29

Extending Existing Dependency Theory to Temporal Databases

Christian S. Jensen, Richard T. Snodgrass, and Michael D. Soo

Normal forms play a central role in the design of relational databases. Several normal forms for temporal relational databases have been proposed. These definitions are particular to specific temporal data models, which are numerous and incompatible.

This paper attempts to rectify this situation. We define a consistent framework of temporal equivalents of the important conventional database design concepts: functional dependencies, primary keys, and third and Boyce-Codd normal forms. This framework is enabled by making a clear distinction between the logical concept of a temporal relation and its physical representation. As a result, the role played by temporal normal forms during temporal database design closely parallels that of normal forms during conventional database design. These new normal forms apply equally well to all temporal data models that have timeslice operators, including those employing tuple timestamping, backlogs, and attribute value timestamping.

As a basis for our research, we conduct a thorough examination of existing proposals for temporal dependencies, keys, and normal forms. To demonstrate the generality of our approach, we outline how normal forms and dependency theory can also be applied to spatial and spatiotemporal databases.

Keywords: temporal relation, valid time, transaction time, functional dependency, data semantics, normal form, database design

1 Introduction

A central goal of relational database design is to produce a *database schema*, consisting of a set of *relation schemas*. Each relation schema is a collection of attribute names and their associated domains.

Normal forms are an attempt to characterize "good" relation schemes. A wide variety of normal forms has been proposed, the most prominent being third normal form and Boyce-Codd normal form. An extensive theory has been developed to provide a solid formal footing.

There is also a need for temporal normal forms and underlying concepts that may serve as important guidelines during temporal database design. In response to this need, an array of temporal normalization concepts have been previously proposed, including *first temporal normal form* [42], two variants of *time normal form* [4, 32], and *P* and *Q normal forms* [26].

The proposals are significant since each, in the context of a particular temporal data model, can be used to design temporal database schemas. However, the specificity of the proposals is a weakness since a given normal form inherits the inherent peculiarities of its data model, and, having chosen a particular temporal normal form, it is unsatisfactory to be required to define all of the normal forms anew for each of the two dozen existing temporal data models [45], should another model be better suited for representing the semantics of the application. Furthermore, the existing normal forms often deviate substantially in nature from conventional normal forms and are in some sense not "true" extensions of these, for a variety of reasons that we detail later in this paper.

In this paper, we show how temporal normal forms, including the related concepts of temporal dependencies and temporal keys, may be defined so that they apply to all temporal data models, and so that temporal database design concepts closely parallel their conventional counterparts. We do not simply focus on a single temporal data model. Instead, we utilize a new data model, termed the *bitemporal conceptual data model* (BCDM), that is, in some sense, the "largest common denominator" of existing temporal models [21]. Specifically, we have shown how to map relations and operations in several quite different temporal relational data models into relations and operations in this data model. This is an important property, as it ensures that the normal forms expressed in this model are applicable also to other models. We define the temporal normal forms in the context of this model. Our proposal accommodates *valid-time*, *transaction-time* and *bitemporal* relations [43, 20]. We also note that the BCDM has been adopted as the underlying data model of the consensus temporal query language TSQL2 [47]. Design of TSQL2 schemas thus directly benefits from the definitions of temporal dependencies and normal forms introduced here.

Our focus is on the design of temporal database schemas. A substantial body of work exists on the specification and efficient checking of more general temporal, or dynamic, integrity constraints. The dependencies, keys and normal forms of this paper can be seen as constraints on database instances, but are different in two respects. Unlike general constraints, they impact the design of temporal databases. And since the focus is on database design, on-line checking of the constraints is not of relevance here.

We also believe that additional normalization concepts are needed that take the different temporal characteristics of data into consideration, but the development of such concepts is beyond the scope of this paper. Instead, this paper is restricted to providing data-model independent mappings of the existing conventional normalization concepts to temporal databases.

We also limit the scope of the paper to so-called *intra-state dependencies* [5]. Intra-state dependencies are defined in terms of individual snapshots of a temporal database. For example, the conventional notions of functional dependency and multivalued dependency are, by definition, intra-state dependencies. *Inter-state dependencies*, on the other hand, express constraints between attribute values in different snapshots.

The paper is organized as follows. In Sections 2, 3, and 4, we examine all existing definitions, to our knowledge, of temporal dependencies, keys, and normal forms, respectively. This is the first thorough survey of work in these areas. Each section first briefly describes the relevant conventional normalization concepts, and lists a number of important properties that should carry over to their temporal counterparts. On this basis, the temporal database design proposals known to us are introduced and evaluated. The existing definitions satisfy many, but not all, of the properties required of entirely natural extensions of conventional normal forms. The existing temporal design concepts provide a valuable foundation upon which we subsequently build.

The topic of Section 5 is the bitemporal conceptual data model. We describe the type of relation supported by the BCDM, and briefly describe a few algebraic operators needed to support the new temporal normal forms defined in the next section, where we then develop temporal counterparts of the conventional dependencies, keys, and normal forms, again, limiting ourselves in this paper to intra-state variants. This is done is such a way that virtually all of the conventional normalization theory carries over to the temporal context. The result is that the role played by temporal normal forms during temporal database design closely parallels that of normal forms during conventional database design. This is possible, in part, because of a careful choice of temporal data model. Section 7 explores the properties of the temporal framework.

To demonstrate the generality of our approach, we outline, in Section 8, how normal forms and dependency theory can also be applied to spatial and spatiotemporal databases [2]. Recent work, conclusions, and future research are the subjects of Sections 9 and 10.

2 Previous Proposals for Temporal Dependencies

In this section, we consider previous proposals of temporal dependencies (in chronological order, of course!). To provide a basis for this, we first review the definition of the conventional functional dependency, and then we highlight those properties that we feel should also be satisfied by corresponding temporal database dependencies.

2.1 Conventional Functional Dependency

Throughout the paper, we generally use R to denote an arbitrary relation schema, and $r(R)$ to denote that r is an instance of R. Explicit (non-temporal) attributes of a relation schema are generally denoted $A_1, \ldots A_n$, and X and Y are used to denote sets of attributes. For tuples, the symbol s is used (possibly indexed), and $s[X]$ denotes the projection of tuple s onto the attributes X.

For the purpose of database design, a functional dependency [11] is an *intensional* property of a database schema. We associate with each schema a set of all instances that are possible in the modeled reality, termed the *meaningful instances*.

Definition 1 Let a relation schema R be defined as $R = (A_1, A_2, \ldots, A_n)$, and let X and Y be sets of attributes of R. The set Y is *functionally dependent* on the set X, denoted $X \rightarrow Y$, if for all meaningful instances r of R

$$\forall s_1, s_2 \in r \ (s_1[X] = s_2[X] \Rightarrow s_1[Y] = s_2[Y]).$$

If $X \rightarrow Y$, we say that X *determines* Y. A functional dependency $X \rightarrow Y$ is *trivial* if $Y \subseteq X$. □

A functional dependency describes (and constrains) the set of possible extensions of a relation. Which functional dependencies are applicable to a schema reflects the reality being modeled and the intended use of the database. Determining the relevant functional dependencies is a primary task of the database designer.

The two most important normal forms, third normal form [10] and Boyce-Codd normal form [12], as well as the concept of key, all rely on the concept of functional dependency.

Example 1 To illustrate, consider a database recording the phone numbers, departments, and employees in a company. This can be modeled with the schema Emp = (Name, Dept, PhNo). In this company, an employee can belong to only one department, meaning that Name → Dept. An employee may have several phone numbers, so Name does not determine PhNo. □

Definition 2 The *closure* of a set of functional dependencies, F, is the set of dependencies, denoted F^+, that are logicall implied by F. □

Rather than applying the definition of functional dependency directly, it is customary to apply a set of inference rules to derive new, implied dependencies.

Armstrong's axioms, a set of three inference rules, are among the most popular of such rules. This set has been proven to be sound and complete, meaning that precisely those dependencies that can be derived using the definition of functional dependency can also be derived using the rules [3].

Example 2 In the example database, the closure of the given functional dependency contains the following additional non-trivial dependencies.

$$\{\text{Name, PhNo}\} \rightarrow \{\text{Dept}\}$$
$$\{\text{Name, PhNo}\} \rightarrow \{\text{Name, Dept}\}$$
$$\{\text{Name, PhNo}\} \rightarrow \{\text{PhNo, Dept}\}$$
$$\{\text{Name, PhNo}\} \rightarrow \{\text{Name, PhNo, Dept}\} \qquad \square$$

As a basis for the subsequent discussion, we summarize here two fundamental qualities of dependencies (e,g., functional).

D1. Dependencies are intensional, not extensional, properties.
While it may require only few textual modifications to change an extensional definition into an obvious intensional counterpart, the conceptual difference between intensional and extensional concepts is significant. Dependencies and normal forms are applied to relation schemas during database design where no instances are present yet. Thus, extensional definitions make little sense conceptually.

D2. Dependencies are defined independently of the representation of a relation. These concepts are based on semantics, not on an arbitrary representation.
The meaning of this desideratum will become clearer once the necessary concepts have been introduced; see Section 5.3. Briefly, in some data models it is possible to have different relation instances that nevertheless contain the same temporal information (i.e., are snapshot equivalent [21]). Such instances should satisfy the exact same dependencies. Further, instances in different temporal data models with the same information content should satisfy the same dependencies.

While conventional dependencies may share additional qualities, these two qualities are fundamental and are satisfied by all conventional dependencies, including multivalued and join dependencies.

We add a third desideratum that is also related to the discussions on temporal keys in Section 3. However, it is more appropriately applied to functional dependencies.

D3. Functional dependencies are used to define keys.

We now characterize each of the previously proposed temporal dependencies based on these desiderata.

