ACHIEVING ZERO INFORMATION-LOSS IN A CLASSICAL DATABASE ENVIRONMENT*

Gautam Bhargava
Shashi K. Gadia

Computer Science Department
Iowa State University
Ames, IA 50011

Abstract. The research in temporal databases has, so far, concentrated on the history of an object as it exists in the real world. Instead, in this paper we view the history of an object as it is recorded in a database. Such a history is obtained by extrapolating the outcome of the updates (insert, modify, and delete) made to the object at discrete instants. Our model supports two kinds of query-users: the *system-user*, and the *classical-user*. For the classical-user, the interface to the database is identical to the usual interface in classical snapshot databases. We extend the classical relational model so that a transaction, i.e., an update or a (retrieval) query, is recorded in such a way that its effect can be determined at any time in the future; thus, our model is a *zero information-loss model* (Theorem 1). The logical structure imposed upon the model allows us to give a powerful algebra for the system-user to query the circumstantial information surrounding updates and queries. In addition, a single execution of a query can be identified with the relation it retrieves; thus, a user can query queries, query queries on queries, ad infinitum. The model represents an application of temporal databases to mainstream databases. It can be used in auditing, and as a foundation for building secure systems.

1. INTRODUCTION.

An update operation in classical databases is destructive - it not only destroys the environment in which it is executed, but also destroys the environment for queries. After an update is made, only the new database state is available, without even a clue to its past states.

A transaction in a database system is either an update, or a query. The activities in a database system consist of a sequence T of such transactions. We present a model, called the *zero information-loss model*, in which no information is ever lost - not even the circumstantial information surrounding the transactions. Transactions in our model are recorded by imposing a logical structure on T in such a way that the precise effect of any transaction, as well as the transaction itself, can be determined at any time in the future. In addition, the logical structure imposed upon transactions gives rise to a simple, yet powerful, algebra.

We use certain temporal relations, called transaction history relations, in which we associate a timestamp with every data value. Such relations form a foundation for the zero information-loss model. We use the time universe [0,NOW] to model the past and present (NOW) history of data in a uniform manner. As opposed to the usual approach of looking at the time dimension as the real world time, we consider it as the *transaction time* and use it to model the changing values of object attributes in the database.

1.1. Example. Consider the data value taken by the attribute COLOR, as shown in Figure 1.1. The semantics of this data value are: "from transaction time 15 to 20, the value of COLOR was known to be 'red' in the database, and from 21 to 25 the value was known to be 'blue'." Thus, for example, a user querying the color at transaction time 24 would retrieve 'blue' as its value. From this data value we can infer that the COLOR object was created at transaction time 15, modified at transaction time 21, and deleted at transaction time 26. This illustrates how the temporal relations in our model can be used to "store" update operations.

COLOR
[15,20] red [21,25] blue

Figure 1.1. A timestamped data value of COLOR

*This work was supported in part by the National Science Foundation under grant IRI-8810704.

Proceedings of the Fifteenth International
Conference on Very Large Data Bases

Amsterdam, 1989

1.2. Definition of the zero information-loss model. Let $\mathcal{R}$ be a database scheme. Then, a *zero information-loss* model over $\mathcal{R}$ is a triple $\langle\langle\mathcal{D}$, Shadow$\rangle$, Q-log$\rangle$, where,

- $\mathcal{D}$ is the *basic model*. For every $R \in \mathcal{R}$ there is a transaction history relation r(R) $\in \mathcal{D}$, and a relation shadow(r) $\in$ Shadow.
- Q-log is a single relation.

The pair $\langle\mathcal{D}$,Shadow$\rangle$ can be viewed as a logically structured log of all updates. In a database system, the role of an update is to change the state of a database by modifying certain data values. Also, an update operation is associated with certain environmental parameters - the person authorizing the update, the reason for the update, the time of the update, etc. Along these lines, an update u to a relation r in our model can be decomposed into two parts: u^1, consisting of the new data value, and u^2, consisting of the circumstantial information surrounding the update u. The effects of u^1 and u^2 are reflected in r and shadow(r), respectively. This decomposition of the data and the circumstances of u is not lossy: the key of the updated tuple, and the time of the update can be used to glue them back together. The reason for separating r and shadow(r) is that their structures are very different. This separation allows us to maximally harness the potential of the relational approach for querying them.

The Q-log is a single relation used for storing a log of all the queries, together with the information about their execution environments. With Q-logs not only can we look at the details of (or, even re-execute) past queries, but also query queries, query queries on queries, ad infinitum. This extended capability has enormous potential in applications where it is important to monitor the accesses made to the database.

