Dynamic Functional Dependencies and Database Aging

VICTOR VIANU

University of California, San Diego, La Jolla, California

Abstract. A simple extension of the relational model is introduced to study the effects of dynamic constraints on database evolution. Both static and dynamic constraints are used in conjunction with the model. The static constraints considered here are functional dependencies (FDs). The dynamic constraints involve global updates and are restricted to certain analogs of FDs, called "dynamic" FDs. The results concern the effect of the dynamic constraints on the static constraints satisfied by the database in the course of time. The effect of the past history of the database on the static constraints is investigated using the notions of age and age closure. The connection between the static constraints and the potential future evolution of the database is briefly discussed using the notions of survivability and survivability closure.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design; H.2.3 [**Database Management**]: Languages—*data description languages*

General Terms: Design, Management, Theory, Verification

Additional Key Words and Phrases: Aging, database schema, dynamic constraints, evolution, functional dependency, relational database, static constraints

1. *Introduction*

One important aspect of understanding objects in the real world concerns the laws that govern their evolution in time. Many database models capture this essential semantic component by imposing "dynamic constraints" on how the database can be changed [6, 9, 13, 19]. Previous work on this topic has focused mainly on expressing wide classes of dynamic constraints using a logic-based formalism [7, 8, 10, 16, 22]. Missing from this predominantly descriptive approach has been an investigation of the *effects* of dynamic constraints on database evolution. This paper marks the beginning of such an effort. The purpose of the paper is to introduce a simple, formal model for the evolution of databases in time, and to use this model to study the effects on database evolution of a significant but tractable type of dynamic constraint.

Our model is a simple extension of the relational model. A database is viewed as a sequence of (relational) instances in time. Each new instance is obtained from the previous one by updates, insertions, and deletions. The fact that tuples preserve

An extended abstract of this paper appeared in *Proceedings of the ACM SIGACT-SIGMOD Symposium on Principles of Database Systems.* ACM, New York, 1983, pp. 389–399, under the title "Dynamic Constraints and Database Evolution."

The author was supported in part by the National Science Foundation under grant MCS 79-25004.

Author's address: Department of Electrical Engineering and Computer Science, MC-014, University of California, San Diego, La Jolla, CA 92093.

their identity through updates is formally expressed. Two types of constraints are used in conjunction with database sequences: static and dynamic. The static constraints (imposed on each instance in a sequence) are restricted here to functional dependencies (FDs). The dynamic constraints (which connect each instance in a sequence to its successor) are related to global updates, that is, updates affecting several (or all) tuples in the database simultaneously. These constraints are restricted to certain "dynamic" analogs of FDs. The types of static and dynamic constraints chosen are motivated by their simplicity and frequent occurrence in real situations. A wide variety of other constraints can also be used in conjunction with our model.

Our results focus primarily on inferring static constraints from knowledge about the evolution of the database, as expressed by the dynamic constraints. The paper is divided into seven sections, of which the first is the introduction and the second is devoted to preliminary concepts. The formal description of the model and constraints, and two motivating examples, are presented in Section 3. In Section 4 it is shown how to infer static constraints in the context of a single global update satisfying given dynamic constraints. Section 5 concerns the notion of "age" (which summarizes the past history of the database) and its connection with static constraints. The effect of age is described through the notion of "age closure" of the static constraints. Two methods of computing age closure are presented. The sequence of age closures is shown to converge to a constant, and an inference-rule-based mechanism for computing the "limit" is exhibited. Section 6 is a summary of results concerning the notion of "survivability" (which reflects the potential future evolution of a database) and its effect on the static constraints. Finally, Section 7 contains some conclusions and a discussion of future work.

2. *Preliminaries*

In this section we present some well-known concepts of the relational model, and state a few basic facts used throughout the paper. Concepts occurring in just a few places will be introduced when needed.

Let Attr and Dom be two infinite sets. Attr is a set of attributes, and Dom (called the *domain*) is the set of *values*. (Dom is the domain for each attribute in Attr.)

Let U be a set of attributes. A *tuple over* U is a mapping from U to Dom. The set of all tuples over U is denoted $\mathrm{Tup}(U)$. A *relation* or *instance over* U is a finite subset of $\mathrm{Tup}(U)$. A *family of instances over* U is a set of instances over U.

In general, $U, V, \ldots, (u, v, \ldots)$ denotes sets of attributes[1] (tuples over those sets of attributes). Usually, $A, B, \ldots$ are attributes, 1, 2, ... values in Dom, and $I, J, \ldots$ instances. We adopt the usual convention of writing UV for the union $U \cup V$ and A for a set $\{A\}$ of one element.

We use the classical representation for tuples and instances. Thus $\langle a_1, \ldots, a_n \rangle$ stands for the tuple u over $U = A_1, \cdots, A_n$ such that $u(A_i) = a_i$ for each i. And an instance is represented by a table in which a *column* is associated with each attribute and a *row* with each tuple.

We now recall the operation of projection of tuples, instances, and families of instances.

Definition. Let U be a set of attributes and $V \subseteq U$. If u is a tuple over U, then the *projection of* u *onto* V (denoted $\Pi_V(u)$) is the unique tuple v over V such that $v(A) = u(A)$ for each A in V. If I is an instance over U, then the *projection*

[1] Most sets of attributes considered in this paper are finite. Therefore, a set of attributes is assumed finite unless otherwise specified.

of I onto V is the instance $\Pi_V(I) = \{\Pi_V(u) \mid u \in I\}$ over V. And if $\mathcal{I}$ is a family of instances over U, then the *projection of* $\mathcal{I}$ *onto* V is the family of instances $\Pi_V(\mathcal{I}) = \{\Pi_V(I) \mid I \in \mathcal{I}\}$ over V.

In the remainder of the section we focus on functional dependencies [3, 11, 14, 24], a concept of major concern to us in this paper.

Definition. A functional dependency (FD) over U is a formal expression $X \rightarrow Y$ where[2] $X \neq \emptyset$, $Y \neq \emptyset$, and $XY \subseteq U$. The set of all FDs over U is denoted by $\mathrm{FD}(U)$. An *FD schema* is a pair (U, Σ), where $\Sigma \subseteq \mathrm{FD}(U)$. An instance I over U *satisfies* $X \rightarrow Y$ (denoted $I \models X \rightarrow Y$) if for each $u, v \in I$, $\Pi_X(u) = \Pi_X(v)$ implies $\Pi_Y(u) = \Pi_Y(v)$. Let Σ be a set of FDs over U. The *closure* Σ^{*U} (or Σ^* when U is understood) of Σ is the set of all FDs f over U such that each instance satisfying Σ also satisfies f. Σ is said to be *closed* if $\Sigma = \Sigma^*$.

Closure is one of the central notions relating to FDs. It can be computed using (one of several known) sets of sound and complete inference rules for FDs [3, 4, 24].

Let $\Sigma \subseteq \mathrm{FD}(U)$. The set of all instances over U satisfying Σ is denoted by $\mathrm{SAT}(U, \Sigma)$ ($\mathrm{SAT}(\Sigma)$ whenever U is understood). This leads to the notion of an "FD family" [18].

Definition. A family of instances over U is an *FD family* if it equals $\mathrm{SAT}(U, \Sigma)$ for some $\Sigma \subseteq \mathrm{FD}(U)$.

The projection of a set of FDs will be used extensively in the paper. This operation is defined next.

Definition. For each $\Sigma \subseteq \mathrm{FD}(U)$ and $V \subseteq U$, the *projection of* Σ *onto* V is the set $\Pi_V(\Sigma) = \Sigma \cap \mathrm{FD}(V)$ of FDs over V.

The following is an immediate consequence of the definitions of projection and closure.

PROPOSITION 2.1. *For each* $\Sigma \subseteq FD(U)$ *and* $V \subseteq U$, $\Pi_V(\Sigma^*)$ *is closed.*

Finally, we define the notion of "saturated" (or "closed") sets of attributes [3, 17, 24] and state a few basic facts about them.

Definition. Let $\Sigma \subseteq \mathrm{FD}(U)$. A set of attributes $X \subseteq U$ is *saturated* (or *closed*) with respect to Σ if $Z \subseteq X$ for each $Y \rightarrow Z \in \Sigma$ with $Y \subseteq X$. The *saturation* (or *closure*) $\mathrm{Sat}_\Sigma(X)$ of X with respect to Σ is the smallest set of attributes containing X and saturated with respect to Σ.

It is known that the saturation of a set of attributes always exists.

The following useful facts are immediate consequences of the definition of $\mathrm{Sat}_\Sigma(X)$.

PROPOSITION 2.2. *For each* $\Sigma, \Sigma' \subseteq FD(U)$ *and* $X, Y \subseteq U$,

(*a*) $\Sigma^* = \{Z \rightarrow Sat_\Sigma(Z) \mid Z \subseteq U\}^*$,
(*b*) $\Sigma \subseteq \Sigma'$ *implies* $Sat_\Sigma(X) \subseteq Sat_{\Sigma'}(X)$,
(*c*) $X \subseteq Y$ *implies* $Sat_\Sigma(X) \subseteq Sat_\Sigma(Y)$,
(*d*) $X \subseteq Sat_\Sigma(X)$,
(*e*) $Sat_\Sigma(Sat_\Sigma(X)) = Sat_\Sigma(X)$,
(*f*) $Sat_\Sigma(X) = Sat_{\{f\}}(X)$, *where* $f = X \rightarrow Sat_\Sigma(X)$,

[2] Other definitions of FDs allow empty left- or right-hand sides [4].

(*g*) $Sat_\Sigma(Sat_{\Sigma'}(X)) \subseteq Sat_{\Sigma\cup\Sigma'}(X)$,
(*h*) *for each* $A \in U$, $A \in Sat_\Sigma(X)$ *iff* $X \to A \in \Sigma^*$,
(*i*) $X \subseteq Y \subseteq Sat_\Sigma(X)$ *implies* $Sat_\Sigma(X) = Sat_\Sigma(Y)$.

The next proposition shows how $\mathrm{Sat}_\Sigma(X)$ can be computed from X and Σ (see [24] for proof).

PROPOSITION 2.3. *Let* $\Sigma \subseteq FD(U)$ *and* $X \subseteq U$. *For each* $A \in Sat_\Sigma(X)$ *there exist* $n \geq 0$ *and* $X^{(i)}$, $0 \leq i \leq n$, *such that* $X^{(0)} = X$, $A \in X^{(n)}$, $A \notin X^{(i)}$ *for* $i < n$, *and* $X^{(i+1)} = X^{(i)} \cup \{B \in U \mid$ *there exists* $T \to V \in \Sigma$ *such that* $T \subseteq X^{(i)}$ *and* $B \in V\}$.

Using Proposition 2.3, we can easily obtain the following two useful results (proofs omitted):

PROPOSITION 2.4. *For each* Σ, $\Sigma' \subseteq FD(U)$ *and*[3] $X \subseteq U$,

(*a*) $Sat_{\Sigma\cup\Sigma'}(X) = \bigcup_{i\geq 0} (Sat_\Sigma \circ Sat_{\Sigma'})^i(X)$,
(*b*) $Sat_{\Sigma\cup\Sigma'}(X) = (Sat_\Sigma \circ Sat_{\Sigma'})^n(X)$ *for some* $n \geq 0$,
(*c*) $(Sat_\Sigma \circ Sat_{\Sigma'})^i(X) \subseteq (Sat_\Sigma \circ Sat_{\Sigma'})^{i+1}(X)$ *for each* $i \geq 0$.

PROPOSITION 2.5. *For each* $\Sigma \subseteq FD(U)$ *and* $X \to Y \in \Sigma^*$ *there exist* $n \geq 0$, $X^{(i)}$, $Y^{(i)}$, $Z^{(i)} \subseteq U$ $(0 \leq i \leq n)$ *such that* $X^{(0)} = X$, $X^{(n)} \supseteq Y$, $Y^{(0)} = Z^{(n)} = \emptyset$, *and* $Z^{(i)} \subseteq X^{(i)}$, $Z^{(i)} \to Y^{(i+1)} \in \Sigma$, *and* $X^{(i+1)} = X^{(i)} \cup Y^{(i+1)}$ *for each* $i < n$.

3. *The Model*

In this section we present our "dynamic" model for the evolution of databases in time. We then motivate and define the dynamic constraints used in conjunction with the model and prove a few basic facts about them. In particular, we present two extended examples (3.1 and 3.2) used to motivate the dynamic constraints and to illustrate some of the main concepts of the paper.

As mentioned in the introduction to the paper, our model is a simple extension of the relational model. Informally, a database is viewed as a sequence of instances in time. Each instance of the database consists of a single relation.[4] A change of state is realized by updates, insertions, and deletions. (Updates consist of changing some values of tuples already in the relation—however, tuples can be left unaltered in an update. Insertions consist of inserting new tuples in the relation. Deletions consist of removing tuples.)

The above is formalized by the fundamental notion of a "database sequence." Intuitively, a database sequence is a sequence of successive instances of the database, together with "update mappings" from each "old" to each "new" instance. In order to treat finite and infinite database sequences in a unified way, we shall affix to them index sets S (sometimes unspecified), which are defined next.

Definition. Let N be the set of nonnegative integers. A subset S of N is *initial* if either $S = N$ or $S = \{i \mid 0 \leq i \leq n\}$ for some $n \in N$. For each initial subset S of N, let $S_0 = S - \max S$ if $\max S$ exists, and $S_0 = S$ otherwise. Thus, $S = S_0$ iff $S = N$.

Using the above terminology, we are now able to formalize the notion of a database sequence.

[3] Sat_Σ is the function from 2^U into 2^U that maps X into $\mathrm{Sat}_\Sigma(X)$. Suppose f and g are functions. Then (i) $(f \circ g)(x) = g(f(x))$, and (ii) f^0 is the identity mapping and f^i is defined inductively by $f^i = f^{i-1} \circ f$ for all $i \geq 1$.

[4] The model can be extended easily to multirelational databases.

Definition. A *database sequence* (DBS) over U is a sequence $\{(I_i, \mu_i)\}_{i \in S}$, where S is an initial subset of N, each I_i ($i \in S$) is an instance over U, each μ_i ($i \in S_0$) is a partial one-to-one mapping from I_i to I_{i+1}, and $\mu_{\max S} = \emptyset$ if max S exists.

Semantically, a tuple is regarded as a representation of an object. Since a tuple and its updated version represent the same object (but at two different moments in time), each tuple preserves its identity in an update. This is formally captured by the update mappings. Thus, for each tuple x in I_i, $\mu_i(x)$ is the updated version of x in I_{i+1}. (In particular, $\mu_i(x) = x$ if x is unchanged.) Also, each tuple in I_i not in dom μ_i is deleted and each tuple in I_{i+1} not in range μ_i is an inserted tuple.[5] (All the remarks apply for $i \in S_0$.)

In our extension of the relational model, database sequences are the natural analog of relations. While a relation represents just a "snapshot" of a database, a DBS models, in addition, the evolution of a database in time. As in the case of relations, DBSs alone capture little semantic information about the real-world objects being modeled in the database. Just as in the case of relations, constraints are used to incorporate additional semantics in the model. The constraints reflect certain properties of real-world objects. Properties describing the object at each moment in time are reflected by traditional "static" constraints (of relational database theory) imposed on each instance in the DBS. Properties concerning the *evolution* of the object in time are reflected by "dynamic" constraints, that is, constraints showing how each instance in the DBS is connected to the others. Our model, in conjunction with the static and dynamic constraints, enables us to study certain phenomena that could not be investigated within the traditional framework of relational database theory. The essential new element is the presence of dynamic constraints (mentioned above), which restrict the evolution of databases (according to certain semantic requirements). Understanding the effect of dynamic constraints on database evolution is the primary concern in this paper.

In this first investigation we focus on database sequences described by certain simple types of dynamic and static constraints. The static constraints (imposed on each instance in the sequence) are restricted to functional dependencies. The dynamic constraints (which relate each instance in the sequence to its successor) are based on functional dependencies and are discussed shortly. First, however, we present two examples that motivate our dynamic constraints and other related concepts.

Example 3.1. Consider an "employee" database containing a relation EMP with attributes ID#, NAME, SEX, AGE, RACE, MERIT, SAL.

The ID# uniquely identifies each employee; MERIT is an integer between 0 and 10 that reflects the performance of the employee, 10 being best,[6] and SAL is the salary. Since ID# is a key, the static constraint

(s1) ID# $\rightarrow$ NAME SEX AGE RACE MERIT SAL

must always hold. The policy of the company is to have an annual salary increase on each July 1. At that time, the database is globally updated to reflect changes in salary. When this occurs, the new values of certain attributes are related to the old values of other attributes. For example, the old ID# of an employee determines

[5] As is customary, dom f and range f denote the domain and range of the mapping f, respectively.