2.2 Extensional versus Intensional Database Constraints

In one of the first papers in temporal databases, Clifford and Warren make the distinction between *extensional database constraints,* which "can be said to hold (or not hold) simply on the basis of the extension of the database with respect to a single state" [[9], p. 246] (where "single state" is at a particular point in time), and *intensional database constraints,* which "can be said to hold (or not hold) only by examining at least two states of the" valid-time database (this terminology is somewhat inconsistent with the normal definition of *intensional* as applying to all possible states, or *extensions*). They classify conventional functional dependencies as extensional constraints. They then show that their intensional logic (IL_s) allows one to specify explicitly that a functional dependency must hold over all states of the database. Their logic is also able to specify other kinds of intensional constraints, such as "No employee can later return to the same department."

Concerning the subject of this paper, this early work is preliminary, in that a functional dependency over time was never defined. Subsequent efforts, to be discussed next, to define temporal variants of functional dependencies have taken different tacks. We feel, however, that the general approach introduced in Clifford's paper is the appropriate one. We will define in Section 6 a *temporal functional dependency* and associated normal forms by formalizing the notion that a functional dependency should hold over all time.

2.3 Dynamic Functional Dependencies

In the context of a formal model for the evolution of databases in time, Vianu extended the notion of functional dependencies (FDs) to hold over consecutive states of a database. In this definition, U is a set of attributes in the relational schema, $\check{U}$ represents the values of these attributes in a state of the database, and $\hat{U}$ represents the values of these attributes in the next state of the database (Vianu provides formal definitions for these sets). The basic idea is to have attribute values in consecutive states determine values of other attributes in these states.

Definition 3 "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 $XA \cap \hat{U} \neq \emptyset$. ... 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 $\hat{U}$. (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." [[52], p. 36] $\square$

In an example provided by Vianu of an "equal opportunity" policy, a new salary is determined solely by merit in conjunction with the old salary. This policy may be encoded in the DFD $M\check{E}RIT\ S\check{A}L \to S\hat{A}L$.

This definition satisfies Desiderata **D1** and **D2** listed in Section 2.1. No keys are defined in terms of DFDs.

In the present paper, we will be concerned only with dependencies on temporal databases that can be expressed on individual snapshots. Dynamic functional dependencies address a different problem. Also, we will accommodate valid-time, transaction-time, and bitemporal relations; dynamic dependencies are defined solely over transaction-time relations.

2.4 Temporal Dependency in the Temporal Relational Model

The Time Relational Model (TRM) [32] is a valid-time data model. Valid-time is supported by appending to each tuple two time attributes (T_s, T_e) denoting that the tuple was valid during the closed interval $[T_s, T_e]$.

As the basis for the definition of a temporal normal form (to be introduced later), Navathe and Ahmed defined the notion of temporal dependency as follows.

Definition 4 There exists a *temporal dependency* between two time-varying attributes, A_i and A_j, in a relation schema $R = (A_1, A_2, \ldots, A_n, T_s, T_e)$ if there exists an extension $r(R)$ containing two distinct tuples, t and t', that satisfy each of the following three properties.

1. $t[K] = t'[K]$ where K is the time invariant key.
2. $t[T_e] = t'[T_s] - 1 \vee t'[T_e] = t[T_s] - 1$.
3. $t[A_i] = t'[A_i]$ XOR $t[A_j] = t'[A_j]$.

[[32], p. 156 and [1]] □

Thus, two attributes are mutually dependent if we are able to find, in some extension, two tuples that represent the same object of the modeled reality, have consecutive valid-time intervals, and agree on attribute A_i and disagree on attribute A_j or disagree on A_i and agree on A_j. If two attributes are *not* mutually dependent, then they are termed *synchronous*, as they change simultaneously.

The desire is to include only synchronous attributes in a relation. Otherwise, when the value of an attribute changes, the other attributes retain their previous value; these values must be replicated in the new tuple, creating redundant information.

Example 3 Consider the following relation instance with time-invariant key Emp.

Emp	Dept	Mgr	T_s	T_e
Bill	Shipping	Zoe	1	4
Bill	Loading	Zoe	5	10

In this relation instance a temporal dependency exists between the Dept and Mgr attributes. Because of this temporal dependency, the Emp and Mgr attribute values had to be copied to a new tuple when Bill changed departments. □

This dependency satisfies only Desideratum **D1**. It is dependent on the representation of a temporal relation and is not used for defining keys.

2.5 Nested Relations with Valid-Time

Tansel and Garnett showed how nested relations can be augmented with valid-time to support complex objects [50]. This data model uses attribute value timestamping. Attribute values have the form $< t, v >$ where t is a valid-time element (a set of maximal valid-time intervals) and v is a conventional attribute value. Informally, t indicates the time intervals when the attribute had the value v. Attribute values may be either atomic attribute values, as just described, or they may be nested relations themselves.

This model does not define temporal dependencies. Instead, it defines snapshot multivalued dependencies between atomic attribute values, where the valid-time associated with the atomic attribute value is treated as an explicit part of the attribute.

Example 4 Consider the following relation instance with key Emp. For simplicity, we only show a single valid-time interval associated with each attribute.

<u>Emp</u>	Dept	Mgr
< [1, 10], Bill >	< [1, 4], Shipping > < [5, 10], Loading >	< [1, 6], Zoe > < [7, 10], Janet >

The relation contains a single tuple showing Bill's employment history for the interval from time 1 until time 10. Temporal intersection of the attribute timestamp is used to interpret the relation. From time 1 until time 4, Bill worked for the shipping department and was managed by Zoe. From time 5 to time 6, Bill worked for the loading department. During this time his manager remained Zoe. At time 7 and continuing until time 10, Bill remained at the loading department but had Janet as his manager.

We can unnest the nested relation to explicitly show the multivalued dependencies. Some of the tuples produced in the unnesting may not be meaningful since their attribute timestamps have an empty intersection.

<u>Emp</u>	Dept	Mgr
< [1, 10], Bill >	< [1, 4], Shipping >	< [1, 6], Zoe >
< [1, 10], Bill >	< [1, 4], Shipping >	< [7, 10], Janet >
< [1, 10], Bill >	< [5, 10], Loading >	< [1, 6], Zoe >
< [1, 10], Bill >	< [5, 10], Loading >	< [7, 10], Janet >

The first, third, and fourth tuples in the unnested relation encode the same facts over the intervals [1,4], [5,6], and [7,10], respectively, as does the single tuple in the nested relation. The second tuple does not record a meaningful fact since the intersection of its attribute timestamps is empty.

If we regard the attributes as atomic, i.e., the timestamps contained in the attributes are considered explicit values, it should be clear from the unnested relation that the multivalued dependencies Emp$\twoheadrightarrow$ Dept and Emp$\twoheadrightarrow$ Mgr hold.

Moreover, these dependencies are the traditional multivalued dependencies used in snapshot database design to obtain fourth normal form relation schemas. □

The central observation is that multivalued dependencies exist between the key attribute and set-valued attributes in nested relations. While also present in non-temporal nested relations [33], the same observation holds when valid-time is added to the model. Tansel and Garnett use these multivalued dependencies to guide the normalization of nested valid-time schemas.

This approach satisfies Desideratum **D1**, but not **D2** (although the distinction between semantics and representation perhaps is somewhat unclear). Desideratum **D3** is not applicable to multivalued dependencies.

2.6 The Interval Extended Relational Model

The interval extended relational model (IXRM) [27] integrates (n-dimensional) intervals into the snapshot relational model and meets in this way the needs of many application areas. The intervals, one per attribute, may be drawn from any data type, including time and space. Interval-valued attributes are accommodated, and new operators that manipulate relations with interval attributes are defined.

The IXRM is not a temporal data model. The timestamps in a tuple do not specify when that tuple, or even an attribute value in that tuple, was valid. Put differently, the query language does not interpret interval attributes as temporal attributes. The IXRM was designed to be used in applications (e.g., soil management, e.g., [26]) where according interval attributes temporal semantics would render such attributes of little use. Rather, such timestamps are more properly thought of as *user-defined time* [43]. The IXRM is mentioned here because, while it is not a valid-time model, database users may think of relations in this model as *representing* valid-time relations. Indeed, some of the operators of the IXRM query language may conveniently be used for valid-time queries.

Lorentzos extended the notion of functional dependency in two ways. In the following definitions, I(D) denotes the domain of intervals delimited by points in domain D; and X(D) denotes an arbitrary domain of either points or intervals.

Definition 5 "If R(A=I(D), B=X(D), C=X(D)) is a relation scheme it is said that A *Interval Functionally Determines* (IFD) B if and only if whenever $(a_1, b_1, c_1) \in R$, $(a_2, b_2, c_2) \in R$ and $a_1 = a_2$ then $b_1 = b_2$." [[27], p. 43] □

Note that the value of attribute C is not used in the definition. Hence this dependency is similar to the snapshot functional dependency, with the added constraint that the left-hand side be an attribute over an interval domain, interpreted as an atomic value.

Of relevance for the next definition, the result of *S-UNFOLD*$[A](R)$, where R is a relation instance and A is an interval-valued attribute of R, is obtained by "expanding" each tuple of R in turn. An argument tuple is expanded by generating one result tuple for each point in its A interval value. The point becomes the new A value of the result tuple which is otherwise left unchanged. For example, unfolding the tuple $([1, 3], b, c)$ yields $\{(1, b, c), (2, b, c), (3, b, c)\}$.

Definition 6 "Let R(A=X(D_1), B=X(D_2), C=X(D_3)) be a relation scheme and let S = *S-UNFOLD*[A](R). It is said that A *Point Functionally Determines* (PFD) B if and only if whenever $(a_1, b_1, c_1) \in S$, $(a_2, b_2, c_2) \in S$ and $a_1 = a_2$ then $b_1 = b_2$." [[27], p. 44] □

In this dependency, an interval is *not* interpreted as an atomic value, but rather as a set of points.

Note that a A PFD implies an IFD, but not vice versa. These dependencies may be combined to treat some interval-valued attributes as atomic and others as sets of points.