1.3. A comprehensive example. Suppose we have an emp transaction history relation with NAME, SALARY, and DEPT attributes, with NAME as the key, and that the shadow information corresponding to any update in the emp relation includes the transaction time (TT), the AUTHORIZER, the USER, and the REASON for the update. Now, consider the following sequence of transactions in the database:

T1: TT = 8; USER = Mark.
insert (NAME:John; SALARY:15K; DEPT:Toys) *in* emp
with (AUTHORIZER=Don; REASON=New Employee);

T2: TT = 40; USER = Ryne.
modify (NAME: John) *to* (DEPT: Shoes; SALARY: 20K)
in emp
with (AUTHORIZER = Don; REASON = Reassignment);

T3: TT = 42; USER = Vance.
Q1: What is John's salary?

T4: TT = 48; USER = Rick.
insert (NAME: Doug; SALARY: 20K; DEPT: Auto)
in emp
with (AUTHORIZER = Joe; REASON = New Employee);

T5: TT = 53; USER = Damon.
delete (NAME: John) *in* emp
with (AUTHORIZER = Don; REASON = Fired);

T6: TT = 54; USER = Andre.
Q1: What is John's salary?

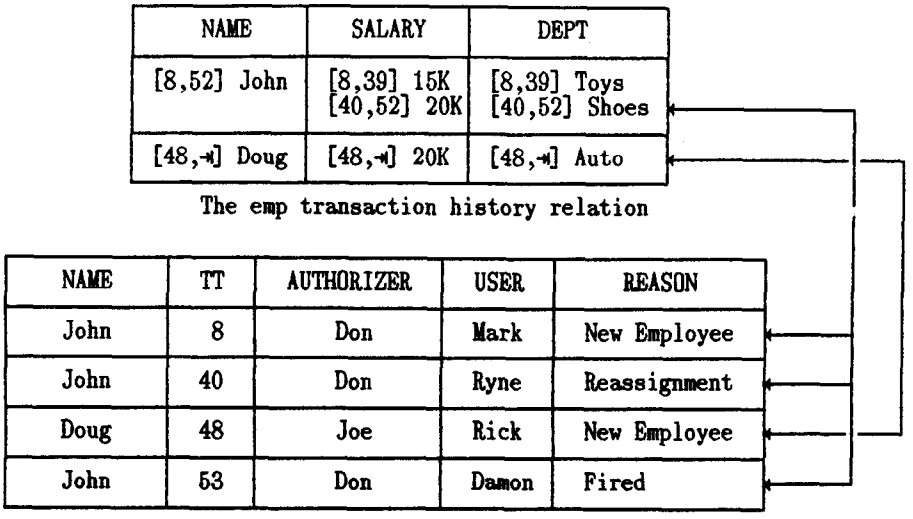
T7: TT = 55; USER = Mitch.
Q2: What is John's department?

T8: TT = 56; USER = Don.
Q3: Who made inquiries about John's salary?

Corresponding to these transactions, the instance of our model $\langle\langle\mathcal{D}$, Shadow$\rangle$, Q-log$\rangle$ is shown in Figure 1.2. Note that the symbol $\dashv$ represents NOW.

Note that the values of NAME in emp and shadow(emp) set up a logical correspondence between tuples of emp and shadow(emp). By including the transaction time, this correspondence can be refined to a one-to-one correspondence between all updates to emp and the tuples in shadow(emp). As a result, the transactions {T1, T2, T4, T5} can be completely restored from the current state of emp and shadow(emp), and {T3, T6, T7, T8} can be completely restored from emp and the Q-log. Now, using TT, the transaction time, we can order {T1, T2, T4, T5} U {T3, T6, T7, T8} to obtain the original sequence of transactions. Theorem 1 (the Zero-loss Theorem) says that this is true in general, i.e., a transaction log can be restored from the current state of the zero information-loss model; thus, indeed, there is no loss of information in our model.

