst}\langle \gamma, t\rangle \mid t \in \mathsf{GrAtoms}(\mathbf{V})\} \cup \Sigma$. Basically, $\mathsf{SDConstr}\langle \mathbf{D}, \Gamma, \Sigma\rangle$ is the set of all sentences which can arise in the construction of updates on $\mathbf{D}$ induced via updates on the view Γ. $\mathsf{Constr}(\mathbf{D})$ is (a basis for) the set of all constraints on $\mathbf{D}$, $\mathsf{GrAtoms}(\mathbf{D})$ is the set of all ground atoms which can occur in a database of $\mathbf{D}$, $\{\mathsf{Subst}\langle \gamma, t\rangle \mid t \in \mathsf{GrAtoms}(\mathbf{V})\}$ is the set of all sentences on $\mathbf{D}$ which can arise by reflecting an atom of the view Γ to the main schema, and Σ is the set of all sentences which are used in measuring information content. If all of these together split disjunctions, the constructions will work. Formally, say that the triple $(\mathbf{D}, \Gamma, \Sigma)$ *supports disjunction splitting over finite databases* if $\mathsf{SDConstr}\langle \mathbf{D}, \Gamma, \Sigma\rangle$ splits disjunctions over finite databases.

4.3 Theorem — Disjunction splitting implies that admissible insertions are optimal *Assume that* $(\mathbf{D}, \Gamma, \Sigma)$ *supports disjunction splitting over finite databases, and let* (M_1, N_2) *be an insertion request from* Γ *to* $\mathbf{D}$ *with the property that* M_1 *is relationwise nonempty. Then all* $\langle \Sigma, \uparrow\rangle$*-admissible realizations of* (M_1, N_2) *are* $\langle \Sigma, \uparrow\rangle$*-optimal.*

PROOF: First of all, observe that $\Psi_1 = \mathsf{Constr}(\mathbf{D}) \cup \{\mathsf{Subst}\langle \gamma, t\rangle \mid t \in N_2\} \cup M_1$ is precisely the set of constraints which the updated database of $\mathbf{D}$ must satisfy; $(M_1, M_2) \in \mathsf{InsRealiz}\langle M_1, N_2, \Gamma\rangle$ iff $M_2 \in \mathsf{Mod}_{\mathcal{I}}(\Psi_1)$. Furthermore, since $\Psi_1 \subseteq \mathsf{SDConstr}\langle \mathbf{D}, \Gamma, \Sigma\rangle$, it splits disjunctions over finite databases.

Now, let S denote the set of all $M_2 \in \mathsf{LDB}(\mathbf{D})$ for which (M_1, M_2) is a $\langle \Sigma, \uparrow\rangle$-admissible realization of (M_1, N_2), and assume that S is nonempty. Let Ψ_2 denote the set of all $\psi \in \Sigma$ with the property that $M_2 \in \mathsf{Mod}_{\mathcal{I}}(\psi)$ for some, but not all, $M_2 \in S$. If $\Psi_2 = \emptyset$, then all members of S are Σ-equivalent, and so all are least with respect to Σ and hence $\langle \Sigma, \uparrow\rangle$-optimal. If $\Psi_2 \neq \emptyset$, then for each $M_2 \in S$, there must be some $\psi \in \Psi_2$ with the property that $M_2 \in \mathsf{Mod}_{\mathcal{I}}(\psi)$. Otherwise, $\mathsf{Info}\langle M_2, \Sigma\rangle \subsetneq \mathsf{Info}\langle M_2', \Sigma\rangle$ for all $M_2' \in S \cap \mathsf{Mod}_{\mathcal{I}}(\psi)$, which would contradict the Σ-admissibility of any such M_2'. Thus $\mathsf{RNeMod}_{\mathcal{I}}(\Psi_1) \subseteq \bigcup\{\mathsf{RNeMod}_{\mathcal{I}}(\psi') \mid \psi' \in \Psi_2\}$. Since M_1 is relationwise nonempty, so too is each $M_2 \in S$. Now, using the fact that Ψ_1 splits disjunctions, there is some $\psi \in \Psi_2$ with the property that $\mathsf{RNeMod}_{\mathcal{I}}(\Psi_1) \subseteq \mathsf{RNeMod}_{\mathcal{I}}(\psi)$;

i.e., that $M_2 \in \mathsf{Mod}_{\mathcal{I}}(\psi)$ for all $M_2 \in S$. This is a contradiction, and so $\Psi_2 = \emptyset$. Thus all $\langle \Sigma, \uparrow \rangle$-admissible realizations of (M_1, N_2) are are $\langle \Sigma, \uparrow \rangle$-optimal. $\square$