These dependencies are intensional properties, but are tied to the representation of a relation. They are used to define temporal keys. Hence they satisfy Desiderata **D1** and **D3**.

2.7 Wijsen's Temporal Dependency Theory

Wijsen and his colleagues have recently extended snapshot dependency theory to an object-based data model, i.e., a data model supporting object-identity [54, 55]. This data model is a sequence of snapshot relations indexed by valid-time. Four types of dependencies are defined: snapshot functional dependencies (SFDs), dynamic functional dependencies (DFDs), temporal functional dependencies (TFDs), and interval dependencies (IDs) [55, 56].

SFDs are intra-state dependencies, i.e, they are defined in terms of a single snapshot in a temporal database. Essentially, an SFD is the conventional functional dependency extended with object-identity.

DFDs, TFDs, and IDs are inter-state dependencies, i.e., they apply to the sequence of snapshots constituting the temporal relation. Like SFDs, these dependencies use object identity. DFDs constrain pairs of adjacent valid-time states; TFDs and IDs constrain a sequence of multiple valid-time states.

In terms of the desiderata, this proposal satisfies Desideratum **D1**, in that the defined dependencies are intensional properties. However, the reliance on object-identity forces the proposal to be representation-dependent. The dependencies are used to specify keys, and so satisfy Desideratum **D3**.

2.8 Summary

We have surveyed several interesting definitions of temporal dependencies. A few, such as Tansel and Garnett and Lorentzos, treat relations with temporal information as snapshot relations with explicit temporal attributes. The remaining define dependencies, specifically Vianu's dynamic dependency, Navathe and Ahmed's temporal dependency, and Wijsen et al.'s DFD, TFD, and ID, are inter-state dependencies, and thus are more ambitious than the intra-state dependencies considered further in this paper.

3 Previous Proposals for Temporal Keys

We now turn to the related topic of defining keys for a temporal data model. We briefly review the notion of keys in conventional databases and the important properties of conventional keys. Then the existing temporal database keys are introduced and contrasted with the properties of conventional keys.

3.1 Conventional Keys

Definition 7 The set of attributes X is a *superkey* of R if $X \rightarrow R$. A superkey is *minimal* if when any attribute is removed, it is no longer a superkey. A relation schema may have many minimal keys, termed *candidate keys*. One such key is selected as the *primary* key. □

Example 5 In the example database, introduced in the previous section, there are two superkeys, {Name, PhNo} and {Name, PhNo, Dept}. Only the former is minimal; hence, it is the primary key, and there are no other candidate keys. □

The following five fundamental properties are held by the definitions of snapshot keys. We find it desirable that temporal keys also have these properties.

K1. Keys are intensional.

K2. Keys are properties of stored (base) relations only.

K3. Particular attributes are not a priori designated as keys.
In some temporal data models, relations have mandatory timestamp attributes. The values of such attributes indicate when the non-temporal attribute values are valid (or current). The desideratum states that these mandatory attributes or other attributes should not be required to always be (part of) keys. Rather, the database designer should be able to choose more freely.

K4. Keys are independent of the representation.
This desideratum, along with the previous one, will be clarified in Section 5.3.

K5. Primary keys are minimal.

Next we examine various proposals for temporal keys. Relation instances are used for illustration, even though the notion of key should be applied to relation schemas during database design. The key attributes are underlined.

3.2 Keys in the Time Relational Model

In Ben-Zvi's pioneering dissertation [4], the standard definition of snapshot key is retained (though the definition is stated informally, without explicit reference to functional dependencies). To explain the notion of temporal key, we must first introduce the notions of tuple-version, tuple-version-set, and time-relation as defined in the Time Relational Model.

Definition 8 "Given a Relation R, a Key K for R, and letting $\mathrm{D}_i = (\mathrm{d}_1,\ \mathrm{d}_2,\ \mathrm{d}_3, \ldots \mathrm{d}_n)$ denote a typical tuple in R. A *Tuple-Version* I_i, is the ordered list: $\mathrm{I}_i = (\mathrm{D}_i,\ \mathrm{T}^i_{es},\ \mathrm{T}^i_{rs},\ \mathrm{T}^i_{ee},\ \mathrm{T}^i_{re},\ \mathrm{T}^i_d)$."
"A *Tuple-Version-Set*, L, is a set of Tuple-Versions $\{\mathrm{I}_i / i = 1, \ldots m\}$, all having the same key value Ki which compose the whole history of a unique tuple; this tuple can be uniquely determined by the key Ki."
"Given a Relation R, a *Time-Relation* R_t is the collection of all tuple-version-sets $\{\mathrm{L}_j\ \ j = 1, \ldots n\}$ constructed from R's tuples." [[4], pp. 47–50] □

The five T attributes used above encode the valid and transaction time of a tuple-version. A time-relation is a set of tuple-version-sets.

A set of attributes K is a *temporal key* of a time-relation, R_t, if (1) the attributes K form a (conventional) key of the corresponding non-temporal relation, R, and (2) the tuple-version-sets in R_t are defined by partitioning tuple-versions so that tuple-versions with identical values for the attributes K are in the same tuple-version-set.

This definitions has several notable properties. Clearly, the definition of a temporal key is intensional and satisfies Desideratum **K1**. As views are not discussed in this context, it is not known whether Desideratum **K2** is satisfied. Unlike conventional relations, a time-relation has precisely one temporal key. This is so because tuple-versions partitioned on a set of attributes K are generally not guaranteed to also be partitioned on any other set of attributes. A temporal key is a priori designated, violating Desideratum **K3**, because a particular temporal key is chosen to achieve a desirable structuring of the tuple-versions. The choice of temporal key is not determined by the representation of a time-relation, so Desideratum **K4** is satisfied. Finally, the notion of minimality of temporal keys for time-relations makes little sense, since a time-relation can only have a single key. The temporal key of a time-relation, R_t, may or may not be a minimal key of the corresponding snapshot relation, R.

3.3 The HQL Data Model

In the data model associated with the query language HQL [36], valid-time relations are represented by snapshot relations where tuples are timestamped with intervals. Thus, a valid-time relation with explicit attributes $A_1, \ldots, A_n$ is represented by a snapshot relation with schema $(A_1, \ldots, A_n, \textbf{start}, \textbf{end})$.

Without providing further explanation of the notion of key, it is required that the attributes **start** and **end** be part of any primary key.

Two points can be made. First, using both timestamp attributes seems unnecessary. Indeed, one of the attributes is redundant, violating the minimality requirement of a primary key. Second, the definition of key appears to be representation-dependent. In summary, this definition may satisfy Desiderata **K1**–**K3** (insufficient discussion makes it impossible to know for sure), but it does not satisfy Desiderata **K4** or **K5**.

Example 6 Consider the following relation instance.

Emp	Dept	**start**	**end**
Bill	Shipping	1	5
Bill	Shipping	5	10

The primary key of the relation schema is {Emp, **start**, **end**}. It may be observed that generally either {Emp, **start**} or {Emp, **end**} are sufficient. □

3.4 The TRM Data Model

A key of a TRM relation schema is defined as follows. In the definition, the time-invariant key (TIK) is the primary key of a snapshot version of the valid-time relation schema.

Definition 9 The candidate keys of a TRM relation schema are (TIK,T_s) or (TIK, T_e), i.e., the snapshot key appended with either the starting or ending timestamp. (TIK,T_s) is designated as the primary key. [32] □

This definition is clearly intensional and therefore satisfies Desideratum **K1**. However, the definition does not satisfy Desideratum **K2** since derived relations have the given key. Similarly, Desiderata **K3** and **K4** are not satisfied since T_s (or T_e) must be part of a candidate key, and the definition is dependent on the given tuple-timestamped representation. We assume Desideratum **K5** is satisfied since the TIK is assumed to be a minimal key.

3.5 The Interval Extended Relational Model

As before, we emphasize that the IXRM is not a temporal data model, in that it supports only user-defined time [27]. It can, however, be used as the representation of a valid-time relation.

Keys are defined in this data model in terms of point and interval functional dependencies, in a manner very similar to snapshot keys. A key is required to be minimal. As Lorentzos mentions "the" key, it is assumed that only the primary key was being defined. This definition of key satisfies all but Desideratum **K4**.

Example 7 Consider a sample relation instance.

Emp	Dept	T
Bill	Shipping	[1,5)
Bill	Shipping	[5,10)
Bill	Shipping	[3,10)

The key of this relation is T, since T determines {Emp, Dept}. □

3.6 The TempSQL Data Model

Gadia and Nair define a special notion of key in the data model associated with the query language TempSQL [17, 31]. To examine this concept, the type of relation employed must be understood first.

Example 8 Consider the following relation instance indicating the managers for departments.

Mgr	Dept
[10, 14] Bill [15, 19] Al	[10, 19] Shipping
[15, 30] Bill	[15, 30] Loading

Attribute values are stamped with finite unions of intervals (i.e., valid-time elements [16]). All information about the Shipping department is contained in the first tuple, which states that Bill was the manager from time 10 to 14 and that Al was the manager from time 15 to 19. □

We now state the definition, then explain it using the example.

Definition 10 "A relation over R, with $K \subseteq R$ as its *key*, is a finite set of non-empty tuples such that no key attribute value of a tuple changes with time, and no two tuples agree on all their key attributes." [[17] p. 10] □

The definition lists two requirements that must be fulfilled for a set of attributes to be a key. In the example, the attribute Dept is a key because for each tuple, there is only one value of attribute Dept and no two tuples have the same value for attribute Dept.

It appears that a key is a property of a relational instance, making the definition extensional. Also, the definition is independent of the notion of temporal functional dependency. The dependencies Dept $\rightarrow$ Mgr and Mgr $\rightarrow$ Dept are assumed to hold, making both Dept and Mgr keys of the schema (Dept, Mgr) in the conventional sense. Yet, in the relation instance above, the attribute Mgr is not a key in the sense defined here. An operator is available that restructures the instance to yield the following, equivalent relation, now with Mgr as the only key.