The remainder of this paper is organised as follows. The basic model, $\mathcal{D}$, and a summary of a relational algebra for it, are briefly presented in Section 2. In Section 3 we formally define update operations, transactions, transaction logs, and state the Zero-loss Theorem. In Section 4 we



NAME	SALARY	DEPT
[8,52] John	[8,39] 15K [40,52] 20K	[8,39] Toys [40,52] Shoes
[48,→] Doug	[48,→] 20K	[48,→] Auto

The emp transaction history relation

NAME	TT	AUTHORIZER	USER	REASON
John	8	Don	Mark	New Employee
John	40	Don	Ryne	Reassignment
Doug	48	Joe	Rick	New Employee
John	53	Don	Damon	Fired

The shadow(emp) relation

QUERY	TT	USER
Q1: John's SALARY	42	Vance
Q1: John's SALARY	54	Andre
Q2: John's DEPT	55	Mitch
Q3: USER ID of Q1	56	Don

The Q-log relation

Figure 1.2. The model $\langle\langle\{emp\}, \{shadow(emp)\}\rangle, Q\text{-}log\rangle$

extend the basic algebra to query the whole zero information-loss model. We end this section with a list of assumptions made in this paper, a discussion of related papers, and some remarks about our notational conventions.

1.4. Assumptions. To simplify our presentation and avoid unnecessary distraction from the main theme of this paper, we make the following assumptions. These assumptions can be eliminated along the lines of [GB].

- The key of an object (tuple) in a transaction history relation $r \in \mathscr{D}$ does not change, and it is known correctly at all times. In the absence of this assumption, a new update operation, which allows keys of several objects to be changed simultaneously, becomes necessary [GB].
- At a given transaction time instant, there is only one transaction, i.e., the transactions are not batched together. This assumption is relaxed in [GB].
- No redundant updates are made, i.e., we do not change any attribute value of an object to itself. This assumption is also relaxed in [GB].

1.5. Related works. Algebras for temporal databases have appeared in [Gal,Ta,LJ]. [Sn] introduces a temporal model with two time dimensions and gives a QUEL like query language for it. [GB], frequently referenced here, relaxes the assumptions made in this paper; a very preliminary version of [GB] appeared in [GY2].

1.6. Notations. We write a singleton $\{x\}$ of any sort without braces. If R and S are schemes, RS denotes R∪S. If τ and τ' are tuples, then $\tau\circ\tau'$ denotes their concatenation; thus, if the schemes of τ and τ' are R and S, respectively, then the scheme of $\tau\circ\tau'$ is RS.

2. THE BASIC MODEL AND ITS ALGEBRA.

In this section we present a brief overview of the basic model $\mathscr{D}$. For a more detailed and generalized treatment of this basic model, the reader may refer to [GB].

2.1. The concept of time. We assume that we are given a universe of time instants $[0,NOW] = \{0, 1, 2, \cdots, NOW\}$, along with a linear order $\leq$ on it. NOW denotes the current

time according to the system clock. A subset I of [0,NOW] is an *interval* if $\forall t_1 t_2 t_3 (t_1 \in I \wedge t_2 \in I \wedge t_1 \leq t_3 \leq t_2 \Rightarrow t_3 \in I)$.

2.2. Temporal elements. A *temporal element* is a finite union of intervals. An interval is obviously a temporal element. An instant t may be identified with the temporal element [t,t]; thus it may be regarded as a temporal element. The set of all temporal elements is closed under + (union), * (intersection) and - (complementation), and thus forms a Boolean algebra. Temporal elements make a striking simplification in query languages for temporal databases [GY1,Ga2].

2.3. Temporal assignments. To capture the time-variant properties of objects, we introduce the notion of a temporal assignment. A *temporal assignment* ξ to an attribute A, with a temporal element μ as its *temporal domain*, is a function from μ such that for each instant $t \in \mu$, $\xi(t)$ is an element of dom(A). Figure 1.1 shows a temporal assignment to COLOR with [15,25] as its temporal domain.

If ξ is a temporal assignment, then $[\![\xi]\!]$ denotes its temporal domain and $|\xi|$ denotes its range $\{\xi(t) : t \in [\![\xi]\!]\}$. We may denote a temporal assignment ξ as $\langle \nu_1 \mapsto a_1, \cdots, \nu_m \mapsto a_m \rangle$, where $\nu_1 \cdots, \nu_m$ are temporal elements, $\xi(t) = a_i$ if $t \in \nu_i$, $1 \leq i \leq m$. If ξ is a temporal assignment, and μ a temporal element, then $\xi \restriction \mu$ denotes the restriction of ξ to the temporal domain $[\![\xi]\!] * \mu$.