4.4 Extended tuple databases The results which follow use heavily the framework developed by Fagin in [Fag82]. It is necessary in particular to be able to construct infinite products of databases. This leads to two complications. First of all, the databases of this paper are finite, while such products may be infinite. Second, the databases of this paper here have a fixed bijective correspondence between the domain of the interpretation and constant symbols which cannot be preserved completely under products. Fortunately, such products are not really used as databases; rather they are just artefacts which arise in the proof to show that a certain context supports the splitting of disjunctions. The solution is to embed the $\mathbf{D}$-tuples of this paper into a larger set, called the extended $\mathbf{D}$-tuples, and to carry out the infinite-product constructions on databases of these extended tuples. Since every tuple database in the sense of this paper is also an extended database, the results will follow.

Formally, an *extended tuple database* M over $\mathbf{D}$ consists of the following:

(xtdb-i) A set $\mathsf{Dom}(M)$, called the *domain* of M.

(xtdb-ii) An injective function $\iota_M : \mathsf{Dom}_{\mathcal{I}} \to \mathsf{Dom}(M)$.

(xtdb-iii) A (not necessarily finite) set $\mathsf{XTuples}(M)$ of extended $\mathbf{D}$-tuples over $(\mathsf{Dom}(M), \iota_M)$.

For $R \in \mathsf{Rels}(\mathbf{D})$, an *extended R-tuple* t over $(\mathsf{Dom}(M), \iota_M)$ is a function $t : \{\mathsf{RName}\} \cup \mathsf{Ar}_{\mathbf{D}}(R) \to \mathsf{Dom}(M) \cup \mathsf{Rels}(\mathbf{D})$ with the property that $t[\mathsf{RName}] = R$ and, for all $A \in \mathcal{A}_{\mathcal{D}}$, if $t[A] \in \iota_M(\mathsf{Dom}_{\mathcal{I}})$, then $\iota_M^{-1}(t[A]) \in \mathsf{Dom}_{\mathcal{I}}(A)$. An *extended $\mathbf{D}$-tuple* over $(\mathsf{Dom}(M), \iota_M)$ is an extended R-tuple over that same pair for some $R \in \mathsf{Rels}(\mathbf{D})$. $\mathsf{XTuples}(M)$ denotes $\bigcup\{\mathsf{XTuples}(R, M) \mid R \in \mathsf{Rels}(\mathbf{D})\}$. The collection of all extended tuple databases over $\mathbf{D}$ is denoted $\mathsf{XTDB}(\mathbf{D})$, with $\mathsf{RNeXTDB}(\mathbf{D})$ denoting the subcollection consisting of all relationwise-nonempty members (obvious definition). As a slight abuse of notation, $t \in M$ will be used as shorthand for $t \in \mathsf{XTuples}(M)$.

Note that every $M \in \mathsf{TDB}(\mathbf{D})$ may be regarded as an extended tuple database by setting $\mathsf{Dom}(M) = \mathsf{Dom}_{\mathcal{I}}$ and taking ι_M to be the identity function. In an extended $\mathbf{D}$-tuple, domain elements which are not in $\iota_M(\mathsf{Dom}_{\mathcal{I}})$ are not associated with any constant symbol.

For $\varphi \in \mathsf{WFS}(\mathbf{D})$, define $\mathsf{XMod}_{\mathcal{I}}(\varphi)$ to be the set of $M \in \mathsf{XTDB}(\mathbf{D})$ which interpret the constant symbols according to $\mathsf{IntFn}_{\mathcal{I}}$, and which are models (in the usual logical sense) of both φ and $\mathsf{UNA}(\mathcal{D})$. For $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$, $\mathsf{XMod}_{\mathcal{I}}(\Phi) = \bigcap\{\mathsf{XMod}_{\mathcal{I}}(\varphi) \mid \varphi \in \Phi\}$. The relationwise-nonempty versions, $\mathsf{RNeXMod}_{\mathcal{I}}(\varphi)$ and $\mathsf{RNeXMod}_{\mathcal{I}}(\Phi)$, are defined analogously. Note that $\mathsf{Mod}_{\mathcal{I}}(\varphi) \subseteq \mathsf{XMod}_{\mathcal{I}}(\varphi)$ and $\mathsf{RNeMod}_{\mathcal{I}}(\varphi) \subseteq \mathsf{RNeXMod}_{\mathcal{I}}(\varphi)$ under these definitions; *i.e.*, ordinary models are extended models.