Mgr	Dept
[10, 30] Bill	[10, 14] Shipping [15, 30] Loading
[15, 19] Al	[15, 19] Shipping

To summarize, this definition of key satisfies Desiderata **K3** and **K4**, but does not satisfy Desiderata **K1** or **K2** (because operators can change the key of a relation). Primary keys are not defined in the model.

3.7 Wijsen's Theory of Keys

In conjunction with their work in temporal dependency theory, Wijsen and his colleagues have developed a notion of keys for temporal relations [54, 55]. Three types of keys, snapshot keys (SK), dynamic keys (DK), and temporal keys (TK), corresponding to the notions of snapshot functional dependency, dynamic dependency, and temporal dependency, respectively, are defined. Recall from Section 2.7 that a major motivation for this work was to incorporate the concept of object-identity into a temporal database. As object-identity is normally a hidden attribute of an object, i.e., is an attribute that cannot be directly referenced or queried, the defined keys attempt to make the identification of tuples belonging to the same object possible.

For example, the snapshot key is (informally) defined as follows.

Definition 11 A snapshot key is a set of attributes that snapshot functionally determines the object-identity of an object, for any snapshot which can be taken from a temporal database. The snapshot key is also minimal. [Rephrasing of definition, [54], p. 15] □

As can be seen from the definition, a snapshot key uses the object-identity in its definition, though the object-identity is not part of the key. Notice also, that the snapshot key is an intra-state key, i.e., the snapshot key does not express constraints between attribute values in snapshots taken at different times.

This definition satisfies all of the desiderata except Desideratum **K4**, since it relies on object-identity.

Dynamic keys and temporal keys are defined using inter-state dependencies. For example, a dynamic key, like a dynamic functional dependency, holds between adjacent states of a temporal database, and a temporal key, like a temporal functional dependency, holds between disjoint intervals of time.

Inter-state dependencies and keys are beyond the scope of this paper. Our temporal key, to be defined later, is very similar to Wijsen's snapshot key, except we do not rely on the presence of an object-identity attribute.

3.8 Summary

We have surveyed several interesting notions of keys. Each of the given proposals adapt the notion of conventional relational keys to temporal databases, but none of the keys individually satisfy all five desiderata in Section 3.1. We base our work on the foundation provided by these previous proposals.

4 Previous Proposals for Temporal Normal Forms

We wish to develop normalization concepts that closely parallel their counterparts in conventional normalization theory. We therefore begin by briefly reviewing the two most important relational normal forms, third normal form and Boyce-Codd normal form. This leads to a formulation of the common aspects of conventional normal forms that we wish our temporal normal forms to possess. Existing temporal normal forms are then introduced and examined with respect to these properties.

4.1 Conventional Normal Forms

A normal form is an *intensional* property of a database schema that follows from a set of (functional, multivalued, or other) dependencies. The goal of database design is to obtain a set of relation schemas that, together with their dependencies, satisfy the normal forms.

We define the two most important normal forms, third normal form [10] and Boyce-Codd normal form [12].

Definition 12 The pair of a relation schema, R, and a set, F, of functional dependencies on R is in *third normal form* (3NF) if for all non-trivial dependencies, $X \rightarrow Y$, in F^+, X is a superkey for R or each attribute in Y is contained in a minimal key for R. □

Definition 13 The pair of a relation schema, R, and a set, F, of functional dependencies on R is in *Boyce-Codd normal form* (BCNF) if for all non-trivial dependencies $X \rightarrow Y$ in F^+, X is a superkey for R. □

The normal forms only allow the existence of certain functional dependencies, making other functional dependencies illegal. As we shall see, illegal dependencies indicate either the need for null values, the possible existence of update anomalies, or the presence of redundant information. By obeying the normal forms, some of these undesired effects are avoided.

Example 9 Returning to the example in Section 2.1, Emp = (Name, Dept, PhNo), we have Name $\rightarrow$ Dept. Since Name is not a superkey, BCNF is violated. As Dept is not part of any minimal key, 3NF is also violated. As we may expect, a database using this schema exhibits several problems.

First, insertion anomalies are possible. If we want to insert the department of an employee but do not know the employee's telephone number, either the information cannot be inserted or the phone number must be represented by a null value. This is also true when we do not know the employee's department. Normal forms attempt to avoid excessive use of null values.

Second, update anomalies are possible through redundant information. For example, whenever a new telephone number is inserted for an employee, the department information must be repeated. Apart from being wasteful of space, this means that whenever an employee switches departments, several tuples, one for each of the employee's telephone numbers, must be updated. If one such tuple is not updated then an inconsistency will be generated in the database. Normal forms attempt to avoid redundancy.

Third, deletion anomalies are possible. Suppose that an employee no longer needs a telephone, and all telephone numbers for that employee are deleted from the database. When the last tuple containing that employee's telephone is deleted, the removal of that tuple results in the loss of the employee's department information. Again, undesirable null values may be used to overcome this problem. □

Decomposition is one way to address these problems, by breaking up a large relational schema into several smaller schemas, each of which satisfy the normal forms.

Example 10 All of the anomalies previously mentioned with the example relational schema are avoided by decomposing the schema into EmpDept $=$ (Name, Dept) and EmpPhNo $=$ (Name, PhNo), both of which are in BCNF. □

In some applications, queries involving a join of two relations occur frequently. As joins are expensive operations, performance considerations may dictate that the relation schemas be merged, even if the resulting schema does not conform to a desirable normal form. Thus, anomalies and redundancy may be tolerated in order to enhance the performance of the database management system.

Additional normal forms exist that are more restrictive than BCNF. For example, fourth normal form [13] is a close parallel of BCNF, but which relies on the notion of multivalued dependencies [60].

We do not address these normal forms, for two reasons. First, they are generally believed to have less relevance in practical database design. Second, no proposals for temporal counterparts of these have been proposed. Still, even though we do not cover these further normal forms in this survey, temporal versions of such normal forms such as fourth normal form may be defined within the framework developed in Section 6.3.

In summary, the fundamental qualities of the conventional normal forms may be outlined as follows.

N1. Normal forms are intensional, not extensional, properties.
As argued before, while this may be a subtle distinction, it is a conceptually important one.

N2. Normal forms are defined solely in terms of dependencies that exist or do not exist.

N3. Normal forms are properties of stored (base) relations only—redundancy and anomaly issues do not apply to (computed) views.
We have seen that normal forms are motivated by the desire to avoid update anomalies and redundancy. These issues are only of interest for base relations as they are the only relations that must be stored, and they are the only relations that can be updated. Normal forms do not apply to views or derived relations, and they are independent of query languages.

N4. Normal forms are defined independently of the representation of a relation. These concepts are based on semantics, not on an arbitrary representation.
The meaning of this desideratum will become clearer once the necessary concepts have been introduced, see Section 5.3.

While conventional normal forms may share additional qualities, these four qualities are all fundamental and are satisfied by all conventional normal forms. We characterize each of the previously proposed temporal normal forms based on these desiderata.

When presenting the various normal forms, it is convenient to illustrate these by means of sample relation instances. While normal forms may be applied to individual instances, we emphasize that normal forms should be applied to relation schemas during database design.

4.2 Time Normal Form

In his Time Relational Model, Ben-Zvi defined the first temporal normal form. The definition employs the concepts of contiguous and non-contiguous time-relations. Intuitively, a time-relation is contiguous if "there are no 'holes' in the effective-time history of each tuple-version set" ([[4], p. 137]; a formal definition is also provided). To illustrate this, consider a tuple-version-set that records the department history of employee Bill. If it is always the case that when Bill resigns from one department, he immediately starts in another department, there are no times (i.e., no "holes") during Bill's employment when he does not have a department value. A time-relation "R_t is *non-contiguous* if it is not contiguous."

Definition 14 "A time-relation R_t is in *time normal form* if Relation R is in *any* normal form and *either* all its attributes are contiguous *or* non-contiguous." [[4], p. 139] □

Here, R is the underlying conventional relation on which R_t is defined (see Section 3.2).

The rationale for this normal form is two-fold. In the example above, if tuple-version-sets are known to be contiguous, users need not explicitly terminate the employment of an employee in one department when recording that the employee is now with another department—the system is capable of doing this. Next, the contiguity may be exploited in the implementation of a time-relation. It is not necessary to explicitly record when an employee left a department as this time may be inferred from the time when the when the employee joins another department.

One confusing aspect of the definition is that it refers to *attributes* as contiguous or non-contiguous, even though these notions are only defined for relations. A second confusion is whether first normal form qualifies. Earlier discussion of R implies that it is in first normal form.

This definition satisfies Desiderata **N1** and **N3**. It does not satisfy Desiderata **N2** (since contiguity is not defined in terms of functional dependencies) or **N4** (since the definition of contiguity is in terms of the representation of a time-relation).

4.3 First Temporal Normal Form

Segev and Shoshani define, in their Temporal Data Model, a normal form, 1TNF, for valid-time relations [42]. To understand this normal form, we need to first describe their data model and the special variant of the timeslice operator employed there.

Valid-time relation schemas have a distinguished, so-called *surrogate*, attribute. Surrogates represent objects in the modeled reality, and the time-varying attribute values in a tuple of a relation instance may be thought of as containing information about the object represented by the surrogate of the tuple.

The special timeslice operator relies on the presence of the surrogate attribute. It takes a valid-time relation and a time value as arguments and returns, for each surrogate value, all the values of each time-varying attribute that are valid at the time given as argument. Thus, the result contains precisely one tuple per surrogate, valued with at least one time varying attribute value, valid at the time argument. As another consequence, time varying attributes may be set-valued, leading to a non-1NF result relation.

Definition 15 For a relation to be in *first temporal normal form* (1TNF), "a time-slice at point t has to result in a standard 1NF-relation." [[42], p. 17] □

In addition to giving a conceptual definition, the authors present two representation-dependent definitions of 1TNF, for valid-time relations represented by snapshot relations using interval and event tuple timestamping, respectively. For the interval-based representation, containing the attributes T_S (the starting valid time) and T_e (the ending valid time), the following definition is given.