[6] We assume that MERIT can be freely updated at any time.

the new ID#.[7] For each attribute A, let $\check{A}$ denote its old value and $\hat{A}$ its new value. Then the above can be expressed by the "dynamic" functional dependency $\check{\text{ID\#}} \rightarrow \hat{\text{ID\#}}$. In addition, no two employeess (with different ID#s) receive the same new ID# as a result of an update. Thus, the dynamic FD $\hat{\text{ID\#}} \rightarrow \check{\text{ID\#}}$ also holds. (Two such dynamic FDs are abbreviated by $\check{\text{ID\#}} \leftrightarrow \hat{\text{ID\#}}$.) Similar arguments can be made for NAME, SEX, RACE, and AGE. Thus, the following dynamic FDs hold at each update:

(d1) $\check{\text{ID\#}} \leftrightarrow \hat{\text{ID\#}}$, $\check{\text{NAME}} \leftrightarrow \hat{\text{NAME}}$, $\check{\text{SEX}} \leftrightarrow \hat{\text{SEX}}$, $\check{\text{RACE}} \leftrightarrow \hat{\text{RACE}}$, $\check{\text{AGE}} \leftrightarrow \hat{\text{AGE}}$.

Let us look at two possible policies relating to salary increases. First, consider the "equal opportunity" policy, where each new salary is determined solely by merit in conjunction with the old salary. This is reflected in the database by the satisfaction of the additional dynamic FD

(d2) $\check{\text{MERIT}}\ \check{\text{SAL}} \rightarrow \hat{\text{SAL}}$.

Next consider a "discriminatory" policy, where each salary is determined strictly on the basis of sex, race, and age. Then instead of (d2), each update satisfies the dynamic FD

(d3) $\check{\text{SEX}}\ \check{\text{RACE}}\ \check{\text{AGE}} \rightarrow \hat{\text{SAL}}$.

Now let us look at the interaction between the static constraint (s1) and the various sets of dynamic constraints (d1), (d1, d2), and (d1, d3). Suppose EMP satisfies (s1) and is updated according to (d1). Owing to the dynamic FDs $\check{\text{ID}} \leftrightarrow \hat{\text{ID}}$, it is easily seen that (s1) is satisfied automatically by the new EMP. Also, it can be seen that no static constraints in addition to $\{(s1)\}^*$ must hold. If EMP is updated according to (d1, d2)—the equal opportunity policy—the same situation occurs. Now suppose EMP is updated according to (d1, d3)—the discriminatory policy. As earler, (s1) is preserved in the new EMP. In addition, however, the new version of EMP must satisfy the static FD

(s2) SEX RACE AGE → SAL.

(This is because $\hat{\text{SEX}}\ \hat{\text{RACE}}\ \hat{\text{AGE}} \rightarrow \check{\text{SEX}}\ \check{\text{RACE}}\ \check{\text{AGE}}$ and $\check{\text{SEX}}\ \check{\text{RACE}}\ \check{\text{AGE}} \rightarrow \hat{\text{SAL}}$ hold. Then $\hat{\text{SEX}}\ \hat{\text{RACE}}\ \hat{\text{AGE}} \rightarrow \hat{\text{SAL}}$ holds by transitivity. A formal way to compute the static FDs implied in an update is provided by "dynamic mappings," a mechanism introduced in Section 4.) In fact, each instance of the database must now satisfy (s2) after at least one update has occurred. □

Philosophically, there is an important difference between the nature of the dynamic constraints reflected in (d1) and that of (d2) and (d3) (Example 3.1). The first group, informally referred to as "intrinsic," arises from the very nature of the objects considered (employees) and is inseparable from our understanding of these objects. (For example, an employee cannot change his race.) This is not the case with the second group, called "extrinsic." Indeed, (d2) and (d3) are the result of certain outside variable factors, in this case human policy. Intuitively, intrinsic dynamic constraints should not produce any additional static constraints, since (assuming we have a complete understanding of the object) each such "consequence" should have been anticipated in the first place. (However, certain static

[7] It is conceivable that the value of ID# may occasionally change. For example, U.S.C. uses temporary numbers for its foreign employees who do not have a social security number.

constraints can be preserved by updates. An example is provided by (d1), which preserves (s1) but does not imply any new static FDs.) On the other hand, "extrinsic" dynamic constraints can be expected to imply additional static constraints, which arise from the particular type of evolution imposed on the object.[8] For example, (d3) implies the new static FD (s2).

In the simple example above, the static consequences are easy to anticipate. In particular, they become apparent after a single update. This is not always the case, as seen in the next example. (It is shown in Section 5 that it can take arbitrarily long for all static consequences of the evolution to become present in the database.)

Example 3.2. Consider the following database, used by the Credit and Loans Division of a bank. (For simplicity, many real-life details are omitted.) The database contains a relation called CAMPAIGN, which is used as part of an annual campaign to decide which loans and credit card offers are mailed out to each of the bank's customers. This is done according to a fixed policy, detailed below. The (relevant) attributes are

ACCT#: Account number.
BAL: The value of BAL is 0 if the customer's savings account balance is under $2000 and 1 if it is over $2000.
MC, VISA, AMEX: Each domain is equal to {YES, NO}. The value is YES if the customer has been offered a Master Charge card (MC), VISA card (VISA), or American Express card (AMEX). The value is also YES when a tuple representing a new customer is first inserted in the database, if the customer has the card from his old bank. In all other cases, the value is NO.
CAR, RV, HOUSE: Each domain is equal to {YES, NO}. The value is YES iff the customer has been offered a car loan (CAR), an RV loan (RV), or a house loan (HOUSE).

The database is updated annually, each time a mail campaign occurs.

Let us consider the static and dynamic FDs satisfied by the CAMPAIGN relation. First note that the static FD

(s1) ACCT# $\rightarrow$ BAL MC VISA AMEX CAR RV HOUSE

always holds, since ACCT# is a key. For the same reason, the dynamic FDs

(d1) $\widetilde{\text{ACCT\#}} \leftrightarrow \widehat{\text{ACCT\#}}$

always hold in any update.

Now let us look at the bank's policy with regard to the credit cards and loans offered to customers, and at the dynamic FDs that are a consequence of this policy.

(1) The credit division has the following rules concerning credit cards.[9]

  (a) A Master Charge card is offered to any customer with a savings account balance of at least $2000. This induces the dynamic FDs $\widetilde{\text{BAL}} \leftrightarrow \widehat{\text{MC}}$.
  (b) A VISA card is offered to each person who was offered a Master Charge card last year (or became a customer then and already had an MC) and is still a customer of the bank. Thus, the dynamic FDs $\widetilde{\text{MC}} \leftrightarrow \widehat{\text{VISA}}$ hold.

[8] In other words, specific evolutionary conditions may generate objects endowed with specific additional qualities. Note that similar observations are made in [12].

[9] Since the cost of making offers is very low, the bank is not concerned about superfluous offers (e.g., to persons who already have the credit card). The bank's primary concern (and that reflected in the outlined policy) is to make offers only to customers who meet its eligibility requirements.

(c) An American Express card is offered to each person who was offered a VISA card last year (or became a customer then and already had a VISA) and is still a customer of the bank. Hence the dynamic FDs $\breve{\text{VISA}} \leftrightarrow \hat{\text{AMEX}}$ hold.

(2) The loans division has the following rules:

(a) A car loan is offered to all customers with a balance of at least \$2000. This is reflected by the dynamic FDs $\breve{\text{BAL}} \leftrightarrow \hat{\text{CAR}}$.

(b) An RV loan is offered to all persons who were offered a car loan last year and are still customers of the bank. Thus, the dynamic FDs $\breve{\text{CAR}} \leftrightarrow \hat{\text{RV}}$ hold.

(c) A house loan is offered to all persons who were offered RV loans last year and are still with the bank. Hence the dynamic FDs $\breve{\text{RV}} \leftrightarrow \hat{\text{HOUSE}}$ hold.

Altogether, in addition to (d1) the following dynamic FDs hold:

(d2) $\breve{\text{BAL}} \leftrightarrow \hat{\text{MC}}$, $\breve{\text{MC}} \leftrightarrow \hat{\text{VISA}}$, $\breve{\text{VISA}} \leftrightarrow \hat{\text{AMEX}}$,
$\breve{\text{BAL}} \leftrightarrow \hat{\text{CAR}}$, $\breve{\text{CAR}} \leftrightarrow \hat{\text{RV}}$, $\breve{\text{RV}} \leftrightarrow \hat{\text{HOUSE}}$.

Let us look at the effect of the set of dynamic FDs (d1, d2) on the static constraints. In this example, the full static consequences of the dynamic constraints become apparent in the course of three updates. Indeed, it can be verified that

(3) If CAMPAIGN satisfies (s1) and is updated once according to (d1, d2), then (s1) is preserved in the new CAMPAIGN and the additional FDs

(s2) MC $\leftrightarrow$ CAR

hold.

(4) If CAMPAIGN satisfies (s1) and is updated twice according to (d1, d2), then (s1) is preserved and the additional FDs

(s3) MC $\leftrightarrow$ CAR and VISA $\leftrightarrow$ RV

hold.

(5) If CAMPAIGN satisfies (s1) and is updated three or more times according to (d1, d2), then (s1) is preserved and the additional FDs

(s4) MC $\leftrightarrow$ CAR, VISA $\leftrightarrow$ RV, and AMEX $\leftrightarrow$ HOUSE

hold. Thus the above FDs always hold for customers who were with the bank for at least 3 years after the credit and loan policies went into effect. (The tuples representing these customers are said to have attained age 3 in the database, i.e., were updated at least 3 times.) □

Note that in the above example, the dynamic FDs are "bidirectional" (of the form $X \leftrightarrow Y$). However, "unidirectional" dynamic FDs are by no means uncommon. For instance, suppose that the credit card policy in Example 3.2 is modified so that a customer is offered a VISA card if he or she was offered an MC card last year *or* if his or her savings account balance is at least \$2000. Then each update would satisfy $\breve{\text{MC}}\ \breve{\text{BAL}} \rightarrow \hat{\text{VISA}}$, but the dynamic FD $\hat{\text{VISA}} \rightarrow \breve{\text{MC}}\ \breve{\text{BAL}}$ would not generally hold.

The previous examples exhibit certain dynamic constraints that occur frequently in real situations. These constraints focus solely on global updates and are similar in nature to functional dependencies. (They do not deal with individual tuple updates, insertions, or deletions.) We shall formalize our dynamic constraints along

A	B		A	B
0	1	→	1	0
1	1	→	1	1

μ

FIGURE 1

$\check{A}$	$\check{B}$	$\hat{A}$	$\hat{B}$
0	1	1	0
1	1	1	1

FIGURE 2

the same lines. First, though, we need the notions of an update and its associated "action relation."

Definition. Let U be a set of attributes. An *update* over U is a triple (I, μ, J), where I and J are instances over U and μ is a one-to-one (total) mapping from I onto J.

If $\{(I_i, \mu_i)\}_{i \in S}$ is a DBS over U, then $(\text{dom}\,\mu_i, \mu_i, \text{range}\,\mu_i)$ is an update over U for each $i \in S_0$.

Our dynamic constraints are defined using the "action relation" associated with an update.[10] Intuitively, the action relation is formed by concatenating each tuple in the "old" instance with its updated version. Before giving the formal definition, we introduce some symbolism to denote the corresponding "versions" of the attributes, one for the "old" values and another for the "new."

Notation. For each attribute A, let $\check{A}$ and $\hat{A}$ be new attributes. For each set U of attributes, let $\check{U} = \{\check{A} \mid A \in U\}$ and $\hat{U} = \{\hat{A} \mid A \in U\}$. For each tuple x over U let $\check{x}$ and $\hat{x}$ be the tuples over $\check{U}$ and $\hat{U}$, respectively, such that $\check{x}(\check{A}) = \hat{x}(\hat{A}) = x(A)$ for each $A \in U$. Finally, for each $\Sigma \subseteq \text{FD}(U)$ let $\check{\Sigma} = \{\check{X} \to \check{Y} \mid X \to Y \in \Sigma\}$ and $\hat{\Sigma} = \{\hat{X} \to \hat{Y} \mid X \to Y \in \Sigma\}$.

Intuitively, $\check{A}$ corresponds to the old value of A while $\hat{A}$ corresponds to the new value, $\check{\Sigma}$ and $\hat{\Sigma}$ are the "copies" of Σ corresponding to the two versions of the attributes, and $\check{x}$ and $\hat{x}$ are "copies" of x.

Using this notation, we proceed with the definition of an action relation.

Definition. The *action relation associated with an update* (I, μ, I') is the relation[11] $I\mu I' = \{\check{x} \times \widehat{\mu(x)} \mid x \in I\}$.

Example. Consider the update given in Figure 1. Its associated action relation is given in Figure 2.

We are now ready to define our dynamic constraints, these being an analog of functional dependencies to updates.

Definition. A *dynamic functional dependency* (DFD) over U is an FD $X \to Y$ over $\check{U}\hat{U}$ such that, for each $A \in Y$, $XA \cap \check{U} \neq \emptyset$ and $XA \cap \hat{U} \neq \emptyset$. The set of all DFDs over U is denoted by $\text{DFD}(U)$.

Informally, the above condition on FDs $X \to Y$ over $\check{U}\hat{U}$ ensures that $X \to Y$ does not imply any nontrivial FDs over $\check{U}$ or over $\hat{U}$. (These would not truly be dynamic constraints.) For example, if $U = ABC$, then $\check{A} \to \hat{B}$ is a DFD, while $\check{A} \to \hat{B}\check{C}$ is not.

[10] The notion of action relation discussed here is related to that considered in [22]. There, as in this paper, the marked attributes of action relations are used to denote old and new values of attributes. In [22] the emphasis is on showing how action relations can be used to express a large class of constraints on updates. Here the focus is on a tractable constraint and an exploration of its properties.

[11] The symbol × denotes the cross-product of two tuples over disjoint domains.

The reader will note that DFDs are a natural extension of FDs, and that other types of constraints (e.g., multivalued dependencies (MVDs), join dependencies (JDs), embedded implication dependencies (EIDs), [21]) can be extended in a similar manner.

We now show how to use DFDs as a constraint on updates.

Definition. An update (I, μ, I') *satisfies* a set Δ of DFDs, written $(I, \mu, I') \models \Delta$, if the action relation $I\mu I'$ satisfies Δ.

For example, the update in Figure 1 satisfies $\check{B} \rightarrow \hat{A}$ but not $\hat{A} \rightarrow \check{A}$.

Using the dynamic constraints on updates, we now place constraints on DBSs. Valid DBSs are described by a set of static FDs (Σ) and a set of dynamic FDs (Δ). This leads to the notion of a "DFD schema."

Definition. A *DFD schema* is a triple (U, Σ, Δ), where U is a set of attributes, Σ is a set of FDs over U, and Δ is a set of DFDs over U.

We now have

Definition. Let (U, Σ, Δ) be a DFD schema. *A database sequence* $\{(I_i, \mu_i)\}_{i \in S}$ *satisfies* (Σ, Δ) (denoted $\{(I_i, \mu_i)\}_{i \in S} \models (\Sigma, \Delta)$) if each I_i, $i \in S$, satisfies Σ, and each update $(\text{dom}\,\mu_i, \mu_i, \text{range}\,\mu_i)$, $i \in S_0$, satisfies Δ. The set of all DBSs over U satisfying (Σ, Δ) is denoted by $\text{SAT}(U, \Sigma, \Delta)$.

If $f \in \text{FD}(U)$ and $d \in \text{DFD}(U)$, then $\{(I_i, \mu_i)\}_{i \in S} \models f$ indicates that each I_i satisfies f; similarly, $\{(I_i, \mu_i)\}_{i \in S} \models d$ indicates that each update $(\text{dom}\,\mu_i, \mu_i, \text{range}\,\mu_i)$ satisfies d.

Remark. Consider an update (I, μ, I') over U. Since tuples do not repeat in a relation, the action relation $I\mu I'$ does not contain distinct tuples that agree on $\check{U}$ or on $\hat{U}$. Consequently, each action relation corresponding to an update over U vacuously satisfies the DFDs $\check{U} \rightarrow \hat{U}$ and $\hat{U} \rightarrow \check{U}$. In a sense, these DFDs are trivial. However, they clearly do not follow from the traditional inference rules for FDs. In order to avoid problems arising from the "incompleteness" of the inference rules within the class of action relations, each set of DFDs over U *will henceforth be assumed to contain the above two DFDs.* □

Since "bidirectional" FDs (such as $\check{U} \rightarrow \hat{U}$ and $\hat{U} \rightarrow \check{U}$) occur frequently, the following is used:

Notation. For each set of attributes U and $X, Y \subseteq U$ $(X \neq \emptyset, Y \neq \emptyset)$, $X \leftrightarrow Y$ denotes the FDs $X \rightarrow Y$ and $Y \rightarrow X$. The set $\{\check{U} \leftrightarrow \hat{U}\}$ is denoted by Δ_U.

It turns out that arbitrary DFDs are sometimes troublesome to deal with mathematically. A better behaved subclass of the DFDs, called "bipartite DFDs," is suitable for most real situations. (In particular, all the DFDs in Examples 3.1 and 3.2 are bipartite. So are the DFDs $\check{U} \leftrightarrow \hat{U}$ of Δ_U.) Intuitively, bipartite DFDs are DFDs where "old" and "new" attributes do not mix on the same side. The relation between DFDs and bipartite DFDs is similar to that between phrase-structure grammars and context-free grammars: The first is appealing through its generality, whereas the second is mathematically simpler but still powerful enough to model most applications.