4.5 Products of extended tuple databases Let $P = \{M_j \mid j \in J\}$ be an indexed set of nonempty extended tuple databases over $\mathbf{D}$. The $\mathcal{D}$-product of P, denoted $\otimes^{\mathcal{D}}(P)$, is the extended tuple database defined as follows:

(i) $\mathsf{Dom}(\otimes^{\mathcal{D}}(P)) = \prod_{j \in J} \mathsf{Dom}(M_j)$.

(ii) $\iota_{\otimes^{\mathcal{D}}(P)} : x \mapsto \langle \iota_{M_j}(x) \rangle_{j \in J}$ (the J-tuple whose j^{th} entry is $\iota_{M_j}(x)$).

(iii) $\mathsf{XTuples}(R, \otimes^{\mathcal{D}}(P)) = \{\otimes\langle t_j\rangle_{j\in J} \mid t_j \in \mathsf{XTuples}(R, M_j)\}$.

In the above, $t' = \otimes\langle t_j\rangle_{j\in J}$ is the extended R-tuple with $t'[A] = \langle t_j[A]\rangle_{j\in J}$ for each $A \in \mathsf{Ar}_{\mathbf{D}}(R)$.

Call $\otimes^{\mathcal{D}}(P)$ *lossless* if each M_j can be recovered from it. Note that $\otimes^{\mathcal{D}}(P)$ is lossless if each $M_j \in P$ is in $\mathsf{RNeXTDB}(\mathbf{D})$; however, given $R \in \mathsf{Rels}(\mathbf{D})$, if some M_j contains no R-tuples, then the entire product will contain no R-tuples. Since it is essential to be able to recover P from $\otimes^{\mathcal{D}} D$, the condition that each $M_j \in P$ be relationwise nonempty will be enforced.

4.6 Splitting of disjunctions over extended databases 4.2 can be extended in the obvious fashion to extended databases. Specifically, the family $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ *splits disjunctions over extended databases* if whenever $\Psi_1 \subseteq \Phi$ and $\Psi_2 \subseteq \Phi$ with Ψ_2 nonempty have the property that $\mathsf{RNeXMod}_{\mathcal{I}}(\Psi_1) \subseteq \bigcup\{\mathsf{RNeXMod}_{\mathcal{I}}(\psi') \mid \psi' \in \Psi_2\}$, then there is a $\psi \in \Psi_2$ with $\mathsf{RNeXMod}_{\mathcal{I}}(\Psi_1) \subseteq \mathsf{RNeXMod}_{\mathcal{I}}(\psi)$. Similarly, the triple $\mathsf{SDConstr}\langle\mathbf{D}, \Gamma, \Sigma\rangle$ *supports disjunction splitting over extended databases* if $\mathsf{SDConstr}\langle\mathbf{D}, \Gamma, \Sigma\rangle$ splits disjunctions over extended databases.

Because ordinary tuples may be interpreted as extended tuples, splitting of disjunctions over extended databases trivially implies splitting of disjunction over ordinary finite databases. Due to its importance, this fact is recorded formally.

4.7 Observation *If the family $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ splits disjunctions over extended databases, then it splits disjunctions over finite databases as well. In particular, if the triple $\mathsf{SDConstr}\langle\mathbf{D}, \Gamma, \Sigma\rangle$ supports disjunction splitting over extended databases, then it supports disjunction splitting over finite databases.* □

4.8 Faithful sentences Informally, a sentence $\varphi \in \mathsf{WFS}(\mathbf{D})$ is faithful [Fag82] if it is preserved under the formation of products and under the projection of factors from products. Formally, $\varphi \in \mathsf{WFS}(\mathbf{D})$ is said to be *faithful* if whenever $P = \{M_j \mid j \in J\} \subseteq \mathsf{RNeXTDB}(\mathbf{D})$ is a nonempty (indexed) set, $\otimes^{\mathcal{D}}(P) \in \mathsf{XMod}_{\mathcal{I}}(\varphi)$ iff $M_j \in \mathsf{XMod}_{\mathcal{I}}(\varphi)$ for each $M_j \in P$. The family $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$ is *faithful* precisely in the case that each $\varphi \in \Phi$ is.