Definition 16 "A relation with a schema, $R(S, A_1, \dots, A_n, T_s, T_e)$, is in 1TNF if there do not exist two tuples $r^1(s^1, a_1^1, \dots, a_n^1, t_s^1, t_e^1)$ and $r^2(s^2, a_1^2, \dots, a_n^2, t_s^2, t_e^2)$ such that $s^1 = s^2$ and the intervals $[t_s^1, t_e^1]$ and $[t_s^2, t_e^2]$ intersect." [[42], p. 17] □

Example 11 Consider the following interval timestamped relation instance where the Emp attribute is assumed to contain surrogate values.

Emp	Dept	T_s	T_e
Bill	Shipping	1	4
Bill	Loading	5	10

This relation instance is in 1TNF since no two tuples with the same surrogate have overlapping time intervals. □

The normal form has a specific purpose within the Temporal Data Model. In essence, the model extends the relational model with surrogates. It then proceeds by defining a timeslice operator that uses the surrogates in a way that leads to the possibility of getting set-valued attributes in results of timeslice operations. This normal form is introduced to ensure that the results of timeslice operations are always tuples with atomic attribute values. Thus, 1TNF is required rather than desirable.

This normal form has a different motivation than do conventional normal forms and it is needed because of non-relational extensions in the data model. First, the normal form is extensional—it applies to a relation instance, not a relation schema as do conventional normal forms. Second, the normal form is based on an operator which relies on a designated attribute, the surrogate attribute. In the conventional relational model, no attribute is special.

In summary, 1TNF does satisfy Desideratum **N4**, as a conceptual definition is provided. However, 1TNF does not satisfy Desiderata **N1** (since normal forms are defined on relations, rather than relation schemas), **N2** (though the conceptual definition does employ the notion of snapshot 1NF), or **N3** (since operators are expected to preserve 1TNF, and are only defined over 1TNF relations).

4.4 Time Normal Form

This normal form for valid-time relations also applies to an interval tuple-timestamped representation [32]. Unlike BenZvi's Time Normal Form and Segev and Shoshani's 1TNF, it is based on the notion of temporal dependency, defined in Section 2.

Definition 17 A valid-time relation "is in *time normal form* (TNF) if and only if it is in [snapshot] BCNF and there exists no temporal dependency among its time varying attributes."
[[32], p. 157] □

Example 12 Consider the following relation instance with time-invariant key Emp.

<u>Emp</u>	Dept	Mgr	T_s	T_e
Bill	Shipping	Zoe	1	4
Bill	Loading	Zoe	5	10

Our previous observation that a temporal dependency exists between the Dept and Mgr attributes means that this relation instance violates TNF. □

This definition satisfies Desiderata **N1** and **N2**, but it does not satisfy Desiderata **N3** (because operators are defined only on TNF relations) or **N4**. Also, snapshot normal forms, e.g., BCNF, and therefore snapshot functional dependencies are applied to the representations of valid-time relations [32], violating Desideratum **N4**. The clear distinction between the meaning of and the representation of a valid-time relation is a more recent development in temporal databases.

4.5 The HSQL Data Model

In the valid-time data model associated with the query language HSQL [40], there is an explicit distinction between valid-time relations and their snapshot relation representations. Thus a valid-time relation $\overline{R} = (A_1, \ldots, A_n)$ is represented by a snapshot relation $R = (A_1, \ldots, A_n, \text{PERIOD})$ [39]. It is claimed, but not demonstrated, that conventional normalization techniques apply to the design of a valid-time database. One of the purposes of this paper is to give a formal and concrete characterization of the sense in which this is true.

4.6 P and Q Normal Forms

A so-called P normal form is defined for the Interval Extended Relational Model. We give a simplified definition; the original definition ([27], p. 49) used a rather complex algebraic operator.

Definition 18 The schema of an interval extended relation, representing a valid-time relation, is said to be in *P normal form* (PNF) if, in all extensions of that relation scheme, no two tuples with the same key value have overlapping or adjacent time intervals. □

This normal form satisfies Desiderata **N1** and **N3**. In this temporal context, we use relations in the IXRM for representing valid-time relations, but as discussed above IXRM relations are not valid-time relations. Consequently, PNF does not satisfy Desiderata **N2** or **N4**.

A second normal form is also defined. We now give a simplified version.

Definition 19 The schema of an interval-extended relation, representing a valid-time relation, is said to be in *Q normal form* (QNF) if it is in PNF and the schema contains exactly one non-key attribute. [Rephrasing of [27], p. 51] □

Example 13 Consider the following interval timestamped instance with primary key {Emp, Period}.

Emp	Dept	Mgr	Period
Bill	Shipping	Zoe	[1,5)
Bill	Shipping	Janet	[5,10)

This relation is not in QNF since the Dept and Mgr attributes are both non-key attributes (and since PNF is violated). □

This normal form also satisfies Desiderata **N1** and **N3**. As before, since IXRM relations merely represent valid-time relations and QNF relies on PNF, QNF does not satisfy Desiderata **N2** or **N4**.

4.7 Summary

All existing temporal normal forms known to the authors have been surveyed. While none of them completely satisfied all qualities that could be expected from a natural extension of conventional normal forms, each presented interesting ideas. And together the normal forms provide a platform from which it is possible to reach further.

The data models mentioned in this context present notable exceptions, as the majority of the two dozen temporal data models proposed thus far do not discuss functional dependencies, keys, or normal forms at all.

On the other hand, it would be best to provide *model-independent* definitions. It is generally not possible to apply model-specific definitions (like those surveyed) of functional dependencies or keys to other data models in a straightforward fashion.

5 A Bitemporal Conceptual Data Model

We feel that the reason why so many temporal data models have been proposed, and why so many temporal keys and temporal normal forms have been defined, is that previous models attempted to simultaneously retain the simplicity of the relational model, present all the information concerning an object in one tuple, and ensure ease of implementation and query evaluation efficiency.

It is clear from the number of proposed models that meeting all of these goals simultaneously is a difficult, if not impossible task. We therefore advocate a separation of concerns. The time-varying semantics is obscured in the representation schemes by other considerations of presentation and implementation. We feel that the data model proposed in this section is the most appropriate basis for expressing this semantics. However, in many situations, it is not the most appropriate way to present the stored data to users, nor is it the best way to physically store the data.

We have defined mappings to several representations; these representations may be more amenable to presentation and storage, while retaining the semantics of the conceptual data model.

We first formally characterize a bitemporal relation. Then we define the set of bitemporal algebra operators necessary for the introduction of normal forms. The objects and their operations constitute the *bitemporal conceptual data model*, or BCDM [21]. Finally, we outline a few of the *representational* data models in which instances and operators can be mapped to and from the BCDM.

5.1 Objects in the Model

Tuples in a bitemporal conceptual relation instance are associated with time values from two orthogonal time domains, namely valid time and transaction time. Valid time is used for capturing the time-varying nature of the portion of reality being modeled, and transaction time models the update activity associated with the relation.

For both time domains, we assume that the database system has limited precision; the smallest time units are termed chronons [20]. This restriction greatly simplifies implementation, and since no database system known to the authors supports time domains with unlimited precision, the restriction appears acceptable.

The time domains have total orders and both are isomorphic to subsets of the domain of natural numbers. The domain of valid times may be given as $\mathcal{D}_{VT} = \{t_1, t_2, \dots, t_k\}$, and the domain of transaction times may be given as $\mathcal{D}_{TT} = \{t'_1, t'_2, \dots, t'_j\} \cup \{UC\}$ where UC ("until changed") is a distinguished value that is used for indicating that a tuple is current in the database. A valid-time chronon is thus an element of $\mathcal{D}_{VT}$, a transaction-time chronon is an element of $\mathcal{D}_{TT}$, and a bitemporal chronon is an ordered pair of a transaction-time chronon and a valid-time chronon. We expect that the valid time domain is chosen so that some times are before the current time and some times are after the current time. We also define a set of names $\mathcal{D}_A = \{A_1, A_2, \dots, A_{n_A}\}$ for explicit attributes and a set of attribute domains $\mathcal{D}_D = \{D_1, D_2, \dots, D_{n_D}\}$.

In general, the schema of a bitemporal conceptual relation, R, consists of an arbitrary number, e.g., n, of explicit attributes from $\mathcal{D}_A$ with domains in $\mathcal{D}_D$, and an implicit timestamp attribute, T, with domain $2^{\mathcal{D}_{TT} \times \mathcal{D}_{VT}}$. A set of bitemporal functional (and multivalued) dependencies on the explicit attributes are part of the schema. For now, we ignore these dependencies—they are treated in detail later.

A tuple, $x = (a_1, a_2, \dots, a_n | \ t_b)$, in a bitemporal conceptual relation instance, $r(R)$, consists of a number of attribute values associated with a bitemporal timestamp value. For convenience, we will employ the term "fact" to denote the information recorded or encoded by a tuple, and we say that "a tuple encodes a fact." No additional assumptions are intended by this usage.

An arbitrary subset of the domain of valid times is associated with each tuple, meaning that the fact recorded by the tuple is *true in the modeled reality* during each valid-time chronon in the subset. Each individual valid-time chronon of a single tuple has associated a subset of the domain of transaction times, meaning that the fact, valid during the particular chronon, is *current in the relation* during each of the transaction-time chronons in the subset. Any subset of transaction times less than the current time and including the value *UC* may be associated with a valid time. Notice that while the definition of a bitemporal chronon is symmetric, this explanation is asymmetric. This asymmetry reflects the different semantics of transaction and valid time.

We have thus seen that a tuple has associated a set of so-called *bitemporal chronons* in the two-dimensional space spanned by transaction time and valid time. Such a set is termed a *bitemporal element* [20] and is denoted t_b. Because no two tuples with mutually identical explicit attribute values (termed *value-equivalent*) are allowed in a bitemporal relation instance, the full time history of a fact is contained in a single tuple.