Definition. A DFD $X \rightarrow Y$ over U is a *bipartite DFD* (BDFD) if either $X \subseteq \check{U}$ and $Y \subseteq \hat{U}$ or $X \subseteq \hat{U}$ and $Y \subseteq \check{U}$. The set of all BDFDs over U is denoted by $\text{BDFD}(U)$. Finally, a *BDFD schema* is a DFD schema (U, Σ, Δ) where $\Delta \subseteq \text{BDFD}(U)$.

For instance, given $U = ABC$, $\check{A}\check{B} \to \hat{C}$ is a BDFD, whereas $\check{A}\hat{B} \to \hat{C}$ is not.

Although most definitions in the paper are given for arbitrary DFDs, some of the results in Sections 4–6 hold just for BDFDs.

Note that a restriction analogous to the "bipartite" restriction on DFDs can be made on the dynamic extensions of other static constraints, for example, embedded implicational dependencies [15].

In the remainder of the section we focus on two basic problems relating to DFD schemas. The first concerns the logical closure of the constraints (Σ, Δ) of a DFD schema (U, Σ, Δ). The second concerns the equivalence of DFD schemas.

The logical closure of (Σ, Δ) is defined in a way similar to the closure of traditional "static" constraints:

Definition. Let (U, Σ, Δ) be a DFD schema. The *closure* $(\Sigma, \Delta)^*$ of (Σ, Δ) relative to U is the pair (Σ', Δ'), where

$$\Sigma' = \{f \in \mathrm{FD}(U) \mid \text{for each DBS } s \text{ over } U, \text{ if } s \models (\Sigma, \Delta), \text{ then } s \models f\},$$

and

$$\Delta' = \{d \in \mathrm{DFD}(U) \mid \text{for each DBS } s \text{ over } U, \text{ if } s \models (\Sigma, \Delta), \text{ then } s \models d\}.$$

Thus, the closure $(\Sigma, \Delta)^*$ consists of all static and dynamic constraints that are logically implied by (Σ, Δ).

The next theorem (Theorem 3.4) shows how to compute $(\Sigma, \Delta)^*$. First though, we introduce some notation and state (without proof) a useful technical result (Proposition 3.3).

Notation. For each $X \subseteq \check{U}\hat{U}$, let $\overline{X} = \{A \mid \hat{A} \in X \text{ or } \check{A} \in X\}$. For each set Δ of FDs over $\check{U}\hat{U}$, let $\overline{\Delta} = \{\overline{X} \to \overline{Y} \mid X \to Y \in \Delta\}$. For each tuple x over $\check{U}(\hat{U})$, let x be the tuple over U such that $\bar{x}(A) = x(\check{A})(x(\hat{A}))$ for each $A \in U$. And for each instance I over $\check{U}(\hat{U})$, let $\overline{I} = \{\bar{x} \mid x \in I\}$.

Thus the symbol $\bar{}$ acts on sets of attributes as an operator that erases the markers $\check{}$ and $\hat{}$. The symbol $\bar{}$ on tuples yields the same value vector but over the set of unmarked attributes.

PROPOSITION 3.3. *Let U and V be sets of attributes. Let f be an attribute isomorphism*[12] *from U onto V. Then*

(1) *for $\Sigma, \Sigma' \subseteq FD(U)$, $\Sigma \subseteq \Sigma'$ iff $f(\Sigma) \subseteq f(\Sigma')$, and*
(2) *for each relation R over U and $\Sigma \subseteq FD(U)$, $R \models \Sigma$ iff $f(R) \models f(\Sigma)$.*

Note that the mapping defined from U onto $\check{U}$ by $f(A) = \check{A}$ for each $A \in U$ is an attribute isomorphism. So is the mapping defined from $\check{U}$ onto U by $f(A) = \overline{A}$. In simple cases such as these, Proposition 3.3 is used without explicitly defining f. However, f is specified in less obvious situations.

Finally, we need the following:

Notation. For each set F of FDs over $\check{U}\hat{U}$, $\mathrm{DFD}(F)$ denotes the set of all DFDs in F.

[12] A mapping f from U into V is an *attribute isomorphism* if it is one-to-one and onto. The isomorphism f is extended to sets of FDs, tuples, and relations over U as follows. For each $\Sigma \subseteq \mathrm{FD}(U)$, $f(\Sigma) = \{f(X) \to f(Y) \mid X \to Y \in \Sigma\}$. For each tuple u over U let $f(u)$ be the tuple over V defined by $f(u)(f(A)) = u(A)$ for each $A \in U$. And for each relation R over U, $f(R) = \{f(u) \mid u \in R\}$.

We now have

THEOREM 3.4. *For each DFD schema* (U, Σ, Δ),

$$(\Sigma, \Delta)^* = (\Sigma^*, DFD((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*)).$$

PROOF. Let (U, Σ, Δ) be a DFD schema, $\Sigma' = \Sigma^*$, $\Delta' = \text{DFD}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*)$, and $(\Sigma'', \Delta'') = (\Sigma, \Delta)^*$. Clearly, it suffices to show that

(i) $\Sigma' \subseteq \Sigma''$ and $\Delta' \subseteq \Delta''$;
(ii) $\Sigma'' \subseteq \Sigma'$ and $\Delta'' \subseteq \Delta'$.

To see (i), let $s = \{(I_i, \mu_i)\}_{i \in S}$ be a DBS over U satisfying (Σ, Δ). Let $i \in S$. Since $I_i \models \Sigma$, $I_i \models \Sigma^*$. Thus $s \models \Sigma^*$, so $\Sigma' = \Sigma^* \subseteq \Sigma''$. Let $i \in S_0$. Since $I_i \models \Sigma$, $(\text{dom}\,\mu_i, \mu_i, \text{range}\,\mu_i) \models \Delta$ and $I_{i+1} \models \Sigma$, $(\text{dom}\,\mu_i)\mu_i(\text{range}\,\mu_i) \models \check{\Sigma} \cup \Delta \cup \hat{\Sigma}$. Hence $(\text{dom}\,\mu_i)\mu_i(\text{range}\,\mu_i) \models (\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*$, so $(\text{dom}\,\mu_i)\mu_i(\text{range}\,\mu_i) \models \text{DFD}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*)$. Thus, $s \models \Delta'$. Then $\Delta' \subseteq \Delta''$, so (i) holds.

Consider (ii). Let $f \in \Sigma''$ and let I be an instance over U satisfying Σ. The DBS (of length 1) $s = \{(I, \varnothing)\}$ satisfies (Σ, Δ). Hence $s \models f$ by the definition of Σ''. Thus $I \models f$. It follows that each instance I over U satisfying Σ must satisfy f. Hence $f \in \Sigma^* = \Sigma'$, and $\Sigma'' \subseteq \Sigma'$. Let $d \in \Delta^*$. Let R be a relation over $\check{U}\hat{U}$ satisfying $\check{\Sigma} \cup \Delta \cup \hat{\Sigma}$. Let (I, μ, J) be the update defined by $I = \overline{\Pi_{\check{U}}(R)}$, $J = \overline{\Pi_{\hat{U}}(R)}$, and $\mu(\Pi_{\check{U}}(u)) = \Pi_{\hat{U}}(u)$ for each $u \in R$. (The mapping μ is well defined and one to one because R satisfies the DFDs $\check{U} \leftrightarrow \hat{U}$.) Since $R \models \check{\Sigma} \cup \Delta \cup \hat{\Sigma}$, $I \models \Sigma$ and $J \models \Sigma$ by Proposition 3.3. Also, $I\mu J = R$, so $(I, \mu, J) \models \Delta$. Then the DBS $s = \{(I_i, \mu_i)\}_{0 \le i \le 1}$, where $I_0 = I$, $\mu_0 = \mu$, and $I_1 = J$, satisfies (Σ, Δ). Hence, $s \models d$ by the definition of Δ''. Thus $(I, \mu, J) \models d$ and $R = I\mu J \models d$. Consequently, each relation over $\check{U}\hat{U}$ satisfying $\check{\Sigma} \cup \Delta \cup \hat{\Sigma}$ must satisfy d. Therefore, $d \in (\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*$. Since d is a DFD, $d \in \text{DFD}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*) = \Delta'$. Hence, $\Delta'' \subseteq \Delta'$, and (ii) holds. □

Finally, let us consider the equivalence of two DFD schemas. Formally, we have

Definition. Two DFD schemas $(U_1, \Sigma_1, \Delta_1)$ and $(U_2, \Sigma_2, \Delta_2)$ are *equivalent* if

$$\text{SAT}(U_1, \Sigma_1, \Delta_1) = \text{SAT}(U_2, \Sigma_2, \Delta_2).$$

We now have the following.

FACT 3.5. *Two* DFD *schemas* $(U_1, \Sigma_1, \Delta_1)$ *and* $(U_2, \Sigma_2, \Delta_2)$ *are equivalent iff* $U_1 = U_2$ *and* $(\Sigma_1, \Delta_1)^* = (\Sigma_2, \Delta_2)^*$.

4. *Dynamic Mappings*

In Section 3 the relational model was extended to include dynamic constraints on updates. It is of particular interest to understand the interaction between these dynamic constraints and the traditional static constraints. In this section we look at one aspect of this interaction by studying the *dynamic mappings* defined by sets of dynamic constraints. The mappings describe the connection between the dynamic constraints satisfied by an update and the static constraints satisfied by the "old" and "new" versions of an instance. (The four possible combinations of "old" and "new" yield four different kinds of dynamic mappings.) After defining dynamic mappings, we give some of their basic properties and exhibit a method for computing them. (The method is based on an analogy between dynamic mappings and logical implication, and is similar to closure.)

The notion of dynamic mappings is now formalized.

Definition. Let Δ be a set of DFDs over U. The following four mappings from $2^{FD(U)}$ into $2^{FD(U)}$ are the *dynamic mappings* associated with Δ:

(a) The *old-new* mapping on_Δ associated with Δ is defined by $on_\Delta(\Sigma) = \{X \to Y \in FD(U) \mid \text{if } I \models \Sigma \text{ and } (I, \mu, I') \text{ is an update over } U \text{ satisfying } \Delta, \text{ then } I' \models X \to Y.\}$.

(b) The *new-old* mapping no_Δ associated with Δ is defined by $no_\Delta(\Sigma) = \{X \to Y \in FD(U) \mid \text{if } I' \models \Sigma \text{ and } (I, \mu, I') \text{ is an update over } U \text{ satisfying } \Delta, \text{ then } I \models X \to Y\}$.

(c) The *old-old* mapping oo_Δ associated with Δ is defined by $oo_\Delta(\Sigma) = \{X \to Y \in FD(U) \mid \text{if } I \models \Sigma \text{ and there exists an update } (I, \mu, I') \text{ over } U \text{ satisfying } \Delta, \text{ then } I \models X \to Y\}$.

(d) The *new-new* mapping nn_Δ associated with Δ is defined by $nn_\Delta(\Sigma) = \{X \to Y \in FD(U) \mid \text{if } I' \models \Sigma \text{ and there exists an update } (I, \mu, I') \text{ satisfying } \Delta, \text{ then } I' \models X \to Y\}$.

Note that the dynamic mappings are analogous to logical implication. Indeed, for each $\Sigma \subseteq FD(U)$, $on_\Delta(\Sigma)$ is the set of all FDs that must be satisfied by each "new" instance I' over U whenever the "old" instance I satisfies Σ and the update (I, μ, I') satisfies Δ. Similar remarks can be made about no_Δ, oo_Δ, and nn_Δ.

We now present without proof some immediate (and useful) properties of dynamic mappings.

PROPOSITION 4.1. *Let Δ be a set of DFDs over U and x_Δ be in $\{on_\Delta, no_\Delta, oo_\Delta, nn_\Delta\}$. Then for each $\Sigma \subseteq FD(U)$, $x_\Delta(\Sigma) = x_\Delta(\Sigma^*)$ and $x_\Delta(\Sigma)$ is a closed set of FDs.*

The definitions of the dynamic mappings are not effective. For instance, how would one compute $on_\Delta(A \to B)$ for $U = ABC$ and $\Delta = \{\hat{A} \to \check{A}, \check{A} \to \hat{B}, \check{B} \to \hat{C}\} \cup \Delta_U$? We now give one method for computing dynamic mappings:

PROPOSITION 4.2. *Let Δ be a set of DFDs over U. Then for each set Σ of FDs over U,*

(a) $on_\Delta(\Sigma) = \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^*)}$,

(b) $no_\Delta(\Sigma) = \overline{\Pi_{\check{U}}((\hat{\Sigma} \cup \Delta)^*)}$,

(c) $oo_\Delta(\Sigma) = \overline{\Pi_{\check{U}}((\check{\Sigma} \cup \Delta)^*)}$,

(d) $nn_\Delta(\Sigma) = \overline{\Pi_{\hat{U}}((\hat{\Sigma} \cup \Delta)^*)}$.

PROOF. We shall only present the argument for (a) since the other cases are similar. We first show that $on_\Delta(\Sigma) \subseteq \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^*)}$. To see this, it is enough to show that if R is a relation over $\check{U}\hat{U}$ which satisfies $\check{\Sigma} \cup \Delta$, then $\Pi_{\hat{U}}(R)$ satisfies $\widehat{on_\Delta(\Sigma)}$. Indeed, this implies that $\widehat{on_\Delta(\Sigma)} \subseteq \Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^*)$, whence $on_\Delta(\Sigma) \subseteq \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^*)}$ by Proposition 3.3.

Let R be as above. Let μ be the one-to-one mapping from $\Pi_{\check{U}}(R)$ onto $\Pi_{\hat{U}}(R)$ defined by $\mu(\Pi_{\check{U}}(u)) = \Pi_{\hat{U}}(u)$ for each $u \in R$. (The mapping is well defined because R satisfies[13] $\check{U} \to \hat{U}$. It is one to one because R satisfies $\hat{U} \to \check{U}$, and is onto from the definition.)

Let $I = \overline{\Pi_{\check{U}}(R)}$ and $I' = \overline{\Pi_{\hat{U}}(R)}$. Let $\tilde{\mu}$ be the one-to-one mapping (induced by μ) from I onto I' defined by $\tilde{\mu}(v) = \mu(\check{v})$ for each v in I. Then $(I, \tilde{\mu}, I')$ is an update, $I\tilde{\mu}I' = R$ and, since $R \models \Delta$, $(I, \tilde{\mu}, I') \models \Delta$. This and the fact that $I \models \Sigma$ imply that $I' \models on_\Delta(\Sigma)$. Thus, $\Pi_{\hat{U}}(R) \models \widehat{on_\Delta(\Sigma)}$ by Proposition 3.3.

[13] Recall that $\check{U} \to \hat{U}$ and $\hat{U} \to \check{U}$ are assumed to belong to each given set of DFDs over U.

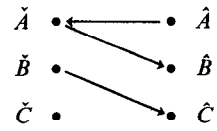


FIGURE 3

Consider the reverse inclusion, that is, $\overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^*)} \subseteq \mathrm{on}_\Delta(\Sigma)$. Let $I \models \Sigma$ and (I, μ, I') be an update over U that satisfies Δ. It suffices to show that $I' \models \Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^*)$. Let $R = I \mu I'$. Then $R \models \check{\Sigma} \cup \Delta$, so $\Pi_{\hat{U}}(R) \models \Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^*)$. Since $I' = \Pi_{\hat{U}} R$, $I' \models \Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^*)$ by Proposition 3.3. □

Let us now reconsider the earlier example, that is, $U = ABC$ and $\Delta = \{\hat{A} \to \check{A}, \check{A} \to \hat{B}, \check{B} \to \hat{C}\} \cup \Delta_U$ (Figure 3).[14] Using Proposition 4.2,

$$\begin{aligned}\mathrm{on}_\Delta(\{A \to B\}) &= \overline{\Pi_{\hat{U}}((\{\check{A} \to \check{B}\} \cup \Delta)^*)} \\ &= \overline{\Pi_{\hat{U}}(\{\check{A} \to \check{B}, \hat{A} \to \check{A}, \check{A} \to \hat{B}, \check{B} \to \hat{C}, \hat{A}\hat{B}\hat{C} \to \check{A}\check{B}\check{C}, \check{A}\check{B}\check{C} \to \hat{A}\hat{B}\hat{C}\}^*)} \\ &= \overline{\{\hat{A} \to \hat{B}, \hat{A} \to \hat{C}\}^*} = \{A \to B, A \to C\}^*.\end{aligned}$$

Similarly,

$$\begin{aligned}\mathrm{no}_\Delta(\{C \to A\}) &= \overline{\Pi_{\hat{U}}((\{\hat{C} \to \hat{A}\} \cup \Delta)^*)} = \{B \to A\}^*, \\ \mathrm{oo}_\Delta(\varnothing) &= \overline{\Pi_{\hat{U}}((\varnothing \cup \Delta)^*)} = \varnothing^*, \\ \mathrm{nn}_\Delta(\varnothing) &= \overline{\Pi_{\hat{U}}((\varnothing \cup \Delta)^*)} = \overline{\Pi_{\hat{U}}(\Delta^*)} = \{A \to B\}^*.\end{aligned}$$

The following is a useful consequence of Proposition 4.2.