2.4. θ-navigation. Suppose ξ_1 and ξ_2 are temporal assignments to θ-comparable attributes. We define $[\![\xi_1 \theta \xi_2]\!] = \{t \in [\![\xi_1]\!] * [\![\xi_2]\!] : \xi_1(t) \theta \xi_2(t)\}$. The construct $[\![\xi_1 \theta \xi_2]\!]$ is of fundamental importance in temporal databases. It evaluates to a time domain between $\emptyset$ and $[\![\xi_1]\!] * [\![\xi_2]\!]$. If the value is $\emptyset$, it implies that the two temporal assignments were never related. We define the Boolean $\xi_1 \theta \xi_2$ to be an abbreviation for $([\![\xi_1 \ \theta \ \xi_2]\!] = [\![\xi_1]\!] * [\![\xi_2]\!] \wedge [\![\xi_1]\!] * [\![\xi_2]\!] \neq \emptyset)$.

2.4.1. Example. If ξ_1 denotes the assignment to COLOR, as in Figure 1.1, and ξ_2 denotes $\langle [0,NOW] \mapsto blue \rangle$, then $[\![\xi_1 = \xi_2]\!] = [21,25]$.

2.5. Transaction history tuples. A *transaction history tuple* τ, over a scheme R, is a function from R such that for each attribute A in R, $\tau(A)$ is a temporal assignment to A. Suppose τ is a transaction history tuple over R. If μ is a temporal element, then $\tau \restriction \mu$ is the function from R such that $(\tau \restriction \mu)(A) = \tau(A) \restriction \mu$, for every $A \in R$. We say that τ is *homogeneous* if $[\![\tau(A)]\!]$ is the same for all attributes $A \in R$. In this paper we will only be interested in homogeneous tuples. We say that τ is *null* if $[\![\tau(A)]\!] = \emptyset$ for some $A \in R$.

2.6. Transaction history relations. A *transaction history relation* r over R, with $K \subseteq R$ as its *key*, is a finite set of non-null homogeneous transaction history tuples over R such that (i) $|\tau(A)|$ is a singleton for every $A \in K$, and (ii) if τ and τ' are tuples in r such that $\forall A \in K(|\tau(A)| = |\tau'(A)|)$, then $\tau = \tau'$. Figure 1.2 shows the emp transaction history relation over NAME SALARY DEPT with NAME as its key.

Note that our assumption that the value of a key attribute does not change with time has been used in the definition of a key of a transaction history relation. The behavior of a key in temporal databases is more complex than its classical counterpart; its detailed account would force the surfacing of several subtleties and auxiliary definitions [GY2,GB], and cause a substantial distraction from the main theme of the paper.

The algebra presented in this Section is limited in the following two respects: (i) We do not introduce certain restructuring operators which are necessary to change the key attributes of a transaction history relations. (ii) We give a limited definition of the projection and natural join operators. Both of these limitations are overcome in the detailed presentation provided in [GB]. An informal discussion of the problems involved is given in [GY2].

If r is a transaction history relation over R, then $[\![r]\!] = \bigcup_{\tau \in r} [\![\tau]\!]$. For example, for the state of the emp relation shown in Figure 1.2, $[\![emp]\!] = [8,NOW]$. If μ is a temporal element then $r \restriction \mu = \{\tau \restriction \mu : \tau \in r\}$. If t is an instant, then $r \restriction t$ is called a *temporal snapshot* of r at t.

2.7. An algebra for transaction history relations. We assume that a database $\mathcal{D}$, consisting of finitely many transaction history relations is given. The set of all algebraic expressions can be divided into three mutually exclusive groups: temporal expressions, Boolean expressions, and relational expressions. In the following definitions, a constant temporal assignment $\langle [0,NOW] \mapsto a \rangle$ is abbreviated as a.

2.7.1. Temporal expressions. Temporal expressions, the syntactic counterpart of temporal elements are formed from $[\![r]\!]$, $[\![\xi_1 \theta \xi_2]\!]$, +, *, and -. If μ is a temporal expression and τ is a tuple, then $\mu(\tau)$, the result of substituting τ in μ, is defined in a natural manner.

2.7.1.1. Example. Let us consider the emp relation of Figure 1.2. Suppose τ denotes John's tuple. Then [[SALARY ≠ 20K]] is a temporal expression, and for τ its value is [8,39]. Similarly, [[DEPT = Shoes]](τ) = [40,52]. The result of substituting τ in the temporal expression [[SALARY ≠ 20K]] + [[DEPT = Shoes]] is [8,52].

2.7.2. Boolean expressions. Atomic Boolean expressions are of the form, TRUE, FALSE, $\xi_1 \theta \xi_2$, and $\mu \subseteq \nu$. More complex expressions are formed by using the Boolean operators ∧, ∨, and ¬.