In graphical representations of bitemporal space, we choose the x-axis as the transaction-time dimension, and the y-axis as the valid-time dimension. Hence, the ordered pair (t, v) represents the bitemporal chronon with transaction time t and valid time v.

Example 14 Consider a relation recording employee/department information, such as “Al works for the shipping department.” We assume that the granularity of chronons is one day for both valid time and transaction time, and the period of interest is some given month in a given year, e.g., June 1994. Throughout, we use integers as timestamp components. The reader may informally think of these integers as dates, e.g., the integer 15 in a timestamp represents the date June 15, 1994. The current time is assumed to be 19.

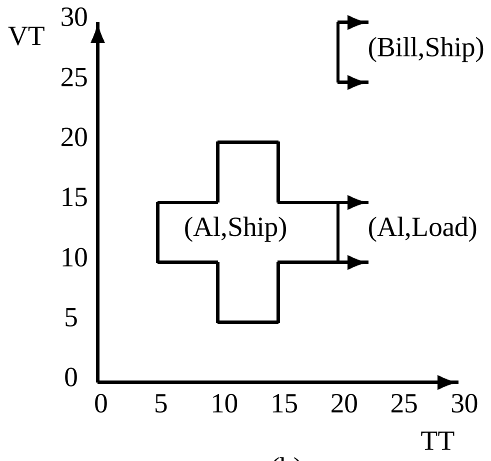
Figure 1(a) shows an instance, empDep, of this relation. A graphical illustration of the empDep relation is shown in Figure 1(b). Right-pointing arrows in the graph and the special value *UC* in the relation signify that the given tuple is still current in the database and that new chronons will be added to the timestamps as time passes and until the tuple is logically deleted.

The relation shows the employment information for two employees, Al and Bill, contained in three tuples. The first two tuples indicate when Al worked for the shipping and loading departments, respectively. These two tuples are shown in the graph as the regions labelled “(Al, Ship),” and “(Al, Load),” respectively. The last tuple indicates when Bill worked for the shipping department, and corresponds to the region of the graph labelled “(Bill, Ship).” □

Depending on the extent of decomposition, a tuple in a bitemporal relation may be thought of as encoding an atomic or a composite fact. We simply use the

Emp	Dept	T
Al	Shipping	$\{(5,10),\ldots,(5,15),\ldots,(9,10),\ldots,(9,15),$ $(10,5),\ldots,(10,20),\ldots,(14,5),\ldots,(14,20),$ $(15,10),\ldots,(15,15)\ldots,(19,10),\ldots,(19,15)\}$
Al	Loading	$\{(UC,10),\ldots,(UC,15)\}$
Bill	Shipping	$\{(UC,25),\ldots,(UC,30)\}$

(a)



(b)

Figure 1: A Conceptual Bitemporal Relation

terminology that a tuple encodes a fact and that a bitemporal relation instance is a collection of (bitemporal) facts.

Valid-time relations and transaction-time relations are special cases of bitemporal relations that support only valid time or transaction time, respectively. Thus a valid-time tuple has associated a set of valid-time chronons (termed a *valid-time element* and denoted t_v), and a transaction-time tuple has associated a set of transaction-time chronons (termed a *transaction-time element* and denoted t_t). For clarity, we use the term *snapshot relation* for a conventional relation. Snapshot relations support neither valid time nor transaction time.

5.2 Operators in the Model

The previous section described the objects in the bitemporal conceptual data model, tuples timestamped with a bitemporal element. We now define some algebraic operators on these objects that will be used in the definition of temporal normal forms. A complete operator set for the BCDM can be found elsewhere [21, 48].

We first define bitemporal analogues of some of the snapshot relational operators, to be denoted with the superscript “$^{\text{B}}$”.

Define a relation schema $R = (A_1, \ldots, A_n|\mathrm{T})$, and let r be an instance of this schema. Let D be an arbitrary set of explicit (i.e., non-timestamp) attributes of relation schema R. The projection on D of r, $\pi^{\mathrm{B}}_D(r)$, is defined as follows.

$$\begin{aligned}\pi^{\mathrm{B}}_D(r) = \{z^{(|D|+1)} \mid {} & \exists x \in r\ (z[D] = x[D]) \wedge \\ & \forall y \in r\ (y[D] = z[D] \Rightarrow y[\mathrm{T}] \subseteq z[\mathrm{T}]) \wedge \\ & \forall t \in z[\mathrm{T}]\ \exists y \in r\ (y[D] = z[D] \wedge t \in y[\mathrm{T}])\}\end{aligned}$$

The first line ensures that no chronon in any value-equivalent tuple of r is left unaccounted for, and the second line ensures that no spurious chronons are introduced.

Let P be a predicate defined on $A_1, \ldots, A_n$. The selection P on r, $\sigma^{\mathrm{B}}_P(r)$, is defined as follows.

$$\sigma^{\mathrm{B}}_P(r) = \{z \mid z \in r\ \wedge\ P(z[A])\}$$

As can be seen from the definition, $\sigma^{\mathrm{B}}_P(r)$ simply performs the familiar snapshot selection, with the addition that each selected tuple carries along its timestamp T.

In the bitemporal natural join, two tuples join if they match on the join attributes and have overlapping bitemporal element timestamps. Define r and s to be instances of R and S, respectively, and let R and S be bitemporal relation schemas given as follows.

$$\begin{aligned} R &= (A_1, \ldots, A_n, B_1, \ldots, B_l \mid \mathrm{T}) \\ S &= (A_1, \ldots, A_n, C_1, \ldots, C_m \mid \mathrm{T}) \end{aligned}$$

The bitemporal natural join of r and s, $r \bowtie^{\mathrm{B}} s$, is defined below. As can be seen, the timestamp of a tuple in the join-result is computed as the intersection of the timestamps of the two tuples that produced it.

$$\begin{aligned} r \bowtie^{\mathrm{B}} s = \{z^{(n+l+m+1)} \mid \exists x \in r\ \exists y \in s\ (& x[A] = y[A] \wedge x[\mathrm{T}] \cap y[\mathrm{T}] \neq \emptyset \wedge \\ & z[A] = x[A] \wedge z[B] = x[B] \wedge \\ & z[C] = y[C] \wedge z[\mathrm{T}] = x[\mathrm{T}] \cap y[\mathrm{T}])\} \end{aligned}$$

We have only defined operators for bitemporal relations. The similar operators for valid-time and transaction-time relations are special cases. The valid and transaction-time natural joins are denoted $\bowtie^{\mathrm{V}}$ and $\bowtie^{\mathrm{T}}$, respectively. The same naming convention is used for the remaining operators.

Finally, we define two operators that select on valid time and transaction time. Let t_1 denote a time value not exceeding the current time and let t_2 denote an arbitrary time. The *transaction-timeslice* operator (ρ^{B}) evaluates to a relation timestamped with valid-time elements[1]; and the *valid-timeslice* operator (τ^{B}) yields a relation timestamped with transaction-time elements.

$$\begin{aligned} \rho^{\mathrm{B}}_{t_1}(r) = \{z^{(n+1)} \mid \exists x \in r\ (z[A] = x[A] \wedge z[\mathrm{T}_v] = \{t_2 | (t_1, t_2) \in x[\mathrm{T}]\} \wedge z[\mathrm{T}_v] \neq \emptyset)\} \\ \tau^{\mathrm{B}}_{t_2}(r) = \{z^{(n+1)} \mid \exists x \in r\ (z[A] = x[A] \wedge z[\mathrm{T}_t] = \{t_1 | (t_1, t_2) \in x[\mathrm{T}]\} \wedge z[\mathrm{T}_t] \neq \emptyset)\} \end{aligned}$$

[1]Operator ρ was originally termed the *rollback* operator, hence its name.

Here, $\rho^{\mathrm{B}}_{t_1}(r)$ simply returns all tuples in r that were current during the chronon t_1. The timestamp of a returned tuple is set to all valid-time chronons associated with t_1. Operator $\tau^{\mathrm{B}}_{t_2}(r)$ performs the same operation except the roles of t_1 and t_2 are reversed.

Example 15 Consider the empDep relation shown in Figure 1(a). The following result is produced by $\tau^{\mathrm{B}}_{12}(\text{empDep})$.

Emp	Dept	T$_t$
Bill	Ship	$\{5, \ldots, 19\}$
Bill	Load	$\{UC\}$

Using the graphical representation, valid timeslice can be visualized by drawing a horizontal line through the graph at the given valid time. The tuples returned are those that overlap with the drawn line. The timestamps of the returned tuples are set to the segments of transaction time corresponding to the overlapped regions. □

There also exist timeslice variants that extract a snapshot relation from valid-time relations and transaction-time relations. To extract from r the tuples current at time t_1 and valid in the database during t_2 (termed a *snapshot* of r), it is immaterial whether r is first transaction timesliced and then valid timesliced or first valid timesliced and then transaction timesliced. In the following, we will use the former order, i.e., use $\tau^{\mathrm{V}}_{t_2}(\rho^{\mathrm{B}}_{t_1}(r))$ where superscript "$^{\mathrm{V}}$" indicates a valid-time operator, to produce a snapshot from r.

Note that since relations in the data model are *homogeneous*, i.e., all attribute values in a tuple are associated with the same timestamp [16], the valid or transaction timeslice of a relation will not introduce any nulls into the resulting relation.

5.3 Summary

We have previously described the role of the BCDM in the context of a temporal DBMS where data models are needed for several tasks [21]. Specifically, the BCDM is intended to provide the conceptual model that the query language, e.g., TSQL2, is based on. Other, so-called representational data models are better suited for the tasks of physical storage or data display and are utilized for those tasks. As a consequence the conceptual database schema, designed using normalization techniques to be described in the next section, is captured in the context of the BCDM.