4.9 Theorem — Faithful ≡ disjunction splitting *Let $\Phi \subseteq \mathsf{WFS}(\mathbf{D})$. Then Φ is faithful iff it splits disjunctions over extended databases.*

PROOF: See [Fag82, Thm. 3.1]. □

4.10 XEIDs The *extended embedded implicational dependencies (XEIDs)* form a very general class which includes most types of dependencies which have been studied, including functional dependencies, multivalued dependencies, (embedded) join dependencies, and inclusion dependencies. Formally, an *XEID* [Fag82, Sec. 7] is a sentence in $\mathsf{WFS}(\mathbf{D})$ of the form

$$(\forall x_1)(\forall x_2)\ldots(\forall x_n)((A_1 \wedge A_2 \wedge \ldots \wedge A_n) \Rightarrow (\exists y_1)(\exists y_2)\ldots(\exists y_r)(B_1 \wedge B_2 \wedge \ldots \wedge B_s))$$

such that each A_i is a relational atom for the same relation, *i.e.*, the left-hand side is unirelational, each B_i is a relational atom or an equality, each x_i occurs in some A_j, the left-hand side is typed in the sense that no variable is used for more than one attribute. In the original definition of Fagin, it is also required that $n \geq 1$. However, this is an inessential constraint which may

easily be dropped, as long as at least one of the $B_i's$ is a relational atom (and not an equality). Indeed, let $\varphi = (\exists y_1)(\exists y_2)\ldots(\exists y_r)(B_1 \wedge B_2 \wedge \ldots \wedge B_s)) \in \mathsf{WFS}(\mathbf{D}, \exists\wedge+)$, with B_i, say, a relational atom for relation symbol R. Let $A = R(x_1, x_2, .., x_n)$ be a relational atom for R with variables as arguments. Then $\varphi' = (\forall x_1)(\forall x_2)\ldots(\forall x_n)(A \Rightarrow (\exists y_1)(\exists y_2)\ldots(\exists y_r)(A \wedge B_1 \wedge B_2 \wedge \ldots \wedge B_s)) \in \mathsf{WFS}(\mathbf{D})$ is equivalent to φ, but with $n \geq 1$. Thus, without loss of generality, in this paper sentences in $\mathsf{WFS}(\mathbf{D}, \exists\wedge+)$ will also be regarded as XEIDs.

The set of all XEIDs on $\mathbf{D}$ is denoted $\mathsf{XEID}(\mathbf{D})$, while those XEIDs involving only the constant symbols in $S \subseteq \mathsf{Const}(\mathcal{D})$ is denoted $\mathsf{XEID}(\mathbf{D}, S)$.

The reason that XEIDS are of interest here is the following.

4.11 Proposition *Every $\Phi \subseteq \mathsf{XEID}(\mathbf{D})$ is faithful.*

PROOF: This is essentially [Fag82, Thm. 7.2]. The only complication is the constant symbols, which are not part of the framework of [Fag82], so the integrity of $\mathsf{UNA}(\mathcal{D})$ (not a set of XEIDs) must be verified. To this end, simply note that a domain value in an extended model is associated with constant a iff *each* of its projections is the domain value $\mathsf{IntFn}_I(a)$, so $\mathsf{UNA}(\mathcal{D})$ is enforced by construction. □

4.12 Lemma — XEIDs support disjunction splitting *Let $\mathsf{Constr}(\mathbf{D}) \subseteq \mathsf{XEID}(\mathbf{D})$, Γ a view of class $\exists\wedge+$, and Σ an update classification pair for $\mathbf{D}$ with $\Sigma \subseteq \mathsf{XEID}(\mathbf{D})$. Then $\mathsf{SDConstr}\langle\mathbf{D}, \Gamma, \Sigma\rangle$ supports disjunction splitting over extended databases.*

PROOF: By construction $\mathsf{SDConstr}\langle\mathbf{D}, \Gamma, \Sigma\rangle \subseteq \mathsf{XEID}(\mathbf{D})$, and so the result follows from 4.9 and 4.11. □

Finally, the main result on the existence of optimal reflections may be established.