COROLLARY 4.3. *Let $\Delta \subseteq DFD(U)$ and $\Sigma \subseteq FD(U)$. Then*

(*a*) $on_\Delta(oo_\Delta(\Sigma)) = on_\Delta(\Sigma)$,
(*b*) $nn_\Delta(on_\Delta(\Sigma)) = on_\Delta(\Sigma)$,
(*c*) $no_\Delta(nn_\Delta(\Sigma)) = no_\Delta(\Sigma)$,
(*d*) $oo_\Delta(no_\Delta(\Sigma)) = no_\Delta(\Sigma)$,
(*e*) $on_\Delta(no_\Delta(\Sigma)) \subseteq nn_\Delta(\Sigma)$,
(*f*) $no_\Delta(on_\Delta(\Sigma)) \subseteq oo_\Delta(\Sigma)$,
(*g*) $oo_\Delta(oo_\Delta(\Sigma)) = oo_\Delta(\Sigma)$,
(*h*) $nn_\Delta(nn_\Delta(\Sigma)) = nn_\Delta(\Sigma)$.

In some sense, the mappings no_Δ and nn_Δ can be viewed as symmetric to on_Δ and oo_Δ, respectively. This is formalized in the next proposition.

Notation. Let U be a set of attributes and σ the isomorphism from $\check{U}\hat{U}$ onto $\check{U}\hat{U}$ defined by $\sigma(\hat{A}) = \check{A}$ and $\sigma(\check{A}) = \hat{A}$ for each $A \in U$.

Note that

(*) $\overline{X} = \overline{\sigma(X)}$ for each $X \subseteq \check{U}\,(\hat{U})$,
(**) $\Pi_{\check{U}}(\sigma(\Delta)) = \sigma(\Pi_{\hat{U}}(\Delta))$ for each $\Delta \subseteq \mathrm{FD}(\check{U}\hat{U})$,
(***) $\sigma(\Delta^*) = (\sigma(\Delta))^*$ for each $\Delta \subseteq \mathrm{FD}(\check{U}\hat{U})$.

PROPOSITION 4.4 (SYMMETRY). *For each $\Delta \subseteq DFD(U)$, $no_\Delta = on_{\sigma(\Delta)}$ and $nn_\Delta = oo_{\sigma(\Delta)}$.*

[14] In order to simplify this and subsequent figures, the BDFDs $\check{U} \leftrightarrow \hat{U}$ in Δ_U are not represented.

PROOF. We only give the proof for $no_\Delta = on_{\sigma(\Delta)}$, the other case being similar.

Let $\Sigma \subseteq FD(U)$. It is easily seen that $\hat{\Sigma} \cup \Delta = \sigma(\check{\Sigma} \cup \sigma(\Delta))$, whence $(\hat{\Sigma} \cup \Delta)^* = \sigma((\check{\Sigma} \cup \sigma(\Delta))^*)$ by (***) above. Thus,

$$\begin{aligned}\Pi_{\check{U}}((\hat{\Sigma} \cup \Delta)^*) &= \Pi_{\check{U}}(\sigma((\check{\Sigma} \cup \sigma(\Delta))^*)) \\ &= \sigma(\Pi_{\hat{U}}((\check{\Sigma} \cup \sigma(\Delta))^*))\end{aligned}$$

by (**) above. By Proposition 3.3,

$$\overline{\Pi_{\check{U}}((\hat{\Sigma} \cup \Delta)^*)} = \overline{\sigma(\Pi_{\hat{U}}((\check{\Sigma} \cup \sigma(\Delta))^*))}.$$

Also, by (*),

$$\overline{\sigma(\Pi_{\hat{U}}((\check{\Sigma} \cup \sigma(\Delta))^*))} = \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \sigma(\Delta))^*)}.$$

Since $no_\Delta(\Sigma) = \overline{\Pi_{\hat{U}}((\hat{\Sigma} \cup \Delta)^*)}$ and $on_{\sigma(\Delta)}(\Sigma) = \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \sigma(\Delta))^*)}$ (by Proposition 4.2), $no_\Delta(\Sigma) = on_{\sigma(\Delta)}(\Sigma)$. □

As we shall see, the Symmetry Proposition permits us to obtain results about no_Δ or nn_Δ from corresponding results about on_Δ and oo_Δ, respectively.

The following example shows that the minimum size of a cover for $on_\Delta(\Sigma)$ can be very large compared with the sizes of Σ and Δ. Specifically, the example shows that, for each positive n, there exist sets Σ_n of FDs and Δ_n of BDFDs such that $|\Sigma_n| = 1$, $|\Delta_n| = n + 1$, and the minimum size of a cover for $on_{\Delta_n}(\Sigma_n)$ is 2^n. The example has several consequences. First, it shows that there is no algorithm for computing a cover of $on_\Delta(\Sigma)$ in time polynomial in the sizes of Σ and Δ. (However, it can be shown that there is an algorithm for computing a minimum-size cover C of $on_\Delta(\Sigma)$ in time polynomial in the sizes of Σ, Δ, *and* C. Thus, if $on_\Delta(\Sigma)$ has a "small" cover, then such a cover can be computed efficiently.) On the positive side, the example shows that for some sets Σ' of static FDs there exists sets Δ of dynamic FDs that preserve Σ' and are much easier to check than Σ', since they contain drastically fewer constraints. (Indeed, consider the example, and let Σ'_n be a minimum-size cover of $on_{\Delta_n}(\Sigma_n)$; clearly, Δ_n preserves Σ'_n and the size of Σ'_n is 2^n, whereas the size of Δ_n is only $2n + 1$.) Thus, this shows that the use of dynamic constraints can dramatically increase the efficiency of constraint checking.

Example. Let

$$\begin{aligned}U_n &= \{A_i \mid 0 \le i \le 2n\}, \\ \Sigma_n &= \{A_0 \cdots A_{n-1} \to A_{2n}\}, \\ \Delta_n &= \{\hat{A}_i \to \check{A}_{i \bmod n} \mid 0 \le i < 2n\} \cup \{\check{A}_{2n} \to \hat{A}_{2n}\} \cup \Delta_{U_n}.\end{aligned}$$

(Figure 4 represents $\check{\Sigma}_n$ and Δ_n.)

Clearly, the following is a minimum-size cover for $on_{\Delta_n}(\Sigma_n)$, and its size is 2^n: $\{A_{i_0}A_{i_1} \cdots A_{i_{n-1}} \to A_{2n} \mid 0 \le i_j < 2n,\ i_j = j \bmod n,\ 0 \le j < n\}$.

5. *Age in a Database*

As already mentioned, it is of particular interest to study the interaction between the dynamic and the static constraints satisfied by a database. In this section we look at how information about the past of the database can be used to infer static constraints satisfied by the current state. A fundamental notion here is that of age.

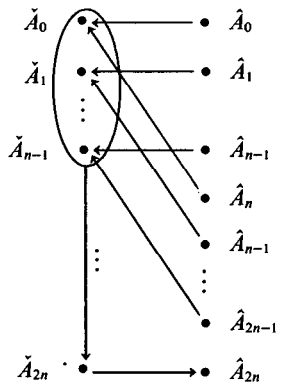


FIGURE 4

(A tuple has age k if it has been updated k times.) In order to understand the effect of age on static constraints, we define the notion of age closure of the static constraints. Two methods for computing age closure are then exhibited. Following this, we show that the sequence of age closures "converges" to a constant $\tilde{\Sigma}$. Finally, we give some properties describing $\tilde{\Sigma}$ and the convergence process, and present a simple inference-rule-based mechanism for computing $\tilde{\Sigma}$. As in the previous section, most results are proved for dynamic constraints restricted to BDFDs.

Owing to the nature of our dynamic constraints, it is useful to consider first databases where only updates are allowed. (Databases where insertions and deletions are allowed are discussed later in this section). The formalism for this is now presented.

Definition. A DBS $\{(I_i, \mu_i)\}_{i \in S}$ is *stable* if $\operatorname{dom} \mu_i = I_i$ and $\operatorname{range} \mu_i = I_{i+1}$ for each $i \in S_0$.

Since $\operatorname{dom} \mu_i = I_i$, no deletions are permitted in a stable DBS. Since $\operatorname{range} \mu_i = I_{i+1}$, no insertions are permitted. Thus, only updates are allowed in a stable DBS.

We start by looking at how the history of the database can be used to infer satisfaction of static constraints in the current state. The relevant information is the *age* of a tuple in the current state, that is, the number of updates to which a tuple has been subjected since having been inserted in the database. This is formalized for stable DBSs as follows:[15]

Definition. Let $s = \{(I_i, \mu_i)\}_{i \in S}$ be a stable DBS and u a tuple in I_k, $k \in S$. The occurrence of u in I_k has *age* k in s.

We simply say that a tuple has age k whenever it is clear which occurrence of the tuple we are referring to.

Clearly, the notion of age becomes irrelevant if each identity update of a valid instance of the database satisfies Δ. (An identity update is an update (I, μ, I'), where $\mu(x) = x$ for each $x \in I$.) Indeed, any instance of the database could then become arbitrarily "old" simply by staying unchanged. It can be shown that, given a DFD schema (U, Σ, Δ), each identity update of a valid instance satisfies Δ iff $\overline{\Delta} \subseteq \Sigma^*$ (recall that the operator $\bar{}$ erases the markers $\check{}$ and $\hat{}$).

Consider a database satisfying (Σ, Δ). As was seen from Examples 3.1 and 3.2, if tuples attain a certain age in the database, then the set of such tuples must

[15] The notion will be defined for arbitrary DBSs later in this section.

sometimes satisfy static constraints in addition to Σ^*. Thus, we can talk about the *closure* of Σ in the context of age information.

Definition. For each DFD schema (U, Σ, Δ) and k in N, *the age-k closure of* Σ *under* Δ is the set $\Sigma_k^{\Delta} = \{f \in \mathrm{FD}(U) \mid \text{if } T \text{ is a set of tuples of age } k \text{ in some stable DBS in } \mathrm{SAT}(\Sigma, \Delta) \text{ then } T \models f\}$.

Thus, Σ_k^{Δ} is the set of all FDs satisfied by each set of tuples whose age is k in some stable DBS in $\mathrm{SAT}(\Sigma, \Delta)$.

We write Σ_k instead of Σ_k^{Δ} whenever Δ is understood.

It is easy to see that the definition is equivalent to the following:

Definition. For each DFD schema (U, Σ, Δ) and k in N, $\Sigma_k^{\Delta} = \{f \in \mathrm{FD}(U) \mid \text{if } \{(I_i, \mu_i)\}_{0 \le i \le k} \text{ is a stable DBS in } \mathrm{SAT}(\Sigma, \Delta) \text{ then } I_k \models f\}$.

The two definitions will be used interchangeably, as convenient.

The above definitions do not provide an effective way of computing age-k closure. Our next two results show how this can be done. The first (more theoretical in nature) leads to the second (more practical).[16]

The first method of computing age-k closure is analogous to Proposition 4.2 and requires introducing the notions of *action attributes*, *action constraints*, and *action relation* associated with database sequences. In order to do this, we need some notation for the versions of the attributes and constraints corresponding to each instance in the database sequence.

Notation. Let i be a nonnegative integer. For each attribute A, let A^i be a new attribute. Let f_i be the attribute isomorphism defined by $f_i(A) = A^i$ for each A. Let $U^i = f_i(U)$, $\Sigma^i = f_i(\Sigma)$, and $\Delta^i = \{f_i(X) \to f_{i+1}(Y) \mid \hat{X} \to \hat{Y} \in \Delta\} \cup \{f_{i+1}(X) \to f_i(Y) \mid \hat{X} \to \hat{Y} \in \Delta\}$. For each tuple x over U, let $x^i = f_i(x)$. Finally, for each instance I over U, let $I^i = \{x^i \mid x \in I\} = f_i(I)$.

Intuitively, A^i is the version of attribute A corresponding to the ith instance in a database sequence. Similarly, Σ^i is a 'copy' of the static constraints Σ, corresponding to the ith instance, and Δ^i is the version of Δ that relates the ith instance to the $(i+1)$th. Finally, x^i is a *copy* of a tuple x over U corresponding to the marked attributes U^i. The value vector of x^i is the same as the value vector of x.

We are now ready to define action attributes, action constraints and action relations. The first two are determined by a BDFD schema (U, Σ, Δ) and a database sequence. The third depends on the database sequence alone.

Definition. Let (U, Σ, Δ) be a BDFD schema and $s = \{(I_i, \mu_i)\}_{i \in S}$ a stable DBS in $\mathrm{SAT}(U, \Sigma, \Delta)$.

(a) The set of *action attributes associated with* (U, Σ, Δ) *and* s is[17] $U_s = \bigcup_{i \in S} U^i$.

(b) The set of *action constraints associated with* (U, Σ, Δ) *and* s is[17] $\Gamma_s = \bigcup_{i \in S} \Sigma^i \cup \bigcup_{i \in S_0} \Delta^i$.

(c) The *action relation associated with* s is $R_s = \{x^0 \bowtie \bowtie_{n \in S_0}((\mu_0 \circ \cdots \circ \mu_n)(x))^{n+1} \mid x \in I_0\}$.

Intuitively, the action relation is obtained by concatenating, in order, all successive versions of a tuple occurring in the database sequence. If the database sequence is infinite, there are infinitely many attributes involved.

[16] The first result, proven here for simplicity just for BDFDs, remains true for DFDs. The second result holds only for BDFDs.

[17] In order to simplify the notation, we use U_s and Γ_s to denote action attributes and constraints, although these also depend on (U, Σ, Δ).

A	B		A	B		A	B
0	0	→	0	0	→	0	1
1	1	→	1	0	→	1	1

FIGURE 5

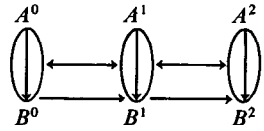


FIGURE 6

A^0	B^0	A^1	B^1	A^2	B^2
0	0	0	0	0	1
1	1	1	0	1	1

FIGURE 7

To illustrate the definition, let $U = AB$, $\Sigma = \{A \to B\}$, $\Delta = \{\check{B} \to \hat{B}\} \cup \Delta_U$, and s be the stable DBS of length 3 in SAT(U, Σ, Δ), shown in Figure 5.

The set of action attributes associated with (U, Σ, Δ) and s is $U_s = U^0U^1U^2$. The set of action constraints is $\Gamma_s = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Delta^0 \cup \Delta^1 = \{A^0 \to B^0, A^1 \to B^1, A^2 \to B^2, B^0 \to B^1, A^0B^0 \leftrightarrow A^1B^1, B^1 \to B^2, A^1B^1 \leftrightarrow A^2B^2\}$ and is represented in Figure 6.

Finally, the action relation associated with s is represented in Figure 7.

Clearly, the action attributes and action constraints are the same for all (U, Σ, Δ) and stable DBSs of fixed length. It will sometimes be convenient to refer to these action attributes and constraints. We therefore introduce the following:

Notation. For each BDFD schema (U, Σ, Δ) and $k \geq 0$, let $U(k) = \bigcup_{i=0}^{k} U^i$ and $\Gamma(\Sigma, \Delta, k) = \bigcup_{i=0}^{k} \Sigma^i \cup \bigcup_{i=0}^{k-1} \Delta^i$.

Thus $U(K)$ is the set of action attributes and $\Gamma(\Sigma, \Delta, k)$ the set of action constraints associated with all (U, Σ, Δ) and stable DBSs of length $k + 1$ over U.

Before turning to the first result of the section, we need one more piece of notation.

Notation. Let $s \in \text{SAT}(U, \Sigma, \Delta)$. For each $X \subseteq U_s$ let $\bar{X} = \{A \in U \mid A^i \in X$ for some $i \in S\}$. For each set $\Gamma \subseteq \text{FD}(U_s)$ let $\bar{\Gamma} = \{\bar{X} \to \bar{Y} \mid X \to Y \in \Gamma\}$. And for each tuple x over U^i, $i \in S$, let $\bar{x}$ be the tuple over U such that $\bar{x}(A) = x(A^i)$ for each $A \in U$.

As in our earlier notation, the symbol $^-$ acts on marked sets of attributes as an operator that erases the markers. The symbol $^-$ on tuples over U^i yields the same value vector but over the unmarked set of attributes U.

We now present the analog of Proposition 4.2 for age-k closure.

PROPOSITION 5.1. *For each BDFD schema* (U, Σ, Δ) *and each* $k \geq 0$, $\Sigma_k^{\Delta} = \overline{\Pi_{U^k}((\Gamma(\Sigma, \Delta, k))^*)}$.

PROOF. We first show that

(1) $\Sigma_k^{\Delta} \subseteq \overline{\Pi_{U^k}((\Gamma(\Sigma, \Delta, k))^*)}$.

To see this, it is enough to prove that

(2) if R is relation over $U(k)$ that satisfies $\Gamma(\Sigma, \Delta, k)$, then $\Pi_{U^k}(R) \models (\Sigma_k^{\Delta})^k$.