2.7.3. Relational expressions. Relational expressions are syntactic counterparts of transaction history relations. Every relation r ∈ $\mathcal{D}$ is a relational expression. In the following, we suppose r is a transaction history relation over R, with K⊆R as its key. The other relational expressions are defined as follows.

- If s is a relational expression over R, with K⊆R as its key, then r−s, r∪s, r∩s are transaction history relations over R, with K as their key. In giving the semantics for r∪s it is assumed that for the same object, the versions of their transaction histories in r and s do not differ at any instant; otherwise, r∪s is undefined. This is analogous to the classical case where, if r and s are snapshot relations with K as their key, then K may not be a key of r∪s.
- If K ⊆ X ⊆ R then $\Pi_X(r)$ is a transaction history relation with K as its key. We recall that the condition K⊆X has been imposed in this paper to ensure that the key of $\Pi_X(r)$ does not have to be changed.
- If f is a Boolean expression, and μ a temporal expression, then $\sigma(r;f;\mu)$ is a transaction history relation over R, with K as its key. σ is an important operator in temporal databases. It allows a tuple τ in r to be examined with respect to a Boolean criterion f, and then restricted to $\mu(\tau)$. It stands for the transaction history relation $\{\tau \upharpoonright \mu(\tau): \tau \in r \wedge f(\tau)\}$.
- If s is a relational expression over S, with K' as its key, then r ⋈ s is a transaction history relation over RS with K∪K' as its key. The expression r ⋈ s evaluates to $\{(\tau \circ \tau') \upharpoonright ([\![\tau]\!] * [\![\tau']\!]): \tau \in r \wedge \tau' \in s\}$. The restriction of $\tau \circ \tau'$ to [[τ]]*[[τ']] is necessary to ensure homogeneity. In general, the key may be a proper subset of K∪K'.
- If A ∈ R and B ∉ R, then the transaction history relation $\rho_{A \leftarrow B}(r)$ is obtained from r by renaming the attribute A to B. If A ∈ K, the new key is (K−A)∪B.

2.8. The query-users. As $\mathcal{D}$ encapsulates the entire database history of objects, it is very rich in content. Our model supports two kinds of query-users: (i) The *system-user*, who has access to the entire database $\mathcal{D}$ and can use the powerful operators to be introduced in Section 4. (ii) The *classical-user*, who can only access $\mathcal{D}$ ↾NOW = {r ↾NOW: r∈$\mathcal{D}$}, the current state of the database. In $\mathcal{D}$ ↾NOW, the only timestamps are NOW. Such timestamps are clearly redundant, and are hidden from the classical-user. This user has no use of temporal expressions; his/her needs are adequately covered by the Boolean expressions. Thus, the third argument of σ is not made available to him/her. In this way, the query interface for the classical-user of our model is similar to that for a user in a classical snapshot database.

2.9. Examples. We look at some queries that can be asked of transaction history relations by a system-user. All these queries refer to the emp relation of Figure 1.2.

2.9.1. Example. According to the database, at what transaction time was John's salary shown as 15K while he was working in Toys? This query is expressed as:

[[σ(emp; NAME = John; [[SALARY = 15K]] * [[DEPT = Toys]])]]

2.9.2. Example. Give details about the employees for whom the database showed a salary greater than 24K while they worked in the Clothing or the Shoes department.

σ(emp; TRUE; [[SALARY > 24K]] * ([[DEPT = Shoes]] + [[DEPT = Clothing]]))

3. TRANSACTIONS AND ZERO INFORMATION-LOSS.

Informally, a transaction is either a single update or a single query. We impose a logical structure on the log of all transactions to obtain the model ⟨⟨$\mathcal{D}$, Shadow⟩, Q-log⟩ from which we can restore the log at any time. Thus, there is no loss of information in this model, and it is called a zero information-loss model. The advantage of the logical structure is that it lends itself to a powerful query language. The query language is the subject of study in the next section.

3.1. Shadow relations. Just as a transaction history relation records the update made in the value of an attribute, the shadow relation records the environment of the update. Consequently, every transaction history relation has a shadow relation associated with it, and for every update represented in the transaction history relation, there is a corresponding tuple in the shadow relation.