4.13 Theorem — XEIDs imply optimal insertions *Let $\mathsf{Constr}(\mathbf{D}) \subseteq \mathsf{XEID}(\mathbf{D})$, Γ a view of class $\exists\wedge+$, (M_1, N_2) an insertion request from Γ to $\mathbf{D}$ with $M \in \mathsf{RNeLDB}(\mathbf{D})$, and Σ an update classification pair for $\mathbf{D}$ with $\Sigma \subseteq \mathsf{XEID}(\mathbf{D})$. Then every $\langle\Sigma, \uparrow\rangle$-minimal realization of (M_1, N_2) is $\langle\Sigma, \uparrow\rangle$-optimal.*

PROOF: Combine 4.3, 4.12, and 4.7. □

4.14 Dependencies which guarantee minimal realizations The above result states that whenever an admissible realization exists, it must be optimal. However, it says nothing about existence, and, indeed it is possible to construct views for which no Σ-admissible update exists. For example, let the schema $\mathbf{E}_2$ have three relational symbols $R[A]$, $S[AB]$, and $T[AB]$ with the inclusion dependencies $R[A] \subseteq S[A]$, $S[A] \subseteq T[A]$, and $T[B] \subseteq S[B]$. Let $M_1 = \{R(a_0), S(a_0, b_0), T(a_0, b_0)\}$. Consider the view $\Pi_{R[A]} = (R[A], \pi_{R[A]})$, which preserves R but discards S and T. Let the current state of this view be $N_1 = \{R(a_0)\}$; consider updating it to $N_2 = N_1 \cup \{R(a_1)\}$. For $\Sigma = \mathsf{StdUCP}(\mathbf{E}_2, M_1)$, there is no Σ-admissible realization of (M_1, N_2). Indeed, a tuple of the form $S(a_1, b_1)$ must be inserted, and this then implies that one of the form $T(a_2, b_1)$ must be inserted as well, which in turn implies that one of the form $S(a_2, b_3)$ must be inserted, and so forth. It is easy to see that if this sequence is terminated by forcing an equality (say, by replacing b_3 with b_2), then the resulting insertion is not Σ-admissible. In other words, relative to $\mathsf{WFS}(\mathbf{D}, \mathsf{ConstSym}^{+}_{\mathcal{D}}(M_1), \exists\wedge+)$, there are no admissible solutions. In

the vernacular of traditional dependency theory, this is a situation in which the chase inference process does not terminate with a finite solution [FKMP05, Def. 3.2]. To ensure termination, attention may be restricted to the subclass of XEIDs consisting of the *weakly acyclic tuple generating dependencies (TGDs)* together with the *equality generating dependencies (EGDs)*. See [FKMP05, Thm. 3.9] for details.

5. Conclusions and Further Directions

A strategy for the optimal reflection of view updates has been developed, based upon the concept of least information change. It has been shown in particular that optimal insertions are supported in a reasonable fashion — they are unique up to a renaming of the newly-inserted constants. Nonetheless, a number of issues remain for future investigation. Among the most important are the following.

Optimization of tuple modification Although the general formulation applies to all types of updates, the results focus almost entirely upon insertions. Due to space limitation, deletions have not been considered in this paper; however, since deletions introduce no new constants or tuples, their analysis is relatively unremarkable within this context. Modification of single tuples ("updates" in SQL), on the other hand, are of fundamental importance. With the standard update classification pair of 3.6, only very special cases admit optimal solutions. The difficulty arises from the fact that the framework, which is based entirely upon information content, cannot distinguish between the process of modifying a tuple and that of deleting it and then inserting a new one. Consequently, both appear as admissible updates, but neither is optimal relative to the other. Further work must therefore look for a way to recapture the distinction between tuple modification and a delete-insert pair.

Application to database components This investigation began as an effort to understand better how updates are propagated between database components, as forwarded in [Heg07, Sec. 4], but then took on a life of its own as it was discovered that the component-based problems were in turn dependent upon more fundamental issues. Nevertheless, it is important to return to the roots of this investigation — database components. This includes not only the purely autonomous case, as sketched in [Heg07, Sec. 4], but also the situation in which users cooperate to achieve a suitable reflection, as introduced in [HS07]

Relationship to work in logic programming The problem of view update has also been studied extensively in the context of deductive databases. Often, only tuple minimality is considered as an admissibility criterion, and the focus then becomes one of identifying efficient algorithms for identifying all such admissible updates [BM04]. However, some recent work has introduced the idea of using active constraints to establish a preference order on admissible updates [GSTZ03]. Thus, rather than employing a preference based upon information content, one based upon explicit rules is employed. The relationship between such approaches and that of this paper warrants further investigation. Also, there has been a substantial body of work on updates to disjunctive deductive databases [FGM96], in which the extensional database itself consists of

a collection of alternatives. The approach of minimizing information change in the disjunctive context deserves further attention as well.