The integration of several temporal data models within the same DBMS hinges on the concept of *snapshot equivalence*, to be defined in Section 7. Snapshot equivalence is a formalization of the notion that two temporal relations have the same information content, and it provides a natural means of comparing rather disparate representations. We have previously developed mappings, respecting snapshot equivalence, between instances of the BCDM and instances of each of the five existing bitemporal data models: a 1NF tuple timestamped data model [44], a data model based on 1NF timestamped change requests recorded in backlog relations [18], a non-1NF data model in which attribute values were stamped with rectangles in transaction-time/valid-time space [17] (discussed in Section 3.6), a bitemporal data model where a bitemporal relation is a sequence of non-1NF tuples [29, 30], and a 1NF data model using five timestamps [4] (discussed in Sections 3.2 and 4.4). We also showed how the relational algebraic operators defined in the previous section could be mapped to analogous operators in the representational models.

A database designer would design the conceptual schema of the database as a (normalized) collection of BCDM relation schemas. The mappings then make it possible to store and display BCDM relations as snapshot equivalent instances of other data models. In the next section, we show how existing dependency theory generalizes naturally to the BCDM. Defining dependencies in terms of BCDM schemas, which are purely conceptual and not intended for implementation, satisfies the desiderata in Sections 2.1, 3.1 and 4.1, respectively, that dependencies, keys, and normal forms be independent of a particular representation of a temporal relation.

6 Generalizing Dependency and Normal Form Theory

In this section we generalize in turn the concepts of functional dependencies, keys, and the normal forms themselves.

6.1 Temporal Dependencies

Functional dependencies are *intensional*, i.e., they apply to every possible extension. This intuitive notion already encompasses time, for a functional dependency may be interpreted as applying at any time in reality and for any stored state of the relation.

To be more specific, consider the restricted case of a transaction-time relation r, with schema $R = (A_1, \dots, A_n|\mathrm{T}_t)$, and a parallel snapshot relation r' with the same schema (but without the implicit timestamp attribute): $R' = (A_1, \dots, A_n)$. The current state of r will faithfully track the current state of r'. Past states of r' will be retained in r, and can be extracted via the appropriate timeslice operator. A functional dependency on R' will hold for all possible extensions, and hence for all past states of r'. Hence, the same functional dependency must hold for all snapshots of r (this insight first appeared over a decade ago [9]). A similar argument can be applied to valid-time relations and to bitemporal relations, yielding the following characterization.

Definition 20 Let X and Y be sets of non-timestamp attributes of a temporal relation schema, R^B, and let t_1 and t_2 be arbitrary times, with t_1 not exceeding the current time. A *temporal functional dependency*, denoted $X \overset{\text{T}}{\rightarrow} Y$, exists on R^B if, for all meaningful instances r^B of R^B,

$$\forall t_1, t_2 \ \forall s_1, s_2 \in \tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r^B)) \ (s_1[X] = s_2[X] \Rightarrow s_1[Y] = s_2[Y]). \qquad \square$$

Note that temporal functional dependencies are generalizations of conventional functional dependencies. In the definition of a temporal functional dependency, a temporal relation is perceived as a collection of snapshot relations. Each such snapshot of any extension must satisfy the corresponding functional dependency.

Also note that this definition applies equally well to valid-time, transaction-time, and bitemporal relations, utilizing the relevant variants of the transaction and valid timeslice operators. While we differentiate valid-time, transaction-time, and bitemporal operator variants, the temporal functional dependency is generic, applying to all forms of temporal relations, with the appropriate operator variants coming into play. The "T" designation in a temporal functional dependency refers to the generic adjective "temporal."

The close parallel between conventional functional dependencies and temporal functional dependencies means that inference rules such as Armstrong's axioms have close temporal counterparts that play the same role in the temporal context as do the non-temporal rules in the non-temporal context.

Example 16 Consider again the database associating phone numbers, departments, and employees in a company. While employees come and go, and phones are added and dropped as needed, at any one time an employee can belong to only one department, and may have zero, one, or several phone numbers. Expressing these properties using temporal dependencies, we have simply Name$\overset{\text{T}}{\rightarrow}$Dept. $\square$

Temporal multivalued dependencies [60] may be defined using the same template as that used for defining temporal functional dependencies.

Snapshot dependencies apply to snapshot relations, and temporal dependencies apply to temporal relations. Further, a snapshot relation records information that is currently believed to currently be true. A bitemporal relation with the same explicit attributes is capable of also recording previous beliefs and beliefs about the past and future. In this sense, a bitemporal relation schema may record more information than its snapshot counterpart.

If a temporal relation schema is used for recording the same information as its snapshot counterpart, a snapshot functional dependency on the snapshot schema implies the corresponding temporal functional dependency on the temporal relation, and vice versa. To make this correspondence between snapshot functional dependencies and temporal functional dependencies more precise, it is practical to first make precise the notion of snapshot and temporal relation schemas recording the same information.

Definition 21 Let $R = (A_1, A_2, \dots, A_n)$ be a snapshot relation schema and $R^B = (A_1, A_2, \dots, A_n|\text{T})$ be a bitemporal relation schema. Let r and r^B range over all possible instances of schemas R and R^B, respectively, and let t_1 and t_2 be arbitrary times where t_1 does not exceed the current time. Then, schemas R and R^B are said to *record corresponding information* if the following three conditions are satisfied.

$$
\begin{aligned}
&(1) \quad \forall r^B\ \forall t_1, t_2\ (t_1 \neq t_2 \Rightarrow \tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r^B)) = \emptyset)\\
&(2) \quad \forall r^B\ \exists r\ (r = \tau^{\text{V}}_{now}(\rho^{\text{B}}_{now}(r^B)))\\
&(3) \quad \forall r\ \exists r^B\ (\tau^{\text{V}}_{now}(\rho^{\text{B}}_{now}(r^B)) = r)
\end{aligned}
$$
□

Note that the first condition restricts instances of R^B to only record information that is valid precisely when it is also current in the database. Using this definition, the following relationship exists between snapshot and temporal functional dependencies.

Theorem 1 Let R be a snapshot relation schema, and let X and Y be subsets of the attributes of R. Also, let R^B be a bitemporal relation schema with same explicit attributes as R. Let the two schemas record corresponding information. Then $X \to Y$ holds for R if and only if $X \overset{\text{T}}{\to} Y$ holds for R^B.

PROOF: The two directions of implication are shown in turn. To show the first, we assume that $X \to Y$ holds on R and show that an arbitrary instance r^B of R^B satisfies $X \overset{\text{T}}{\to} Y$. To show this, we must show that for all t_1 and t_2, $\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r^B))$ satisfies $X \to Y$. From the premise of the theorem, it follows immediately that this is the case for $t_1 = t_2 = now$ and for $t_1 \neq t_2$. Now consider the remaining case, where $t_1 = t_2 \neq now$. We must show that for all t, $r_t = \tau^{\text{V}}_t(\rho^{\text{B}}_t(r^B))$ satisfies $X \to Y$. Again by the premise, R and R^B also recorded corresponding information at (any) time t. At time t, $now = t$, so the definition of the premise implies that r_t is identical to some instance of R. As r satisfies the dependency, so does r_t.

The second direction of implication is straightforward. If $X \overset{\text{T}}{\to} Y$ holds for R^B then for all instances $r^B(R^B)$ and times t_1 and t_2, $X \to Y$ holds for $\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r^B))$. Since R and R^B record corresponding information, each instance of $r(R)$ is identical to some timeslice of an instance of R^B and thus satisfies the dependency. □

It is important to note that two separate data models are involved here. The dependency $X \to Y$ applies to the *snapshot* data model only, whereas $X \overset{\text{T}}{\to} Y$ applies to temporal data models: valid-time, transaction-time, and bitemporal data models. The theorem gives specific content to the statement that the notion of temporal functional dependency as defined in this paper are natural generalizations of the well-known notion of a snapshot functional dependency.

However, it is *not* always the case that functional dependencies on snapshot schemas generalize to *snapshot* functional dependencies on temporal schemas, even when the timestamp attribute is factored in (cf., [39]). Assume that $(A_1, \dots, A_n|\text{T})$

is the schema for a temporal relation R^B. An instance of R^B can be interpreted in two rather different ways: as an instance in the bitemporal conceptual data model, where the timestamp attribute is implicit and is accorded a special semantics, or as an instance in the snapshot data model, with schema $(A_1, \ldots, A_n, \mathrm{T})$, where T is simply another explicit attribute. We can compare functional dependencies in the two interpretations.

Theorem 2 With X and Y denoting arbitrary non-timestamp attributes of a relation schema,

$$X \cup \{\mathrm{T}\} \rightarrow Y \not\Rightarrow X \overset{\mathrm{T}}{\rightarrow} Y.$$

PROOF: The following instance satisfies Emp $\cup$ {T} $\rightarrow$ Dept but not Emp$\overset{\mathrm{T}}{\rightarrow}$Dept. Note that the conventional functional dependency treats T as just another attribute, with values such as “10 – 25” being atomic (e.g., like strings).

Emp	Dept	T
Bill	Shipping	10 – 25
Bill	Loading	15 – 30

Note that the implication does hold when $Y \subseteq X$. □

The problem is that the timestamp attribute is considered to be atomic by the snapshot functional dependency.

It turns out, however, that the converse *does* hold.