Suppose r is transaction history relation over R in $\mathscr{D}$, with K$\subseteq$R as its key. Then the scheme of shadow(r) must contain {USER, TT} U K. A USER is one who makes the update, and it can be a person, an algorithm, a machine, or any other decision making body. The transaction time TT is supplied by the system clock when an update becomes effective. The purpose of K is to establish a connection between an object in a transaction history relation, and the corresponding tuple in its shadow. Other attributes in shadow(R) depend upon the nature of r. For example, in an employee shadow relation it might be relevant to record who authorized the salary update, whereas in a weather record shadow relation it may make sense to record the space coordinates of the satellite reporting the information. Figure 1.2 shows the shadow(emp) relation, over NAME TT AUTHORIZER USER REASON.

3.2. Updates. Throughout this section we will assume that we are given a fixed but arbitrary transaction history relation r $\in \mathscr{D}$ over R, with K$\subseteq$R as its key. If $\tau \in$ r, then key(τ) denotes the snapshot tuple τ_1 over K, such that for every A $\in$ K, $\tau_1(A) = |\tau(A)|$.

In the zero information-loss model, an *update operation* transforms a transaction history relation r, and its corresponding shadow relation, from one state to another. The update operations on $\langle \mathscr{D}$, Shadow$\rangle$, are as follows:

- *insert* τ *in* r *with* τ';
- *modify* key(τ) *to* new_τ *in* r *with* τ';
- *delete* key(τ) *with* τ';

where τ' is the part of the update operation that has the circumstantial or shadow information about the update. Several examples of these operations have already been given in Section 1. As mentioned in the introduction, an update operation u on $\langle \mathscr{D}$, Shadow$\rangle$ can be partitioned into two parts: u^1 and u^2, which modify $\mathscr{D}$ and Shadow, respectively. To make this partitioning more visible, we first make syntactic modifications in our update operations so that an update operation u can be expressed as $\langle u^1, u^2 \rangle$:

- $\langle$*insert* τ *in* r, *append* τ'∘key(τ) *to* shadow(r)$\rangle$
- $\langle$*modify* key(τ) *in* r *to* new_τ, *append* τ'∘key(τ) *to* shadow(r)$\rangle$
- $\langle$*delete* key(τ) *in* r, *append* τ'∘key(τ) *to* shadow(r)$\rangle$

If u is an update operator on $\langle$r,shadow(r)$\rangle$, and u^1 and u^2 are such that u = $\langle u^1, u^2 \rangle$, then u^1 and u^2 are called *update operations* on r and shadow(r), respectively. Note that "*append* τ'∘key(τ) *to* shadow(r)" is the only update operation for the shadow relation shadow(r). The semantics of an update operation on r can be defined and the following lemma can be proved easily.

LEMMA 1. If $\langle$r',shadow(r')$\rangle$ are obtained after update u to $\langle$r,shadow(r)$\rangle$ by the user x at transaction time t, then $\langle$r,shadow(r)$\rangle$, u, x, and t can be determined from $\langle$r',shadow(r')$\rangle$ alone.

3.3. Q-log. Q-log is essentially a log of all queries. A point of interest is that the same query may be made by different users at different times. For example, a compiled query program may be run at various times. The decision about how a query is to be stored - as a string, or as a parse tree, or as some other semantic entity - is left to the choice of the implementer. We only require that there be a decision procedure for deciding whether or not two queries are the same. This algorithm can be smart enough to try and deduce the logical equivalence of queries, or so simple as to consider textually different queries to be different.

We introduce a special attribute QUERY, and assume that dom(QUERY) consists of all queries. We define *Q-scheme* to be QUERY USER TT, and Q-log to be a relation over Q-scheme. Thus, a tuple τ in Q-log is of the form $\langle$Q, x, t$\rangle$, where Q is a query, x a user, and t a transaction time instant. The meaning of τ is that Q was executed by user x at transaction time t. Figure 1.2 shows an example of a Q-log relation.

Suppose s represents the current state of Q-log, the query q is asked by the user x, and the system answers it for the state of the zero information-loss model at transaction time t. Then the new state of Q-log becomes s U {q ∘ x ∘ t}.

LEMMA 2. Suppose the query q, by user x, at transaction time t, transforms the state s of a Q-log to the state s'. Then s, q, x, and t can be determined from s' alone.

3.4. Information about a transaction. The *information* associated with a transaction is the data which is deemed relevant for the application on hand. This definition places the burden of deciding relevance on the system designer. Thus, the system designer has to make a priori decisions about what kind of circumstantial information about transactions is of value and needs to be stored.

3.5. Transactions and transaction log. A *transaction* T in the zero information-loss model is of the form $\langle u/q,x,t\rangle$, where u/q is either an update or a query, x is a user, and t is a transaction time instant. If the transaction T, on the state Z of the zero information-loss model, results in the state Z', we write $Z' = T(Z)$. The following follows from Lemmas 1 and 2.