Indeed, this implies that $(\Sigma_k^{\Delta})^k \subseteq \Pi_{U^k}((\Gamma(\Sigma, \Delta, k))^*)$, whence $\Sigma_k^{\Delta} \subseteq \overline{\Pi_{U^k}((\Gamma(\Sigma, \Delta, k))^*)}$ by Proposition 3.3.

Let R be as in (2). For each i, $0 \leq i \leq k$, let $I_i = \overline{\Pi_{U^i}(R)}$. For each i, $0 \leq i < k$, let μ_i be the one-to-one mapping defined from I_i onto I_{i+1} by

$\mu_i(\overline{\Pi_{U^i}(u)}) = \overline{\Pi_{U^{i+1}}(u)}$ for each $u \in R$. (Indeed, μ_i is well defined because $R \models U^i \to U^{i+1}$, and it is one-to-one because $R \models U^{i+1} \to U^i$.) Finally, let $\mu_k = \varnothing$. Thus, $\{(I_i, \mu_i)\}_{0 \le i \le k}$ is a stable DBS of length $k+1$ over U. To prove (2), it is enough to show that

(3) $\{(I_i, \mu_i)\}_{0 \le i \le k} \models (\Sigma, \Delta)$.

Indeed, then $I_k \models \Sigma_k^\Delta$. But $I_k = \overline{\Pi_{U^k}(R)}$, so $\Pi_{U^k}(R) \models (\Sigma_k^\Delta)^k$ by Proposition 3.3.

Let i be an integer, $0 \le i \le k$. Since $R \models \Gamma(\Sigma, \Delta, k)$, $\Pi_{U^i}(R) \models \Pi_{U^i}((\Gamma(\Sigma, \Delta, k))^*)$. Thus, $I_i = \overline{\Pi_{U^i}(R)} \models \overline{\Pi_{U^i}((\Gamma(\Sigma, \Delta, k))^*)}$, by Proposition 3.3. Now

$$\begin{aligned}\Sigma^i &\subseteq \Pi_{U^i}(\Gamma(\Sigma, \Delta, k)) \\ &\subseteq \Pi_{U^i}((\Gamma(\Sigma, \Delta, k))^*),\end{aligned}$$

so

$$\Sigma \subseteq \overline{\Pi_{U^i}((\Gamma(\Sigma, \Delta, k))^*)}$$

by Proposition 3.3. Thus, $I_i \models \Sigma$.

To complete the proof of (3), it remains to show that $(I_i, \mu_i, I_{i+1}) \models \Delta$ for each i, $0 \le i < k$. Thus let i be an integer, $0 \le i < k$. Since $R \models \Gamma(\Sigma, \Delta, k)$,

$$\Pi_{U^iU^{i+1}}(R) \models \Pi_{U^iU^{i+1}}((\Gamma(\Sigma, \Delta, k))^*).$$

Now

$$\begin{aligned}\Delta^i &\subseteq \Pi_{U^iU^{i+1}}(\Gamma(\Sigma, \Delta, k)) \\ &\subseteq \Pi_{U^iU^{i+1}}((\Gamma(\Sigma, \Delta, k))^*),\end{aligned}$$

so $\Pi_{U^iU^{i+1}}(R) \models \Delta^i$. Let f be the attribute isomorphism from U^iU^{i+1} onto $\check{U}\hat{U}$ defined by $f(A^i) = \check{A}$ and $f(A^{i+1}) = \hat{A}$ for each $A \in U$. Then $f(\Pi_{U^iU^{i+1}}(R)) \models f(\Delta^i)$ by Proposition 3.3. From the definition of Δ^i, it is easily seen that $f(\Delta^i) = \Delta$. Also, $f(\Pi_{U^iU^{i+1}}(R)) = I_i\mu_iI_{i+1}$. Thus $I_i\mu_iI_{i+1} \models \Delta$, so $(I_i, \mu_i, I_{i+1}) \models \Delta$ and the proof of (3) is complete.

Consider the reverse implication of (1), that is,

(4) $\overline{\Pi_{U^k}((\Gamma(\Sigma, \Delta, k))^*)} \subseteq \Sigma_k^\Delta$.

To see this, let $s = \{(I_i, \mu_i)\}_{0 \le i \le k} \in \mathrm{SAT}(U, \Sigma, \Delta)$. Then R_s, the action relation associated with s, is over $U(k)$ and $R_s \models \Gamma(\Sigma, \Delta, k)$. Thus, $\Pi_{U^k}(R_s) \models \Pi_{U^k}((\Gamma(\Sigma, \Delta, k))^*)$. Since $I_k = \overline{\Pi_{U^k}(R_s)}$, $I_k \models \overline{\Pi_{U^k}((\Gamma(\Sigma, \Delta, k))^*)}$ by Proposition 3.3. Hence $\overline{\Pi_{U^k}((\Gamma(\Sigma, \Delta, k))^*)} \subseteq \Sigma_k^\Delta$. □

Although Proposition 5.1 provides an effective way of computing age-k closure, it is not one that would be very appealing in practice. Indeed, it is usually of interest to compute all[18] of the Σ_k^Δ. This involves an unknown, arbitrarily large number of attributes. Furthermore, the fact that Σ_i^Δ, $i < k$, may have been previously computed is of no use in computing Σ_k^Δ. Our next theorem yields a more practical way for computing age-k closure for BDFDs by establishing a recurrence relation between Σ_k^Δ and Σ_{k+1}^Δ. In order to obtain this result, we need some technical lemmas and definitions.

Definition. Let V and W be disjoint sets of attributes. A set Δ of FDs over VW is *bipartite with respect to V and W* if for each $X \to Y \in \Delta$, either $X \subseteq V$ and $Y \subseteq W$ or $X \subseteq W$ and $Y \subseteq V$.

In particular, if $\Delta \subseteq \mathrm{BDFD}(U)$, then Δ is bipartite with respect to $\check{U}$ and $\hat{U}$.

[18] It is shown later in the section that this is a finite process.

LEMMA 5.2. *Let V and W be disjoint, nonempty sets of attributes. Let $\Sigma_1 \subseteq FD(V)$, $\Sigma_2 \subseteq FD(W)$ and let $\Delta \subseteq FD(VW)$ be bipartite with respect to V and W. Then*[19]

$$\Pi_W((\Sigma_1 \cup \Delta \cup \Sigma_2)^*) = [\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^*)]^*.$$

PROOF. Since $\Sigma_2 \subseteq \Pi_W((\Sigma_1 \cup \Delta \cup \Sigma_2)^*)$ and $\Pi_W((\Sigma_1 \cup \Delta)^*) \subseteq \Pi_W((\Sigma_1 \cup \Delta \cup \Sigma_2)^*)$,

$$[\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^*)]^* \subseteq \Pi_W((\Sigma_1 \cup \Delta \cup \Sigma_2)^*).$$

To see the reverse inclusion, let $X \to Y \in \Pi_W((\Sigma_1 \cup \Delta \cup \Sigma_2)^*)$. We shall prove that

(*) $X \to Y \in [\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^*)]^*$.

By Proposition 2.5 there exist $n \geq 0$, $X^{(i)}, Y^{(i)}, Z^{(i)} \subseteq VW$, $0 \leq i \leq n$, such that $X^{(0)} = X$, $Y^{(0)} = \emptyset$, $X^{(n)} \supseteq Y$, $Z^{(n)} = \emptyset$, $Z^{(i)} \subseteq X^{(i)}$, $Z^{(i)} \to Y^{(i+1)} \in \Sigma_1 \cup \Delta \cup \Sigma_2$, and $X^{(i+1)} = X^{(i)}Y^{(i+1)}$ for each i, $0 \leq i < n$. Before verifying (*) it is necessary to show that

(**) $\Pi_W(X^{(i)}) \to \Pi_V(X^{(i)}) \in (\Sigma_1 \cup \Delta)^*$ whenever $\Pi_V(X^{(i)}) \neq \emptyset$, $0 \leq i \leq n$.

The proof of (**) is by induction on i. For $i = 0$, $X^{(0)} = X \subseteq W$, so $\Pi_V(X^{(0)}) = \emptyset$. Suppose the statement is true for i, $i \geq 0$. Consider $X^{(i+1)}$. If $\Pi_V(X^{(i+1)}) = \emptyset$, we are done. Suppose not. Now $X^{(i+1)} = X^{(i)}Y^{(i+1)}$, where $Z^{(i)} \to Y^{(i+1)} \in \Sigma_1 \cup \Delta \cup \Sigma_2$ and $Z^{(i)} \subseteq X^{(i)}$. Three cases arise:

(a) $Z^{(i)} \to Y^{(i+1)} \in \Sigma_2$. Then $\Pi_V(X^{(i)}) = \Pi_V(X^{(i+1)})$. Since $\Pi_W(X^{(i)}) \to \Pi_V(X^{(i)}) \in (\Sigma_1 \cup \Delta)^*$ (by the induction hypothesis) and $\Pi_W(X^{(i+1)}) \supseteq \Pi_W(X^{(i)})$, $\Pi_W(X^{(i+1)}) \to \Pi_V(X^{(i+1)}) \in (\Sigma_1 \cup \Delta)^*$.

(b) $Z^{(i)} \to Y^{(i+1)} \in \Sigma_1$. Then $Z^{(i)} \subseteq \Pi_V(X^{(i)})$ and $\Pi_V(X^{(i+1)}) = \Pi_V(X^{(i)}) \cup Y^{(i+1)}$. Hence, $\Pi_V(X^{(i)}) \to \Pi_V(X^{(i+1)}) \in \Sigma_1^*$. Also, $\Pi_W(X^{(i+1)}) = \Pi_W(X^{(i)})$ and $\Pi_W(X^{(i)}) \to \Pi_V(X^{(i)}) \in (\Sigma_1 \cup \Delta)^*$ (by the induction hypothesis). Thus, $\Pi_W(X^{(i+1)}) \to \Pi_V(X^{(i+1)}) \in (\Sigma_1 \cup \Delta)^*$.

(c) $Z^{(i)} \to Y^{(i+1)} \in \Delta$. Since Δ is bipartite, there are two possibilities. Suppose $Z^{(i)} \subseteq V$ and $Y^{(i+1)} \subseteq W$. Then $\Pi_V(X^{(i+1)}) = \Pi_V(X^{(i)})$. Since $\Pi_W(X^{(i)}) \to \Pi_V(X^{(i)}) \in (\Sigma_1 \cup \Delta)^*$ (by the induction hypothesis) and $\Pi_W(X^{(i+1)}) \subseteq \Pi_W(X^{(i)})$, $\Pi_W(X^{(i+1)}) \to \Pi_V(X^{(i)}) \in (\Sigma_1 \cup \Delta)^*$. Thus, $\Pi_W(X^{(i+1)}) \to \Pi_V(X^{(i+1)}) \in (\Sigma_1 \cup \Delta)^*$. Now suppose $Z^{(i)} \subseteq W$ and $Y^{(i+1)} \subseteq V$. Then $Z^{(i)} \subseteq \Pi_W(X^{(i)})$, so $\Pi_W(X^{(i)}) \to Y^{(i+1)} \in \Delta^*$. By the induction hypothesis, $\Pi_W(X^{(i)}) \to \Pi_V(X^{(i)}) \in (\Sigma_1 \cup \Delta)^*$. Thus, $\Pi_W(X^{(i)}) \to \Pi_V(X^{(i)}) \cup Y^{(i+1)} \in (\Sigma_1 \cup \Delta)^*$. Since $\Pi_W(X^{(i+1)}) \supseteq \Pi_W(X^{(i)})$ and $\Pi_V(X^{(i+1)}) = \Pi_V(X^{(i)}) \cup Y^{(i+1)}$, $\Pi_W(X^{(i+1)}) \to \Pi_V(X^{(i+1)}) \in (\Sigma_1 \cup \Delta)^*$.

In each of (a), (b), and (c), $\Pi_W(X^{(i+1)}) \to \Pi_V(X^{(i+1)}) \in (\Sigma_1 \cup \Delta)^*$. Thus, the induction is extended and the proof of (**) is complete.

Let i_j, $0 \leq j \leq k$, $0 \leq i_1 < i_2 < \cdots < i_k < n$, be those indices for which $Z^{(i_j)} \to Y^{(i_j+1)} \in \Sigma_2$. We prove (*) by induction on k.

Suppose $k = 0$. Then each $Z^{(i)} \to Y^{(i+1)}$ is in $\Sigma_1 \cup \Delta$, so $(X \to Y) = (X^{(0)} \to X^{(n)})$ is in $(\Sigma_1 \cup \Delta)^*$. Since $X \to Y \in \mathrm{FD}(W)$, $X \to Y \in \Pi_W((\Sigma_1 \cup \Delta)^*) \subseteq [\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^*)]^*$. Now suppose $k > 0$ and (*) is true for each j, $j < k$. Then $X^{(0)} \to \Pi_W(X^{(i_k)})$ satisfies the induction hypothesis and hence $X^{(0)} \to \Pi_W(X^{(i_j)}) \in [\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^*)]^*$. Also $Z^{(i_k)} \to Y^{(i_k+1)} \in \Sigma_2$, $Z^{(i_k)} \subseteq \Pi_W(X^{(i_k)})$ and $\Pi_W(X^{(i_k+1)}) = \Pi_W(X^{(i_k)}) \cup Y^{(i_k+1)}$ (since $X^{(i_k+1)} = X^{(i_k)}Y^{(i_k+1)}$ and $Y^{(i_k+1)} \subseteq W$). Hence, $\Pi_W(X^{(i_k)}) \to \Pi_W(X^{(i_k+1)}) \in \Sigma_2^*$. Thus, $X^{(0)} \to \Pi_W(X^{(i_k+1)}) \in [\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^*)]^*$. By (**),

[19] To be more precise, $\Pi_W((\Sigma_1 \cup \Delta \cup \Sigma_2)^{*(VW)}) = [\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^{*(VW)})]^{*W}$.

either $\Pi_V(X^{(i_k+1)}) = \emptyset$ or $\Pi_W(X^{(i_k+1)}) \to \Pi_V(X^{(i_k+1)}) \in (\Sigma_1 \cup \Delta)^*$. In either case, $\Pi_W(X^{(i_k+1)}) \to X^{(i_k+1)} \in (\Sigma_1 \cup \Delta)^*$. Since $Z^{(i)} \to Y^{(i+1)} \in \Sigma_1 \cup \Delta$ for $i_k < i \leq n$, $X^{(i_k+1)} \to X^{(n)} \in (\Sigma_1 \cup \Delta)^*$. Hence, $\Pi_W(X^{(i_k+1)}) \to X^{(n)} \in (\Sigma_1 \cup \Delta)^*$. It follows that $\Pi_W(X^{(i_k+1)}) \to \Pi_W(X^{(n)}) \in \Pi_W((\Sigma_1 \cup \Delta)^*)$. Thus, $X^{(0)} \to \Pi_W(X^{(n)}) \in [\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^*)]^*$. Since $Y \subseteq W$ and $Y \subseteq X^{(n)}$, $Y \subseteq \Pi_W(X^{(n)})$. Hence $(X \to Y) = (X^{(0)} \to Y) \in [\Sigma_2 \cup \Pi_W((\Sigma_1 \cup \Delta)^*)]^*$. □

From Lemma 5.2 (with $V = \check{U}$, $W = \hat{U}$, and $V = \hat{U}$, $W = \check{U}$) and Proposition 4.2, we get

LEMMA 5.3. *Let* $\Sigma_1, \Sigma_2 \subseteq \mathrm{FD}(U)$ *and* $\Delta \subseteq \mathrm{BDFD}(U)$. *Then*

(*a*) $\Pi_{\hat{U}}((\check{\Sigma}_1 \cup \Delta \cup \hat{\Sigma}_2)^*) = [\hat{\Sigma}_2 \cup \Pi_{\hat{U}}((\check{\Sigma}_1 \cup \Delta)^*)]^*$
$= [\hat{\Sigma}_2 \cup \widehat{on_\Delta(\Sigma_1)}]^*$,

(*b*) $\Pi_{\check{U}}((\check{\Sigma}_1 \cup \Delta \cup \hat{\Sigma}_2)^*) = [\check{\Sigma}_1 \cup \Pi_{\check{U}}((\hat{\Sigma}_2 \cup \Delta)^*)]^*$
$= [\check{\Sigma}_1 \cup \widehat{no_\Delta(\Sigma_2)}]^*$.

In our next lemma (as well as in Theorem 5.5) we consider sets of attributes and constraints that have a structure similar to action attributes and constraints associated with database sequences. (The lemma is easily proved using Lemma 5.2. The proof is omitted here but can be found in [25].) Specifically, we have

Definition. An *action sequence* is a sequence $\{(V^{(i)}, \Sigma^{(i)}, \Delta^{(i)})\}_{i \in S}$ with the following properties:

(a) S is an initial subset of N;
(b) $\{V^{(i)}\}_{i \in S}$ is a family of nonempty, pairwise disjoint sets of attributes;
(c) $\Sigma^{(i)} \subseteq \mathrm{FD}(V^{(i)})$ for each $i \in S$;
(d) $\Delta^{(i)} \subseteq \mathrm{FD}(V^{(i)}V^{(i+1)})$ and $\Delta^{(i)}$ is bipartite with respect to $V^{(i)}$ and $V^{(i+1)}$ for each $i \in S_0$;
(e) $\Delta^{(\max S)} = \emptyset$ if $\max S$ exists.