Acknowledgment Much of this research was carried out while the author was a visitor at the Information Systems Engineering Group at the University of Kiel. He is indebted to Bernhard Thalheim and the members of the group for the invitation and for the many discussions which led to this work. Also, the anonymous reviewers made numerous suggestions which (hopefully) have led to a more readable presentation.

References

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu, *Foundations of Databases*, Addison-Wesley, 1995.

[BS81] François Bancilhon and Nicolas Spyratos, "Update semantics of relational views," *ACM Trans. Database Systems*, **6**(1981), pp. 557–575.

[BM04] Andreas Behrend and Rainer Manthey, "Update propagation in deductive databases using soft stratification," in: Georg Gottlob, András A. Benczúr, and János Demetrovics, eds., *Advances in Databases and Information Systems, 8th East European Conference, ADBIS 2004, Budapest, Hungary, September 22-25, 2004, Proceesing*, pp. 22–36, Volume 3255 of Lecture Notes in Computer Science, Springer-Verlag, 2004.

[BL97] Fadila Bentayeb and Dominique Laurent, "Inversion de l'algèbre relationnelle et mises à jour," Technical Report 97-9, Université d'Orléans, LIFO, 1997.

[BL98] Fadila Bentayeb and Dominique Laurent, "View updates translations in relational databases," in: *Proc. DEXA '98, Vienna, Sept. 24-28, 1998*, pp. 322–331, 1998.

[CGT89] Stefano Ceri, Georg Gottlog, and Letizia Tanca, *Logic Programming and Databases*, Springer-Verlag, 1989.

[DB82] Umeshwar Dayal and Philip A. Bernstein, "On the correct translation of update opeartions on relational views," *ACM Trans. Database Systems*, **8**(1982), pp. 381–416.

[Fag82] Ronald Fagin, "Horn clauses and database dependencies," *J. Assoc. Comp. Mach.*, **29**(1982), pp. 952–985.

[FKMP05] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa, "Data exchange: Semantics and query answering," *Theoret. Comput. Sci.*, **336**(2005), pp. 89–124.

[FV83] Ronald Fagin and Moshe Y. Vardi, "Armstrong databases for functional and inclusion dependencies," *Info. Process. Lett.*, **16**(1983), pp. 13–19.

[FGM96] José Alberto Fernández, John Grant, and Jack Minker, "Model theoretic approach to view updates in deductive databases.," *J. Automated Reasoning*, **17**(1996), pp. 171–197.

[GN87] Michael R. Genesereth and Nils J. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan-Kaufmann, 1987.

[GSTZ03] Sergio Greco, Cristina Sirangelo, Irina Trubitsyna, and Ester Zumpano, "Preferred repairs for inconsistent databases," in: *7th International Database Engineering and Applications Symposium (IDEAS 2003), 16-18 July 2003, Hong Kong, China*, pp. 202–211, IEEE Computer Society, 2003.

[Heg04] Stephen J. Hegner, "An order-based theory of updates for database views," *Ann. Math. Art. Intell.*, **40**(2004), pp. 63–125.

[Heg07] Stephen J. Hegner, "A model of database components and their interconnection based upon communicating views," in: Hannu Jakkola, Yashui Kiyoki, and Takehiro Tokuda, eds., *Information Modelling and Knowledge Systems XXIV*, Frontiers in Artificial Intelligence and Applications, IOS Press, 2007, In press.

[HS07] Stephen J. Hegner and Peggy Schmidt, "Update support for database views via cooperation," in: Yannis Ioannis, Boris Novikov, and Boris Rachev, eds., *Advances in Databases and Information Systems, 11th East European Conference, ADBIS 2007, Varna, Bulgaria, September 29 - October 3, 2007, Proceedings*, pp. 98–113, Volume 4690 of Lecture Notes in Computer Science, Springer-Verlag, 2007.

[JAK82] Barry E. Jacobs, Alan R. Aronson, and Anthony C. Klug, "On interpretations of relational languages and solutions to the implied constraint problem," *ACM Trans. Database Systems*, **7**(1982), pp. 291–315.

[Kel85] Arthur M. Keller, "Updating relational databases through views," PhD thesis, Stanford University, 1985.

[Lan90] Rom Langerak, "View updates in relational databases with an independent scheme," *ACM Trans. Database Systems*, **15**(1990), pp. 40–66.

[Mon76] J. Donald Monk, *Mathematical Logic*, Springer-Verlag, 1976.

[PDGV89] Jan Paredaens, Paul De Bra, Marc Gyssens, and Dirk Van Gucht, *The Structure of the Relational Database Model*, Springer-Verlag, 1989.