LEMMA 3. If T is a transaction on Z, then from T(Z) alone, we can restore T and Z.

A *transaction log* is a sequence $T = \langle T_1, T_2, \cdots, T_n\rangle$ of transactions. Suppose $\emptyset$ denotes the initial empty state of the zero information-loss model. Then the *outcome* of the transaction log T is the state $T_n(T_{n-1}(\cdots(T_1(\emptyset))\cdots)$.

THEOREM 1 (The Zero-loss Theorem). A transaction log T can be restored from its outcome.

4. ALGEBRA FOR THE ZERO INFO.-LOSS MODEL.

In this Section we extend the algebra of Section 2 to query the complete zero information-loss model.

4.1. Classical operators for shadow relations. The operators for Shadow are as follows:

- Shadow relations are the classical 1nf relations, and we allow the classical relational operators on them.

4.1.1. Examples. Suppose emp(NAME SALARY DEPT) is a transaction history relation with NAME as its key, and management(DEPT MANAGER) is a transaction history relation with DEPT as its key. Suppose the scheme of shadow(emp) is as before, and the scheme of shadow(management) is TT AUTHORIZER DEPT USER REASON. The following queries are based on the shadow(emp) and shadow(management) relations.

4.1.1.1. Example. Give reasons for all the updates in emp.

$$\Pi_{\text{REASONS}}(\text{shadow(emp)})$$

4.1.1.2. Example. Give reasons for all the updates in emp that were authorized by Harry.

$$\Pi_{\text{REASONS}}(\sigma(\text{shadow(emp)};\ \text{AUTHORIZER} = \text{Harry}))$$

4.1.1.3. Example. Who made changes in both emp and management?

$$\Pi_{\text{shadow(emp).USER}}(\text{shadow(emp)} \bowtie_{\text{USER}} \text{shadow(management)})$$

4.1.1.4. Example. Who made the updates in emp between transaction time(TT) 10 and 20?

$$\Pi_{\text{USER}}(\sigma(\text{shadow(emp)};\ \text{TT} \geq 10 \wedge \text{TT} \leq 20))$$

The queries in the algebra for shadow relations can use temporal expressions involving relations in $\mathcal{D}$. This is illustrated by the following example.

4.1.1.5. Example. It is easily seen that the temporal expression $[\![\sigma(\text{management};\ \text{MANAGER} = \text{Tom};\ [0,\text{NOW}])]\!]$ retrieves the time when Tom is shown to be a manager in some department. Using this, the query "what were the reasons for updates in emp during the time Tom was a manager in some department?" is expressed as follows.

$$\Pi_{\text{REASONS}}(\sigma(\text{shadow(emp)};\ \text{TT} \subseteq [\![\sigma(\text{management};\ \text{MANAGER} = \text{Tom};\ [0,\text{NOW}])]\!]))$$

4.2. Navigation between $\mathcal{D}$ and Shadow. Suppose τ is a transaction history tuple over R. If $X \subseteq R$, we define the temporal expression $\delta(\tau(X)) = \{t:$ for some $A \in X$, $\tau(A)\{t \neq \tau(A)\{(t-1)\}$. Clearly, $\delta(\tau(X))$ is the set of all transaction time instants when there is a change in $\tau(X)$.

4.2.1. Semijoins of relations in $\mathcal{D}$ and Shadow. Because the structures of a transaction history relation and a snapshot relation are quite different, we do not want to define their join. Instead we define semijoin operations, whose navigational nature is like a join, but instead of retrieving the concatenation of the tuples of the two operand relations, we only retrieve parts of tuples of one of the operand relations.

Suppose $r(R) \in \mathcal{D}$, e_1 and e_2 are expressions which evaluate to subsets of r and shadow(r), respectively, and f is a Boolean condition involving attributes of schemes of r and shadow(r). Then we introduce F_1 and F_2, called the *filter operations*, as follows.

$F_1(e_1,e_2,f) = \{\tau \in e_1 : \exists \tau' \in e_2(\tau'(TT) \subseteq \delta(\tau(R)) \wedge f(\tau \circ \tau'))\}$,

$F_2(e_1,e_2,f) = \{\tau' \in e_2 : \exists \tau \in e_1(\tau'(TT) \subseteq \delta(\tau(R)) \wedge f(\tau \circ \tau'))\}$,

where $\tau'(TT) \subseteq \delta(\tau(R))$ expresses "τ' is a shadow tuple of τ corresponding to an update operation which modified at

least one attribute in τ."