LEMMA 5.4. *Let* $\{(V^{(i)}, \Sigma^{(i)}, \Delta^{(i)})\}_{0 \leq i \leq 2}$ *be an action sequence of length* 3. *Then*

(*a*) $\Pi_{V^{(2)}}((\Sigma^{(0)} \cup \Delta^{(0)} \cup \Sigma^{(1)} \cup \Delta^{(1)} \cup \Sigma^{(2)})^*) = \Pi_{V^{(2)}}[(\Pi_{V^{(1)}}((\Sigma^{(0)} \cup \Delta^{(0)} \cup \Sigma^{(1)})^*) \cup \Delta^{(1)} \cup \Sigma^{(2)})^*]$,

(*b*) $\Pi_{V^{(0)}}((\Sigma^{(0)} \cup \Delta^{(0)} \cup \Sigma^{(1)} \cup \Delta^{(1)} \cup \Sigma^{(2)})^*) = \Pi_{V^{(0)}}[(\Sigma^{(0)} \cup \Delta^{(0)} \cup \Pi_{V^{(1)}}((\Sigma^{(1)} \cup \Delta^{(1)} \cup \Sigma^{(2)})^*))^*]$.

We are now ready to prove our second result on computing age-k closure. As mentioned earlier, the theorem establishes a recurrence relation between Σ_i^Δ and Σ_{i+1}^Δ.

THEOREM 5.5. *Let* (U, Σ, Δ) *be a BDFD schema. Then* $\Sigma_0^\Delta = \Sigma^*$, *and*

$$\Sigma_{i+1}^\Delta = [\Sigma_i^\Delta \cup on_\Delta(\Sigma_i^\Delta)]^* = [\Sigma \cup on_\Delta(\Sigma_i^\Delta)]^*$$

for each $i \geq 0$.

PROOF. Clearly, $\Sigma_0^\Delta = \Sigma^*$. We prove the recurrence relation directly, first for $i = 0$, and then for $i > 0$.

Suppose $i = 0$. By Proposition 5.1,

$$\Sigma_1^\Delta = \overline{\Pi_{U^1}((\Gamma(\Sigma, \Delta, 1))^*)} = \overline{\Pi_{U^1}((\Sigma^0 \cup \Delta^0 \cup \Sigma^1)^*)}.$$

Next, note that $\overline{\Pi_{U^1}((\Sigma^0 \cup \Delta^0 \cup \Sigma^1)^*)} = \overline{\Pi_{\check{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*)}$. (Indeed, let f be the attribute isomorphism from U^0U^1 onto $\check{U}\hat{U}$ defined by $f(A^0) = \check{A}$

and $f(A^1) = \hat{A}$ for each $A \in U$. Then $f((\Sigma^0 \cup \Delta^0 \cup \Sigma^1)^*) = (\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*$ and $f(\Pi_{U^1}((\Sigma^0 \cup \Delta^0 \cup \Sigma^1)^*)) = \Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*)$. Hence,

$$\overline{f(\Pi_{U^1}((\Sigma^0 \cup \Delta^0 \cup \Sigma^1)^*))} = \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*)}.$$

From the definition of $^-$ it is seen that

$$\overline{f(\Pi_{U^1}((\Sigma^0 \cup \Delta^0 \cup \Sigma^1)^*))} = \overline{\Pi_{U^1}((\Sigma^0 \cup \Delta^0 \cup \Sigma^1)^*)}.$$

Then

$$\begin{aligned}\Sigma_1^\Delta &= \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^*)} \\ &= (\Sigma \cup \mathrm{on}_\Delta(\Sigma))^* \quad \text{by Lemma 5.3 and Proposition 3.3} \\ &= (\Sigma_0^\Delta \cup \mathrm{on}_\Delta(\Sigma_0^\Delta))^* \\ &= (\Sigma \cup \mathrm{on}_\Delta(\Sigma_0^\Delta))^*.\end{aligned}$$

Now suppose $i > 0$. Consider the action sequence $\{(V^{(i)}, \Sigma^{(i)}, \Delta^{(i)})\}_{0 \le i \le 2}$, where

(1) $V^{(0)} = \bigcup_{0 \le j < i} U^j$, $V^{(1)} = U^i$, $V^{(2)} = U^{i+1}$;
(2) $\Sigma^{(0)} = \Gamma(\Sigma, \Delta, i-1)$, $\Sigma^{(1)} = \Sigma^{(i)}$, $\Sigma^{(2)} = \Sigma^{i+1}$;
(3) $\Delta^{(0)} = \Delta^{i-1}$ and $\Delta^{(1)} = \Delta^i$.

By Proposition 5.1,

$$\begin{aligned}\Sigma_{i+1}^\Delta &= \overline{\Pi_{U^{i+1}}((\Gamma(\Sigma, \Delta, i+1))^*)} \\ &= \overline{\Pi_{V^{(2)}}((\Sigma^{(0)} \cup \Delta^{(0)} \cup \Sigma^{(1)} \cup \Delta^{(1)} \cup \Sigma^{(2)})^*)} \\ &\qquad \text{by (2) and (3)} \\ &= \overline{\Pi_{V^{(2)}}[(\Pi_{V^{(1)}}((\Sigma^{(0)} \cup \Delta^{(0)} \cup \Sigma^{(1)})^*) \cup \Delta^{(1)} \cup \Sigma^{(2)})^*]}, \\ &\qquad \text{by Lemma 5.4 and Proposition 3.3} \\ &= \overline{\Pi_{U^{i+1}}[(\Pi_{U^i}((\Gamma(\Sigma, \Delta, i))^*) \cup \Delta^1 \cup \Sigma^{i+1})^*]} \\ &\qquad \text{by (1)–(3)} \\ &= \overline{\Pi_{U^{i+1}}[((\Sigma_i^\Delta)^i \cup \Delta^i \cup \Sigma^{i+1})^*]} \qquad \text{since } \Pi_{U^i}((\Gamma(\Sigma, \Delta, i))^*) = (\Sigma_i^\Delta)^i, \\ &\qquad \text{by Proposition 5.1 and Proposition 3.3} \\ &= \overline{[\Sigma^{i+1} \cup \Pi_{U^{i+1}}(((\Sigma_i^\Delta)^i \cup \Delta^i)^*)]^*} \qquad \text{by Lemma 5.2 and Proposition 3.3} \\ &= [\Sigma \cup \mathrm{on}_\Delta(\Sigma_i^\Delta)]^* \qquad \text{by Proposition 4.2.}\end{aligned}$$

Thus,

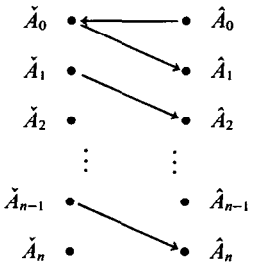
$$\begin{aligned}\Sigma_{i+1}^\Delta &= [\Sigma \cup \mathrm{on}_\Delta(\Sigma_i^\Delta)]^* \\ &\subseteq [\Sigma_i^\Delta \cup \mathrm{on}_\Delta(\Sigma_i^\Delta)]^* \quad \text{since} \quad \Sigma \subseteq \Sigma_i^\Delta \\ &\subseteq \Sigma_{i+1}^\Delta \quad \text{since} \quad \Sigma_i^\Delta \subseteq \Sigma_{i+1}^\Delta \quad \text{and} \quad \mathrm{on}_\Delta(\Sigma_i^\Delta) \subseteq \Sigma_{i+1}^\Delta.\end{aligned}$$

Hence, $\Sigma_{i+1}^\Delta = [\Sigma \cup \mathrm{on}_\Delta(\Sigma_i^\Delta)]^* = [\Sigma_i^\Delta \cup \mathrm{on}_\Delta(\Sigma_i^\Delta)]^*$. □

We continue our study of age in databases by looking at the sequence $\{\Sigma_k^\Delta\}_{k \ge 0}$ of successive age-k closures of Σ under Δ. It is shown next that age-k closures become constant after k has reached a certain value. Furthermore, the convergence is "rapid," in the sense that the consecutive age-k closures must increase at each step before reaching the "limit" $\tilde{\Sigma}$.

THEOREM 5.6. *For each BDFD schema* (U, Σ, Δ), *there exist an integer* $\tilde{k}(\Sigma, \Delta) \ge 0$ *and* $\tilde{\Sigma}^\Delta \subseteq FD(U)$ *(or* $\tilde{\Sigma}$ *when* Δ *is understood), such that* $\Sigma_k^\Delta \subsetneq \Sigma_{k+1}^\Delta$ *if* $0 \le k < \tilde{k}(\Sigma, \Delta)$ *and* $\Sigma_k^\Delta = \tilde{\Sigma}^\Delta$ *if* $k \ge \tilde{k}(\Sigma, \Delta)$.

FIGURE 8



PROOF. Since FD(U) is finite and $\Sigma_k^\Delta \subseteq \Sigma_{k+1}^\Delta \subseteq$ FD(U) for each $k \geq 0$, there exists $i \geq 0$ such that $\Sigma_i^\Delta = \Sigma_{i+1}^\Delta$. Let $\tilde{k}(\Sigma, \Delta)$ be the minimum such i, and let $\tilde{\Sigma} = \Sigma_{\tilde{k}(\Sigma,\Delta)}^\Delta$. By the minimality of $\tilde{k}(\Sigma, \Delta)$, $\Sigma_k^\Delta \subsetneq \Sigma_{k+1}^\Delta$ for each k, $0 \leq k < \tilde{k}(\Sigma, \Delta)$. From Theorem 5.5, it immediately follows that $\Sigma_k^\Delta = \tilde{\Sigma}$ for each k, $k \geq \tilde{k}(\Sigma, \Delta)$. □

Note that, in Example 3.1, $\tilde{k}(\{s_1\}, \{d_1, d_3\}) = 1$ and $\{\widetilde{s_1}\} = \{s_1, s_2\}^*$. In Example 3.2, $\tilde{k}(\{s_1\}, \{d_1, d_2\}) = 3$ and $\{\widetilde{s_1}\} = \{s_1, s_4\}^*$.

In a sense, one can say that a (stable) database becomes "mature" at age $\tilde{k}(\Sigma, \Delta)$. Indeed, no additional static constraints are acquired past that age as a result of getting "older." Clearly, $\tilde{k}(\Sigma, \Delta)$ is bounded by the number of FDs over U not in Σ^*. However, there is no uniform bound for $\tilde{k}(\Sigma, \Delta)$. Indeed, it is shown next that $\tilde{k}(\Sigma, \Delta)$ can be any integer. Thus, it can take arbitrarily long for a database to become mature.

PROPOSITION 5.7. *For each $n \geq 0$ there exists a BDFD schema (U, Σ, Δ) such that $\tilde{k}(\Sigma, \Delta) = n$.*

PROOF. Suppose n is an arbitrary positive integer. Let $U = \{A_i \mid 0 \leq i \leq n\}$, $\Sigma = \varnothing$ and $\Delta = \{\hat{A}_0 \to \check{A}_0\} \cup \{\check{A}_i \to \hat{A}_{i+1} \mid 0 \leq i < n\}$. ($\Delta$ is represented in Figure 8.) It is easily seen (using Theorem 5.5) that $\Sigma_0^\Delta = \varnothing^*$ and $\Sigma_1^\Delta = [\Sigma_0^\Delta \cup \text{on}_\Delta(\Sigma_0^\Delta)]^* = [\text{on}_\Delta(\varnothing)]^* = \{A_0 \to A_1\}^*$. A simple induction on i shows that $\Sigma_i^\Delta = \{A_0 \to A_0A_1 \cdots A_i\}^*$ for all i, $0 \leq i \leq n$. Finally, $\text{on}_\Delta(\Sigma_n^\Delta) = \Sigma_n^\Delta$, so $\Sigma_{n+1}^\Delta = [\Sigma_n^\Delta \cup \text{on}_\Delta(\Sigma_n^\Delta)]^* = [\Sigma_n^\Delta \cup \Sigma_n^\Delta]^* = \Sigma_n^\Delta$. Thus, $\Sigma_j^\Delta \subsetneq \Sigma_{j+1}^\Delta$ for each j, $0 \leq j < n$, and $\Sigma_n^\Delta = \Sigma_{n+1}^\Delta$. By Theorem 5.6, $n = \tilde{k}(\Sigma, \Delta)$. □

The set of static constraints satisfied by a mature database is $\tilde{\Sigma}$ (which may, in general, be larger than Σ^*). The satisfaction of some constraints in addition to Σ^* may itself be of use. Also, the set $\text{on}_\Delta(\tilde{\Sigma})$ of static constraints is *preserved* by a valid update (i.e., an update satisfying Δ) of a mature database. In particular, the set of constraints $\text{on}_\Delta(\tilde{\Sigma}) \cap \Sigma$ is "inherited" through a valid update. Thus fewer constraints need be checked in order to make sure the new instance satisfies Σ. The best one can expect is that $\text{on}_\Delta(\tilde{\Sigma}) \supseteq \Sigma$, in which case all of Σ is preserved. The worst is when $\text{on}_\Delta(\tilde{\Sigma}) \cap \Sigma = \varnothing^*$, in which case none of the static constraints are preserved through a valid update. Most cases fall somewhere between these extremes. The following illustrates several possible situations.

Examples

(a) $\text{on}_\Delta(\tilde{\Sigma}) \supseteq \Sigma$. Let $U = AB$, $\Sigma = \{A \to B\}$, and $\Delta = \{\hat{A} \to \check{A}, \check{B} \to \hat{B}\} \cup \Delta_U$ ($\tilde{\Sigma}$ and $\Delta - \Delta_U$ are shown in Figure 9). It is easily seen that $\tilde{k}(\Sigma, \Delta) = 0$, $\tilde{\Sigma} = \{A \to B\}^*$, and $\text{on}_\Delta(\tilde{\Sigma}) = \{A \to B\}^* = \Sigma^*$. In this case, all of Σ is preserved through a valid update.

(b) $\text{on}_\Delta(\tilde{\Sigma}) \neq \varnothing$, but $\text{on}_\Delta(\tilde{\Sigma}) \cap \Sigma = \varnothing$. Let $U = ABC$, $\Sigma = \{A \to B\}$, and $\Delta = \{\hat{A} \to \check{A}, \check{B} \to \hat{C}\} \cup \Delta_U$ ($\tilde{\Sigma}$ and $\Delta - \Delta_U$ are shown in Figure 10). It is easily

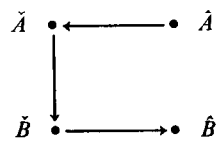


FIGURE 9

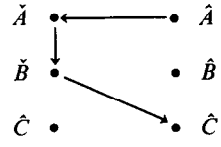


FIGURE 10

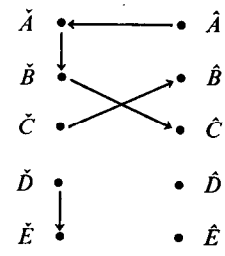


FIGURE 11

seen that $\tilde{k}(\Sigma, \Delta) = 1$, $\tilde{\Sigma} = \{A \to B, A \to C\}^*$, and $\text{on}_\Delta(\tilde{\Sigma}) = \{A \to C\}^*$. Thus, each set of tuples of age 1 satisfies the FD $A \to C$ in addition to Σ. However, none of the static constraints in Σ are preserved through valid updates.

(c) $\Sigma \subsetneq \tilde{\Sigma}$ and $\emptyset^* \subsetneq \text{on}_\Delta(\tilde{\Sigma}) \cap \Sigma \subseteq \Sigma$. Let $U = ABCDE$, $\Sigma = \{A \to B, D \to E\}$, and $\Delta = \{\hat{A} \to \check{A}, \check{B} \to \hat{C}, \check{C} \to \hat{B}\} \cup \Delta_U$ (Σ and $\Delta - \Delta_U$ are shown in Figure 11). It is easily verified that $\tilde{k}(\Sigma, \Delta) = 1$, $\tilde{\Sigma} = \{A \to B, D \to E, A \to C\}^*$ and $\text{on}_\Delta(\tilde{\Sigma}) = \{A \to B, A \to C\}^*$. Thus each set of tuples of age 1 satisfies the constraint $A \to C$ in addition to Σ. Also, a valid update preserves the static constraint $A \to B$ (but not $D \to E$) in Σ, provided that the tuples being updated have attained age 1 in the database. □

In the remainder of this section we focus primarily on the "limit" $\tilde{\Sigma}$ of a sequence of age-k closures. We first prove a few useful properties. Then we give a simple inference-rule-based mechanism for computing, in a straightforward manner, $\tilde{\Sigma}$ and $\text{on}_\Delta(\tilde{\Sigma})$ from Σ and Δ.

We first provide a useful characterization of $\tilde{\Sigma}$ which does not involve the notion of age closure. Indeed, we show that $\tilde{\Sigma}$ is the smallest (with respect to inclusion) closed set of FDs containing Σ and closed under on_Δ. First though, we prove that such a set exists.