4.2.1.1. Example. Which tuples in emp were updated by Harry?

F_2(emp, shadow(emp), USER = Harry)

4.2.1.2. Example. Name the people who made changes in Tom's tuple during [10,20].

$\Pi_{USER}(F_1(\sigma(emp; NAME = Tom; [0,NOW]),$
shadow(emp)), $10 \leq TT \wedge TT \leq 20))$

4.3. Operators for Q-log. Recall, a tuple in Q-log is of the form $\langle Q, x, t\rangle$, with the meaning that the query Q was executed by user x at transaction time t. If τ is the tuple $\langle Q, x, t\rangle$, then it is natural to identify it with the relation retrieved by Q when it was executed at transaction time t. (Note that from x, we can decode whether this user is a system-user or a classical-user.) We denote this relation as $[\tau]$, and define [Q-log] = $\{[\tau]: \tau \in$ Q-log$\}$. Thus, we arrive at the database $Z = \mathscr{D} \cup$ Shadow $\cup$ {Q-log} $\cup$ [Q-log]. Z can be partitioned into two parts: Z^1 consisting of the classical 1-nf relations, and Z^2 consisting of transaction history relations. On Z^1 we allow the classical relational operators, and on Z^2 we allow the algebra described in Section 2. Additionally, in this section we have introduced operators on $\mathscr{D} \cup$ Shadow. Clearly [Q-log] allows a query to be treated as a relation, which can be queried. Thus, we can query queries, query queries on queries, etc.

4.3.1. Example. Name the people who executed Q1.

$\Pi_{USER}(\sigma(Q\text{-log}, QUERY = Q1))$;

4.3.2. Example. What difference was observed in the execution of Q1 at TT = 3 and 10 by Harry?

[Q1,Harry,10] - [Q1,Harry,3].

4.3.3. Example. If Q is the query "what are all the naval bases," executed by Vance at transaction time t, then σ([Q,Vance,t]; LOCATION = Europe) can be used as an answer to the question "What naval base locations in Europe were revealed to Vance?" and the query $\Pi_{COMMANDER}$([Q,Vance,t]) can be used to answer the question "What COMMANDER information was released to Vance?"

The above example demonstrates the utility of querying the Q-log. This has great application in areas where it is necessary to monitor the accesses being made to the database. Such query capability allows the database administrator to analyze the information-flow patterns.

5. CONCLUSIONS.

We have presented a relational model which incorporates the information-rich transaction log. This is done in a manner which allows us to query the transactions and their environment. As a result our model is completely self-contained and has zero information-loss. We believe that it fulfills an important need for being able to examine the nature of transactions in a database system. It also applies to a knowledge based system where it is important to retain, both, a complete historical knowledge and the evolution process of such knowledge over time. We believe that this model can be used as a workbench for building very secure systems.

ACKNOWLEDGEMENT. We thank the anonymous referees for their helpful comments.

REFERENCES

[Ga1] Gadia, Shashi K. *A Homogeneous Relational Model and Query Languages for Temporal Databases.* ACM-Transactions on Database Systems, pp 418-448, vol 13, 1988.

[Ga2] Gadia, Shashi K. *The role of temporal elements in temporal databases.* Quarterly Bull. of the Computer Society of IEEE Technical Committee on Data Engineering, December, 1988.

[GB] Gadia, Shashi K. and Bhargava, Gautam. *A Formal Treatment of Updates and Errors in a Relational Database.* Submitted for publication.

[GY1] Gadia, Shashi K. and Yeung, Chuen-Sing. *Inadequacy of Interval Timestamps in Temporal Databases.* To appear in Information Sciences.

[GY2] Gadia, Shashi K. and Yeung, Chuen-Sing. *A Generalized Model for a Relational Temporal Database.* Proc ACM-SIGMOD Int. Conf. on Management of Data, June 1988, pp. 251-257.

[LJ] Lorentzos, Nikos A. and Johnson, Roger G. *Extending relational algebra to manipulate temporal data.* Information Systems, Vol 13, 1988, pp. 289-296.

[Sn] Snodgrass, Richard. *The Temporal Query Language TQuel.* ACM Transactions on Database Systems, Vol 12, 1987, pp 247-298.

[Ta] Tansel, A.U. *Adding Time Dimension to Relational Model Model and Extending Relational Algebra.* Information Systems, 1986.