LEMMA 5.8. *For each BDFD schema (U, Σ, Δ), there exists a unique minimal (with respect to inclusion) closed set Σ' of FDs over U such that $\Sigma \subseteq \Sigma'$ and $\text{on}_\Delta(\Sigma') \subseteq \Sigma'$.*

PROOF. Let $\mathscr{C} = \{\Gamma \subseteq \text{FD}(U) \mid \Sigma \subset \Gamma, \Gamma \text{ closed}, \text{on}_\Delta(\Gamma) \subseteq \Gamma\}$. Since $\text{FD}(U) \in \mathscr{C}$, $\mathscr{C}$ is not empty. Let $\Sigma' = \bigcap_{\Gamma \in \mathscr{C}} \Gamma$. Clearly, $\Sigma \subseteq \Sigma'$. Furthermore, for each $\Gamma \in \mathscr{C}$, $\text{on}_\Delta(\Sigma') \subseteq \text{on}_\Delta(\Gamma) \subseteq \Gamma$ since $\Sigma' \subseteq \Gamma$. Hence, $\text{on}_\Delta(\Sigma') \subseteq \bigcap_{\Gamma \in \mathscr{C}} \Gamma = \Sigma'$. Also, Σ' is minimal since $\Sigma' \subseteq \Gamma$ for each $\Gamma \in \mathscr{C}$. Since Σ' is an intersection of closed sets, it is closed. Obviously, Σ' is unique with respect to these properties. □

PROPOSITION 5.9. *For each BDFD schema (U, Σ, Δ), $\tilde{\Sigma}$ is the smallest closed set of FDs containing Σ and closed under on_Δ.*

PROOF. Let Σ' be the smallest closed set of FDs containing Σ and closed under on_Δ. By Lemma 5.8, Σ' exists. It suffices to show that $\Sigma' = \tilde{\Sigma}$. Clearly, $\Sigma' \subseteq \tilde{\Sigma}$. Also,

$$\begin{aligned} \text{on}_\Delta(\tilde{\Sigma}) &= \text{on}_\Delta(\Sigma^\Delta_{\tilde{k}(\Sigma,\Delta)}) \\ &\subseteq \Sigma^\Delta_{\tilde{k}(\Sigma,\Delta)+1} && \text{by Theorem 5.5} \\ &= \Sigma^\Delta_{\tilde{k}(\Sigma,\Delta)} && \text{by Theorem 5.6} \\ &= \tilde{\Sigma}, && \text{by Theorem 5.6 again.} \end{aligned}$$

Since $\Sigma \subseteq \tilde{\Sigma}$ and $\text{on}_\Delta(\tilde{\Sigma}) \subseteq \tilde{\Sigma}$, $\Sigma' \subseteq \tilde{\Sigma}$.

To prove the reverse inclusion, we shall show that

(*) $\Sigma_k^{\Delta} \subseteq \Sigma'$ for each $k \geq 0$.

Since $\tilde{\Sigma} = \Sigma^{\Delta}_{k(\Sigma,\Delta)}$, (*) implies that $\tilde{\Sigma} \subseteq \Sigma'$. The proof that $\Sigma_k^{\Delta} \subseteq \Sigma'$ for each $k \geq 0$ is by induction. For $k = 0$, $\Sigma_0^{\Delta} = \Sigma^* \subseteq \Sigma'$ by the definition of Σ'. Suppose $\Sigma_k^{\Delta} \subseteq \Sigma'$, $k \geq 0$. Then

$$\begin{aligned} \Sigma_{k+1}^{\Delta} &= [\Sigma_k^{\Delta} \cup \mathrm{on}_{\Delta}(\Sigma_k^{\Delta})]^* && \text{by Theorem 5.5} \\ &\subseteq [\Sigma' \cup \mathrm{on}_{\Delta}(\Sigma')]^* && \text{by the induction hypothesis} \\ &\subseteq \Sigma', && \text{since } \mathrm{on}_{\Delta}(\Sigma') \subseteq \Sigma' \text{ and } (\Sigma')^* = \Sigma'. \end{aligned}$$

Thus $\Sigma_{k+1}^{\Delta} \subseteq \Sigma'$, and the induction is extended. □

We now present our inference-rule-based mechanism for computing $\tilde{\Sigma}$ and $\mathrm{on}_{\Delta}(\tilde{\Sigma})$, which is, in some sense, sound and complete. The rules, which involve action attributes (i.e., attributes marked with $\check{}$ and $\hat{}$), consist of any sound and complete set of inference rules for FDs, together with one extra rule. The additional rule allows us to infer the FD $\check{X} \rightarrow \check{Y}$ over the "$\check{}$" attributes from the corresponding FD $\hat{X} \rightarrow \hat{Y}$ over the "$\hat{}$" attributes. This is formalized in the following:

Definition. The following is called a *$\tilde{\sigma}$-rule*:

For each $X, Y \subseteq U$, if $\hat{X} \rightarrow \hat{Y}$, then $\check{X} \rightarrow \check{Y}$.

Let $\mathscr{R}$ be a sound and complete set of inference rules for FDs over $\check{U}\hat{U}$. Let $\tilde{S}_{\mathscr{R}}$ be a new set of inference rules consisting of $\mathscr{R}$, together with the $\tilde{\sigma}$-rule. Clearly, the closure of a set F of FDs with respect to $\tilde{S}_{\mathscr{R}}$ is independent of the choice of $\mathscr{R}$.

Notation. For each $F \subseteq \mathrm{FD}(\check{U}\hat{U})$, let F^+ denote the closure of F with respect to $\tilde{S}_{\mathscr{R}}$.

The computation of $\tilde{\Sigma}$ from Σ and Δ is outlined below.

Algorithm AGE-CLOSURE.
Input: Σ, Δ.
Output: $\tilde{\Sigma}$.

(1) Set $F = \check{\Sigma} \cup \Delta \cup \hat{\Sigma}$.
(2) Compute F^+;
(3) Select all FDs over $\hat{U}$ that are in F^+;
(4) Remove all $\hat{}$ markers.

The output of Algorithm AGE-CLOSURE is the set $\overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}$. Theorem 5.13 shows that this is equal to $\tilde{\Sigma}$, thus proving the correctness of the algorithm.

The computation of $\mathrm{on}_{\Delta}(\tilde{\Sigma})$ from Σ and Δ is similar:

Algorithm ON.
Input: Σ, Δ
Output: $\mathrm{on}_{\Delta}(\tilde{\Sigma})$.

(1) Set $F = \check{\Sigma} \cup \Delta$;
(2) Compute F^+;
(3) Select all FDs over $\hat{U}$ which are in F^+;
(4) Remove all $\hat{}$ markers.

The output of Algorithm ON is the set of FDs $\overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta)^+)}$. Theorem 5.13 shows that this is equal to $\mathrm{on}_{\Delta}(\tilde{\Sigma})$.

Before proving Theorem 5.13, we need some notation and three technical results.

Notation. Let * denote the mapping from $2^{\mathrm{FD}(\check{U}\hat{U})}$ into $2^{\mathrm{FD}(\check{U}\hat{U})}$ defined by $*(F) = F^*$ for each $F \subseteq \mathrm{FD}(\check{U}\hat{U})$. Let t denote the mapping from $\mathrm{FD}(\check{U}\hat{U})$ into $2^{\mathrm{FD}(\check{U}\hat{U})}$

defined by $t(f) = \{f, \check{X} \to \check{Y}\}$ if $f = \hat{X} \to \hat{Y}$, and $t(f) = \{f\}$ otherwise. Extend t to a mapping from $2^{\mathrm{FD}(\check{U}\hat{U})}$ into $2^{\mathrm{FD}(\check{U}\hat{U})}$ by $t(F) = \bigcup_{f \in F} t(f)$.

It is easily seen (by induction on the length of derivations of FDs from F using the inference rules in $\tilde{S}_{\mathscr{R}}$) that $F^+ = \bigcup_{i \geq 0} (t \circ *)^i(F)$, for each $F \subseteq \mathrm{FD}(\check{U}\hat{U})$.

Our first technical lemma is

LEMMA 5.10. *Let* $\Sigma_1, \Sigma_2 \subseteq FD(U)$ *and* $\Delta \subseteq BDFD(U)$. *Then* $on_\Delta(\Sigma_1 \cup no_\Delta(\Sigma_2)) \subseteq (\Sigma_2 \cup on_\Delta(\Sigma_1))^*$.

PROOF. Clearly,

$$
\begin{aligned}
&\mathrm{on}_\Delta(\Sigma_1 \cup \mathrm{no}_\Delta(\Sigma_2)) \\
&\quad = \overline{\Pi_{\hat{U}}((\check{\Sigma}_1 \cup \check{\mathrm{no}}_\Delta(\Sigma_2) \cup \Delta)^*)} && \text{by Proposition 4.2} \\
&\quad = \overline{\Pi_{\hat{U}}((\check{\Sigma}_1 \cup \Pi_{\check{U}}((\hat{\Sigma}_2 \cup \Delta)^*) \cup \Delta)^*)} && \text{by Propositions 4.3 and 3.3} \\
&\quad \subseteq \overline{\Pi_{\hat{U}}((\check{\Sigma}_1 \cup \Delta \cup \hat{\Sigma}_2)^*)} \\
&\quad = (\Sigma_2 \cup \mathrm{on}_\Delta(\Sigma_1))^* && \text{by Lemma 5.3 and Proposition 3.3.}
\end{aligned}
$$

Thus, $\mathrm{on}_\Delta(\Sigma_1 \cup \mathrm{no}_\Delta(\Sigma_2)) \subseteq (\Sigma_2 \cup \mathrm{on}_\Delta(\Sigma_1))^*$. □

We proceed with the second technical lemma.

LEMMA 5.11. *Let* $F \subseteq FD(\check{U}\hat{U})$ *be closed with respect to* $+$. *Then* $\overline{\Pi_{\hat{U}}(F)} \subseteq \overline{\Pi_{\check{U}}(F)}$.

PROOF. Let $f \in \overline{\Pi_{\hat{U}}(F)}$. Then there exist $X, Y \subseteq U$ such that $f = X \to Y$ and $\hat{X} \to \hat{Y} \in F$. Since $F^+ = F$, $\check{X} \to \check{Y} \in F$ by the $\tilde{\sigma}$-rule. Hence, $\check{X} \to \check{Y} \in \Pi_{\check{U}}(F)$ and $f = X \to Y = \check{X} \to \check{Y} \in \overline{\Pi_{\check{U}}(F)}$. □

Finally, the third technical lemma is

LEMMA 5.12. *Let* $L, R \subseteq FD(U)$, $\Delta \subseteq BDFD(U)$, *and* $F = \check{L} \cup \Delta \cup \hat{R}$. *For each* $i \geq 0$ *let* $H_i = (t \circ *)^i(F)$, $R_i = \Pi_{\hat{U}}(H_i)$, *and* $L_i = \Pi_{\check{U}}(H_i)$. *Then*

$$R_{i+1} = (R_i \cup on_\Delta(R_i \cup L_i))^* \quad \text{and} \quad L_{i+1} = (L_i \cup R_i \cup no_\Delta(R_i))^*.$$

PROOF. Let $\Delta_i = H_i - (\check{L}_i \cup \hat{R}_i)$ for each $i \geq 0$. Thus, Δ_i consists of all FDs in H_i that are not in $\mathrm{FD}(\check{U})$ or in $\mathrm{FD}(\hat{U})$. We first prove the following statements for each $i \geq 0$:

(*) If $\Delta_i \subseteq (\check{L}_i \cup \Delta \cup \hat{R}_i)^*$, then $H_{i+1} = (\check{L}_i \cup \check{R}_i \cup \Delta \cup \hat{R}_i)^*$.

(**) $\Delta_i \subseteq (\check{L}_i \cup \Delta \cup \hat{R}_i)^*$.

Consider (*). Let $i \geq 0$ and suppose $\Delta_i \subseteq (\check{L}_i \cup \Delta \cup \hat{R}_i)^*$. Then

$$
\begin{aligned}
H_{i+1} &= (t \circ *)(H_i) \\
&= (t(\check{L}_i \cup \Delta_i \cup \hat{R}_i))^* \\
&= (\check{L}_i \cup \check{R}_i \cup \Delta_i \cup \hat{R}_i)^* \\
&= (\check{L}_i \cup \check{R}_i \cup (\check{L}_i \cup \Delta \cup \hat{R}_i)^* \cup \hat{R}_i)^* && \text{since } \Delta_i \subseteq (\check{L}_i \cup \Delta \cup \hat{R}_i)^* \text{ and } \Delta \subseteq \Delta_i \\
&= (\check{L}_i \cup \check{R}_i \cup \Delta \cup \hat{R}_i)^*.
\end{aligned}
$$

This establishes (*).

The proof of (**) is by induction on i. Suppose $i = 0$. Then $H_0 = F = \check{L} \cup \Delta \cup \hat{R}$. Thus, $\Delta_0 = \Delta$, $L_0 = L$, and $R_0 = R$, so (**) holds. Suppose $i \geq 0$

and $\Delta_i \subseteq (\check{L}_i \cup \Delta \cup \hat{R}_i)^*$. Then

$$\begin{aligned}\Delta_{i+1} &\subseteq H_{i+1}\\ &= (\check{L}_i \cup \check{R}_i \cup \Delta \cup \hat{R}_i)^* && \text{by induction and (*)}\\ &\subseteq (\check{L}_{i+1} \cup \Delta \cup \hat{R}_{i+1})^* && \text{since } \check{L}_i \cup \check{R}_i \subseteq \Pi_{\check{U}}(H_{i+1}) = \check{L}_{i+1}\\ & && \text{and } \hat{R}_i \subseteq \Pi_{\hat{U}}(H_{i+1}) = \hat{R}_{i+1}.\end{aligned}$$

Hence, the induction is extended and (**) is proved.

By (*) and (**), we have

(***) $H_{i+1} = (\check{L}_i \cup \check{R}_i \cup \Delta \cup \hat{R}_i)^*$ for each $i \geq 0$.

Let $i \geq 0$. Then

$$\begin{aligned}R_{i+1} &= \overline{\Pi_{\hat{U}}(H_{i+1})}\\ &= \overline{\Pi_{\hat{U}}((\check{L}_i \cup \check{R}_i \cup \Delta \cup \hat{R}_i)^*)} && \text{by (***)}\\ &= (R_i \cup \text{on}_\Delta(L_i \cup R_i))^* && \text{by Lemma 5.3 and Proposition 3.3.}\end{aligned}$$

Finally,

$$\begin{aligned}L_{i+1} &= \overline{\Pi_{\check{U}}(H_{i+1})}\\ &= \overline{\Pi_{\check{U}}((\check{L}_i \cup \check{R}_i \cup \Delta \cup \hat{R}_i)^*)} && \text{by (***)}\\ &= (L_i \cup R_i \cup \text{no}_\Delta(R_i))^* && \text{by Lemma 5.3 and Proposition 3.3.}\end{aligned}$$ □

We are now ready to show that $\tilde{\Sigma}$ and $\text{on}_\Delta(\tilde{\Sigma})$ are computed by Algorithm AGE-CLOSURE and Algorithm ON, respectively, as mentioned earlier.

THEOREM 5.13. *For each BDFD schema* (U, Σ, Δ),

(*a*) $\tilde{\Sigma} = \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}$,

(*b*) $on_\Delta(\tilde{\Sigma}) = \overline{\Pi_{\check{U}}((\check{\Sigma} \cup \Delta)^+)}$.

PROOF. We just prove (a), since (b) is similar. Consider

$$\tilde{\Sigma} \subseteq \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}.$$

By Proposition 5.9, it is enough to show that $\Sigma \subseteq \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}$ and

$$\text{on}_\Delta(\overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}) \subseteq \overline{\Pi_{\check{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}.$$

The first of the two last inclusions is obvious. For the second inclusion, note first that, by Lemma 5.11,

$$\overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)} \subseteq \overline{\Pi_{\check{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}.$$

Hence,

$$\begin{aligned}\text{on}_\Delta(\overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}) &\subseteq \text{on}_\Delta(\overline{\Pi_{\check{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)})\\ &= \overline{\Pi_{\hat{U}}((\Pi_{\check{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+) \cup \Delta)^*)} && \text{by Proposition 4.2,}\\ &\subseteq \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}.\end{aligned}$$

It follows that

$$\tilde{\Sigma} \subseteq \overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)}.$$

The reverse inclusion involves a proof by induction. As was remarked just prior to Lemma 5.10,

$$(\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+ = \bigcup_{i \geq 0} (t \circ *)^i(\check{\Sigma} \cup \Delta \cup \hat{\Sigma}).$$

Hence,

$$\overline{\Pi_{\hat{U}}((\check{\Sigma} \cup \Delta \cup \hat{\Sigma})^+)} = \bigcup_{i \geq 0} \overline{\Pi_{\hat{U}}((t \circ *)^i(\check{\Sigma} \cup \Delta \cup \hat{\Sigma}))}.$$

To complete the proof of (a), it is thus sufficient to prove that

(*) $\overline{\Pi_{\hat{U}}((t \circ *)^i(\check{\Sigma} \cup \Delta \cup \hat{\Sigma}))} \subseteq \tilde{\Sigma}$, for each $i \geq 0$.

Let $H_i = (t \circ *)^i(\check{\Sigma} \cup \Delta \cup \hat{\Sigma})$, $L_i = \overline{\Pi_{\hat{U}}(H_i)}$, and $R_i = \overline{\Pi_{\hat{U}}(H_i)}$. In order to show (*) we establish the following statement:

(**) For each $i \geq 0$,

(1) $R_i \subseteq \tilde{\Sigma}$,
(2) $L_i \subseteq (\tilde{\Sigma} \cup \text{no}_\Delta(\tilde{\Sigma}))^*$.

Note that (1) of (**) is the same as (*). We use induction on i. For $i = 0$, $R_0 = \Sigma$ and $L_0 = \Sigma$. Hence, (**) holds. Now suppose (**) is true for $i \geq 0$. By Lemma 5.12 (with $L = R = \Sigma$),

$$\begin{aligned}
&R_{i+1} \\
&\quad = (R_i \cup \text{on}_\Delta(R_i \cup L_i))^* \\
&\quad \subseteq (\tilde{\Sigma} \cup \text{on}_\Delta(\tilde{\Sigma} \cup (\tilde{\Sigma} \cup \text{no}_\Delta(\tilde{\Sigma}))^*))^* && \text{by the induction hypothesis for (1) and (2)} \\
&\quad = (\tilde{\Sigma} \cup \text{on}_\Delta((\tilde{\Sigma} \cup \text{no}_\Delta(\tilde{\Sigma}))^*))^* \\
&\quad = (\tilde{\Sigma} \cup \text{on}_\Delta(\tilde{\Sigma} \cup \text{no}_\Delta(\tilde{\Sigma})))^* && \text{by Proposition 4.1} \\
&\quad \subseteq (\tilde{\Sigma} \cup (\tilde{\Sigma} \cup \text{on}_\Delta(\tilde{\Sigma}))^*)^* && \text{by Lemma 5.10} \\
&\quad = (\tilde{\Sigma} \cup \text{on}_\Delta(\tilde{\Sigma}))^* \\
&\quad = \tilde{\Sigma}, \text{ since } \text{on}_\Delta(\tilde{\Sigma}) \subseteq \tilde{\Sigma} && \text{by Proposition 5.9.}
\end{aligned}$$

Thus, the induction is extended for (1). Consider (2). By Lemma 5.12 (with $L = R = \Sigma$),

$$\begin{aligned}
L_{i+1} &= (L_i \cup R_i \cup \text{no}_\Delta(R_i))^* \\
&\subseteq ((\tilde{\Sigma} \cup \text{no}_\Delta(\tilde{\Sigma}))^* \cup (\tilde{\Sigma} \cup \text{no}_\Delta(\tilde{\Sigma}))^* && \text{by the induction hypothesis for (**)} \\
&= (\tilde{\Sigma} \cup \text{no}_\Delta(\tilde{\Sigma}))^*.
\end{aligned}$$

Thus, the induction is also extended for (2), and (**) is established. In particular, (*) is proved, and the argument for (a) is complete. □

Remark. Let $T_{\mathscr{R}}$ be the set of inference rules consisting of $\mathscr{R}$, together with the two rules

for each $X, Y \subseteq U$, if $X \rightarrow Y$, then $\check{X} \rightarrow \check{Y}$,

and

for each $X, Y \subseteq U$, if $\hat{X} \rightarrow \hat{Y}$, then $X \rightarrow Y$.

Using Theorem 5.13, it can be easily verified that $\tilde{\Sigma}$ is the closure of $\Sigma \cup \Delta$ under the set of inference rules $T_{\mathscr{R}}$ (with target domain FD(U)). Similarly, $\text{on}_\Delta(\tilde{\Sigma})$ is the closure of $\check{\Sigma} \cup \Delta$ under $T_{\mathscr{R}}$ (with target domain FD(U)). Thus we might

intuitively say that the set of inference rules $T_{\mathscr{R}}$ (with target domain FD(U)) is sound and complete.

The following two methods are now available for computing $\tilde{\Sigma}$:

(1) Compute the age closure Σ_i^{Δ} for each i, $1 \leq i \leq \tilde{k}(\Sigma, \Delta)$, using the recurrence relation between successive age-k closures (Theorems 5.5 and 5.6).
(2) Compute $\tilde{\Sigma}$ directly from Σ and Δ, using Theorem 5.13.

For the purpose of comparing (1) and (2), consider the forest where Σ is the set of roots and $\text{on}_{\Delta}(f)$ the set of children of f. Intuitively speaking, computing $\tilde{\Sigma}$ according to (1) results in a breadth-first traversal of the forest. On the other hand, (2) not only permits a simulation of (1) but also allows a variety of alternative computing strategies. For example, (2) allows depth-first traversals of the forest, some of which are more efficient in certain cases.

We conclude the section by showing how the notion of age extends to arbitrary database sequences. As in the case of stable databases, the age of a tuple is the number of updates to which the tuple has been subjected since having been first inserted in the database. Formally, we have

Definition. Let $s = \{(I_i, \mu_i)\}_{i \in S}$ be a DBS and u a tuple in I_j, $j \in S$. The occurrence of u in I_j is said to have *age* k in s if there exist $u_0, \ldots, u_k$ such that

(a) $u_k = u$,
(b) $u_n \in I_{j-k+n}$, $\quad 0 \leq n \leq k$;
(c) $\mu_{j-k+n}(u_n) = u_{n+1}$, $\quad 0 \leq n < k$;
(d) if $j - k - 1 \geq 0$, then $u_0 \notin \text{range } \mu_{j-k-1}$.

In the special case of stable databases, the above reduces to our previous definition. (Observe that if two tuple occurrences have the same age in a stable DBS s, then they must belong to a common instance of the sequence s. This is not the case for arbitrary DBSs.)

There is a straightforward connection between age in arbitrary databases and age in stable databases. Indeed, from our definitions, we easily obtain (proof omitted)

PROPOSITION 5.14. *Let (U, Σ, Δ) be a BDFD schema and T a set of tuples over U. For each $k \geq 0$, the following are equivalent:*

(a) There exists a DBS $s \in SAT(U, \Sigma, \Delta)$ such that all tuples in T occur in a common instance of s and (the occurrences) have age k in s.
(b) There exists a stable DBS $s \in SAT(U, \Sigma, \Delta)$ such that each tuple in T has age k in s.

It is now apparent how to extend our results on age closure to arbitrary databases. Indeed, from Proposition 5.14 we have (proof omitted)

THEOREM 5.15. *For each BDFD schema (U, Σ, Δ) and $k \geq 0$, Σ_k^{Δ} is the set of all $f \in FD(U)$ satisfied by each set T of tuples over U for which there exists a DBS $\in SAT(U, \Sigma, \Delta)$ such that all tuples in T occur in a common instance of s and (the occurrences) have age k in s.*

In other words, Σ_k^{Δ} is the set of all FDs satisfied by each *subinstance* of a database satisfying (Σ, Δ), whose tuples have age k in the database. Thus, static constraints can be inferred from the evolution of the database even when insertions and deletions are allowed. In certain cases we can only infer satisfaction of static constraints by a subinstance I of the current state J (e.g., newly inserted tuples in

J may invalidate certain static constraints that must be satisfied by older tuples in I). Substantial advantages can still be derived in the latter case. For instance, suppose we know (from age information) that I satisfies the FD f, and we have to check that $J = I \cup L$ satisfies f. Clearly, the fact that $I \models f$ facilitates checking that $J \models f$. The improvement in performance depends on the sizes of J and L and on the specific algorithm used to test satisfaction of FDs. For instance, suppose J contains n tuples and L contains $g(n)$ tuples. Also, suppose that the algorithm used to test satisfaction of FDs is based on the "naive" method of checking that each set of two tuples in the instance satisfies f. Then it can easily be seen, that checking directly, whether $J \models f$ takes $O(n^2)$ comparisons. On the other hand, checking whether $J \models f$, when it is known that $I \models f$, takes $O(n\ g(n))$ comparisons. This improvement is considerable if the number $g(n)$ of "young" tuples in L is small relative to the size n of the database. For example, if $g(n) \leq \log n$, then the improvement is from $O(n^2)$ to $O(n \log n)$.

6. *Survivability in a Database*

In Section 5 we saw how to use knowledge about the history of a database to gain information about the current state. In this section we summarize several results of [25] concerning the connection between the current state and the *future* evolution of the database. The relevant concept here is that of "survivability" in a database. Informally, a set T of tuples over U has *survivability* k if the tuples can be validly updated together k times. In the case in which $k = \infty$, T is said to be *potentially immortal.*

As in the case of age, in order for a set of tuples to have survivability k in a database, it is sometimes necessary that the set of such tuples satisfy static constraints in addition to Σ^*. This leads to the notion of survivability-k closure of Σ under Δ, denoted $\Sigma^{\underline{\Delta}}_k$. This notion is symmetric to that of age (for k finite). The symmetry is formalized by the following result:[20]

PROPOSITION 6.1. *For each BDFD schema* (U, Σ, Δ) *and each* $k \geq 0$, $\Sigma^{\underline{\Delta}}_k = \Sigma^{g^{(\Delta)}}_k$.

In view of the above result, we can use most of our results on age-k closure to obtain analogous results on survivability-k closure (for k finite). (In particular, the limit of the sequence $\{\Sigma^{\underline{\Delta}}_k\}_{k \geq 0}$ is denoted by $\tilde{\tilde{\Sigma}}^{\underline{\Delta}}$.) These results can be found in [25].

The most notable breach in the past–future symmetry arises from the fact that, although we assume that any database has a start in time, there is no reason to assume it will not go on forever. Hence there is no analog of potential immortality (or survivability ∞) for age. Most of the results concerning survivability focus, therefore, on potential immortality. We first look at the relation between the survivability-∞ closure ($\Sigma^{\underline{\Delta}}_{\infty}$) and the survivability-$k$ closures, for k finite.

We obtain the following "convergence" result:

THEOREM 6.2. *For each BDFD schema* (U, Σ, Δ),

$$\Sigma^{\underline{\Delta}}_{\infty} = \bigcup_{k \geq 0} \Sigma^{\underline{\Delta}}_k = \tilde{\tilde{\Sigma}}^{\underline{\Delta}}.$$

Given a valid instance I of a database, it is sometimes of interest to know whether I can be validly updated a given k number of times. (In fact, it may often

[20] Recall that σ was defined prior to the Symmetry Proposition (4.4).

be reasonable to require that I be potentially immortal, i.e., $k = \infty$.) Satisfaction of $\Sigma \stackrel{\Delta}{=}_k$ by I is a necessary condition in order that I have survivability k. However, it is not a sufficient condition (an example is provided in [25]). The question arises whether it is even decidable if an instance has survivability k in a database. For k finite the problem is clearly decidable. We show that the problem is also decidable for $k = \infty$ by exhibiting a connection between finite survivability and potential immortality.

THEOREM 6.3. *Let* (U, Σ, Δ) *be a BDFD schema and I an instance over U. If I has survivability* $(2^{|U|})^{|I|^2}$, *then I is potentially immortal.*

In [25], a characterization is given of the *BDFD* schemas (U, Σ, Δ) for which *every* (valid) instance of the database is potentially immortal.

Finally, the combined effect of age and survivability on the static constraints of a database is investigated. The main result is, intuitively, that the combined effect of age and survivability is "additive."

7. *Conclusions*

In this paper, a simple model (database sequences) was presented for the evolution of databases in time. This model was then used to study the effect on database evolution of a certain type of dynamic constraint (DFDs). The interaction between DFDs and FDs was investigated using the notion of age closure.

Clearly, this paper is just a first step in studying the effect of dynamic constraints on database evolution. Future investigations can involve either our "dynamic" framework, or extensions of our model and constraints. Several important questions can be addressed within the present framework, in addition to those already discussed. For instance, how do DFDs interact with join dependencies [2]? What can be said about the usual normal forms in the presence of dynamic FDs? And, how should this affect database design?

Both the model and constraints can be extended in several directions. Just as FDs give rise to DFDs, so each type δ of static constraint gives rise to a dynamic analog $d\delta$. Our dynamic constraints on updates can be augmented with constraints on insertions, deletions, and individual tuple updates. Constraints involving more than two consecutive instances of the database and dynamic constraints that change in time can be considered. Also, dynamic constraints that involve nonrelational database models (e.g., the IFO Model [1]) or are model independent (in the spirit of Spyratos [23]) can be defined, etc. Questions analogous to those raised in this paper can be asked about any of the above extensions, and new questions are likely to arise. Clearly, much work remains to be done.

ACKNOWLEDGMENTS. I am grateful to my advisor, Seymour Ginsburg, for providing guidance and support while I wrote my Ph.D. dissertation, on which this paper is based. Special thanks to Richard Hull for numerous discussions and very useful suggestions related to the dissertation and this paper.

REFERENCES

1. ABITEBOUL, S., AND HULL, R. IFO: A formal semantic database model. Tech. Rep. TR-84-304, Dept. of Computer Science, Univ. of Southern California, Apr. 1984.
2. AHO, A. V., BEERI, C., AND ULLMAN, J. D. The theory of joins in relational databases. *ACM Trans. Database Syst. 4*, 3 (Sept. 1979), 297–314.
3. ARMSTRONG, V. W. Dependency structures of database relationships. In *Proceedings of IFIP '74*. North-Holland, Amsterdam, 1974, pp. 580–583.

4. BEERI, C., FAGIN, R., AND HOWARD, J. H. A complete axiomatization for functional and multivalued dependencies in database relations. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data* (Toronto, Ont., Canada, Aug. 3–5). ACM, New York, 1977, pp. 47–61.
5. BEERI, C., DOWD, M., FAGIN, R., AND STATMAN, R. On the structure of Armstrong relations for functional dependencies. IBM Res. Rep. RJ290. IBM, San Jose, Calif., Sept. 1980.
6. BRODIE, M. On modelling behavioral semantics of databases. In *Proceedings of the 7th International Conference on Very Large Data Bases.* ACM, New York, 1981, pp. 32–42.
7. CASANOVA, M. A., AND FURTADO, A. L. A family of temporal languages for the description of transition constraints. In *Advances in Database Theory*, vol. 2, H. Gallaire, J. Minker, and J.-M. Nicolas, Eds. Plenum Press, New York, 1985.
8. CASTILHO, I. M. V., CASANOVA, M. A., AND FURTADO, A. L. A temporal framework for database specifications. In *Proceedings of the 8th International Conference on Very Large Data Bases.* ACM, New York, 1982, pp. 280–291.
9. CERI, S., PELAGATTI, G., AND BRACCHI, G. Structured methodology for designing static and dynamic aspects of database applications. *Inf. Syst. 6*, 1 (1981), 31–45.
10. CLIFFORD, J., AND WARREN, D. S. Formal semantics for time in databases. *ACM Trans. Database Syst. 8*, 2 (June 1983), 214–254.
11. CODD, E. F. Further normalization of the database relational model. In *Courant Computer Science C0.Symposia 6: Database Systems,* Prentice-Hall, Englewood Cliffs, N.J., 1971, pp. 33–64.
12. DARWIN, C. *The Evolution of Species*, P. F. Collier & Son, New York, 1902.
13. DE ANTONELLIS, V., AND ZONTA, B. Modelling events in database applications design. In *Proceedings of the 7th International Conference on Very Large Data Bases.* ACM, New York, 1981, pp. 23–31.
14. FAGIN, R. Functional dependencies in a relational database and propositional logic. *IBM J. Res. Dev. 21*, 6 (Nov. 1977), 534–544.
15. FAGIN, R. Horn clauses and database dependencies. *J. ACM 29*, 4 (Oct. 1982), 952–985.
16. GALLAIRE, H. Impacts of logic on databases. In *Proceedings of the 7th International Conference on Very Large Data Bases.* ACM, New York, 1981, pp. 248–259.
17. GINSBURG, S., AND HULL, R. Characterizations for functional dependency and Boyce–Codd normal form families. *J. Theoret. Comput. Sci. 26*, 3 (1983), 243–286.
18. GINSBURG, S., AND ZAIDDAN, S. M. Properties of functional dependency families. *J. ACM 29*, 3 (July 1982), 678–698.
19. HAMMER, M., AND MCLEOD, D. J. Semantic integrity in a relational data base system. In *Proceedings of the 1st International Conference on Very Large Data Bases.* ACM, New York, 1975.
20. HULL, R. Finitely specifiable implicational dependency families. *J. ACM 31*, 2 (Apr. 1984), 210–226.
21. MAIER, D. *The Theory of Relational Databases.* Computer Science Press, Rockville, Md., 1983.
22. NICOLAS, J. M., AND YAZDANIAN, N. Integrity checking in deductive databases. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York, 1978.
23. SPYRATOS, N. An operational approach to data bases. In *Proceedings of the ACM Symposium on Principles of Database Systems.* ACM, New York, 1982, pp. 212–220.
24. ULLMAN, J. *Principles of Database Systems.* Computer Science Press, Rockville, Md., 1980.
25. VIANU, V. Dynamic constraints and database evolution. Ph.D. dissertation. Dept. of Computer Science, Univ. of Southern California, Jan. 1983.

RECEIVED NOVEMBER 1983; REVISED JUNE 1985; ACCEPTED JUNE 1986