Theorem 3 Letting X and Y be sets of non-timestamp attributes of a relation schema,

$$X \overset{\mathrm{T}}{\rightarrow} Y \Rightarrow X \cup \{\mathrm{T}\} \rightarrow Y.$$

PROOF: Assume that $X \overset{\mathrm{T}}{\rightarrow} Y$ holds in R^B and let r^B be an arbitrary instance of R^B. Assume that $X \cup \{\mathrm{T}\} \rightarrow Y$ does *not* hold, i.e., that there exist two separate tuples s_1 and s_2 in r^B such that $s_1[X \cup \{\mathrm{T}\}] = s_2[X \cup \{\mathrm{T}\}]$ but $s_1[Y] \neq s_2[Y]$. Let (t_1, t_2) be a bitemporal chronon in $s_1[\mathrm{T}]$, and let $s_1' = s_1[A_1, \ldots A_n]$ and $s_2' = s_2[A_1, \ldots A_n]$. By construction, $s_1', s_2' \in \tau^{\mathrm{V}}_{t_2}(\rho^{\mathrm{B}}_{t_1}(r))$. However, $s_1'[X] = s_2'[X]$ and by assumption $s_1[Y] \neq s_2[Y]$, and hence $X \overset{\mathrm{T}}{\rightarrow} Y$ is not satisfied. The implication of the theorem is hence true by contradiction. □

6.2 Temporal Keys

Since temporal functional dependencies are a natural extension of conventional functional dependencies, definition of the concepts of temporal keys and temporal closure are straightforward. For that reason, the presentation is brief.

Definition 22 A set of attributes X of a temporal relation schema is a *temporal superkey* of R if $X \overset{\mathrm{T}}{\rightarrow} R$. The *primary temporal key* is a minimal temporal superkey. □

Example 17 Considering again the Emp relation schema introduced in Section 2.1, we see that there are two temporal superkeys, {Name, PhNo} and {Name, PhNo, Dept}, with the former being minimal, and thus serving as the primary temporal key. □

As with functional dependencies, snapshot keys generalize to temporal keys, but only when using temporal dependencies. Specifically, if X is the primary key of the (snapshot) relation schema R, then X is also the primary temporal key, but if $X \cup \{\mathrm{T}\}$ is the (snapshot) primary key of the representation of the temporal relation, it may not be the case that X is a temporal key.

6.3 Temporal Normal Forms

We can now generalize snapshot normal forms in a manner similar to generalizing keys.

Definition 23 A pair (R, F) of a temporal relation schema R and a set of associated temporal functional dependencies F is in *temporal third normal form* (T3NF) if for all non-trivial temporal functional dependencies $X \overset{\mathrm{T}}{\rightarrow} Y$ in F^+, X is a temporal superkey for R or each attribute of Y is part of a minimal temporal key of R. □

Definition 24 A pair (R, F) of a temporal relation schema R and a set of associated temporal functional dependencies F is in *temporal Boyce-Codd normal form* (TBCNF) if for all non-trivial temporal functional dependencies $X \overset{\mathrm{T}}{\rightarrow} Y$ in F^+, X is a temporal superkey for R. □

The comments made in connection with dependencies in Section 6.1 about the inadequacy of using snapshot definitions incorporating the timestamp attribute apply here as well. For example, Theorem 1 can be generalized to R is in BCNF if and only if R^B is in TBCNF.

Example 18 The relational schema Emp = (Name, Dept, PhNo |T) violates both T3NF and TBCNF. □

These definitions are based on the temporal functional dependencies described in Section 6.1, which, in turn, were extensions of the snapshot functional dependencies.

As in our definitions, Tansel and Garnett [50] adapt well-understood snapshot techniques to a temporal setting, but the two approaches are quite different. In Section 2.5, we saw how Tansel and Garnett applied snapshot dependency theory directly to support normalization for their nested valid-time relations. Tansel and Garnett do not define new temporal dependencies; rather, they use conventional snapshot dependencies on timestamped attributes. Essentially, they have embedded a valid-time model within a nested snapshot model, and then directly applied conventional dependency and normalization techniques. In contrast, the temporal normal forms defined above are based on a temporal data model, the BCDM, where conventional dependency theory and normalization concepts do not directly apply, but must first be extended temporally.

Temporal versions of other conventional normal forms based on functional and multivalued dependencies may be expressed analogously, e.g., second normal form and fourth normal form. One can also define temporal variants of join dependencies [34], fifth normal form (also called project-join normal form) [14], embedded join dependencies [13], inclusion dependencies [7], template dependencies [38], domain-key normal form [15], and generalized functional dependencies [37]. The extensions exploit the intensional quality of these properties (i.e., applying to every extension implies applying over all time), as well as the simplicity of the bitemporal conceptual data model.

6.4 Evaluation

It should be clear from the preceding discussion that the bitemporal conceptual data model, with its associated definitions of functional dependency and normal forms, satisfies all desiderata listed in Section 2.1. It should also be evident that the definition of key in this model satisfies all five desiderata listed in Section 3.1, and that TBCNF and T3NF satisfy all four desiderata listed in Section 4.1.

We now briefly compare our approach in turn to each of the previously proposed definitions of temporal normal forms and temporal keys.

The purpose of Ben-Zvi's Time Normal Form [4] was to make updates more user friendly and to aid in chosing a space efficient internal representation of a time-relation. The normal form required the "corresponding" snapshot relation to be in any normal form. It also utilized the concept of contiguity, which does not rely on any notion of dependency.

The normal form 1TNF was introduced as a requirement to relations in the Temporal Data Model [42] that ensures that the results of applying a special valid-timeslice operator are 1NF relations. Without this requirement, non-1NF results are possible because the definition of the operator relies on the presence of a distinguished surrogate attribute [42]. The timeslice operators defined in Section 5.2 do not rely on any distinguished attribute and always returns 1NF relations. In our framework, 1TNF may be defined as follows: A relation schema R is in 1TNF if the surrogate attribute S is a temporal key, i.e., $S \overset{\mathrm{T}}{\rightarrow} R$. Thus, 1TNF is an application of the concept of a key.

Time normal form (TNF) was defined to ensure that time-varying attributes were synchronous, i.e., change at the same time [32]. This aspect is not accommodated in our definitions of temporal normal forms.

Using our definition of temporal key, P Normal Form (PNF) [27] will automatically be satisfied. In our framework, PNF is thus also an application of the concept of a key. Q Normal Form appears to have similarities with Navathe's concept of synchrony, and in any case is not accommodated in our definitions.

The snapshot normal forms were also applied to the representations of valid-time relations in several data models [32, 39, 27, 50]. This contrasts our framework, where temporal normal forms are applied to conceptual temporal relations.

Concerning keys, we formalized and extended the notions present in the HQL [36], HSQL [39], and IXRM [27] data models, using the more general concept of temporal dependency. The concept of key in the TempSQL data model [17] appears to be inconsistent with the concept of a snapshot key.

7 Properties of Temporal Normal Forms

In conventional database design, the notions of lossless-join and dependency preserving decomposition are essential. This section covers issues related to these notions in the temporal context.

During database design a conventional (i.e., non-temporal) relation schema is brought to satisfy a normal form by decomposing it. A decomposition should have two important properties. First, the decomposition should be *lossless*, i.e., the contents of the original relation should be available simply by performing a natural join on the new relations, permitting the decomposition to be reversed without loss of information. More formally, a decomposition of schema R is lossless if every extension of R is the natural join of its projection onto the schemas resulting from the decomposition.

Definition 25 Let X and Y be arbitrary sets of non-timestamp attributes of a temporal relation schema R. Then the pair X, Y is a *lossless-join decomposition* with respect to the join $\bowtie$ if, for all $r(R)$ that satisfy the set of functional dependencies on R,

$$r = \pi_X(r) \bowtie \pi_Y(r). \qquad \square$$

It is possible to guarantee that a given decomposition is lossless. This condition is used to guide the decomposition process, ensuring that the generated decompositions are practical. Assume that a single schema is decomposed into two smaller schemas. If both of the smaller schemas contain a superkey of one of the smaller schemas then the decomposition is guaranteed to be lossless.

Theorem 4 The decomposition X, Y of a relation schema R with a set of functional dependencies F is lossless (w.r.t. $\bowtie$) if

$$X \cap Y \rightarrow X \in F^+ \text{ or } X \cap Y \rightarrow Y \in F^+ .$$

PROOF: The proof may be found elsewhere [51, 24]. □

Second, the decomposition should be *dependency preserving,* in that it must be possible to ensure that all dependencies are preserved when a relation is updated without requiring any joins to be performed.

Definition 26 A decomposition $D = \{R_1, \ldots, R_m\}$ of R is *dependency preserving* with respect to a set of functional dependencies F if

$$(\pi_{R_1}(F) \cup \cdots \cup \pi_{R_m}(F))^+ = F^+.$$

Here, $\pi_{R_i}(F)$ denotes the set of functional dependencies from F defined on the attributes of R_i [51]. □

Some decomposition algorithms can be proven to be dependency preserving; others jettison this property in favor of more desirable ones, such as the lossless-join property. For example, Korth and Silberschatz present a simple 3NF decomposition algorithm that preserves dependencies, and they present a BCNF decomposition algorithm that generally does not preserve dependencies (but always yields a lossless-join decomposition) [24]. Note that BCNF is more restrictive than 3NF and therefore avoids more redundancy than does 3NF. While it is always possible to obtain a 3NF decomposition that is dependency preserving and lossless, such is not the case for BCNF. If a dependency preserving BCNF decomposition is not possible, 3NF is usually preferred, at the risk of added data redundancy.

We now apply these concepts to temporal relations. Specifically, we utilize the temporal natural join operator to identify such lossless join decompositions.

Definition 27 Let X and Y be arbitrary sets of explicit attributes of a temporal relation schema R. Then the pair X, Y is a *lossless-join decomposition* with respect to the join $\bowtie^{\text{B}}$ if for all $r(R)$ that satisfy the set of temporal functional dependencies on R, $r = \pi^{\text{B}}_X(r) \bowtie^{\text{B}} \pi^{\text{B}}_Y(r)$. □

Next, we set the stage for proving the temporal equivalent of Theorem 4. In doing so, we first define two bitemporal relation instances, r and s, to be *snapshot equivalent*, $r \overset{\text{S}}{\equiv} s$, if for all times t_1 not exceeding the current time and for all times t_2, $\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r)) = \tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(s))$. We have shown elsewhere [21] that the notions of snapshot equivalence and identity of relations coincide in the BCDM model, i.e., that $r_1 = r_2$ is equivalent to $r_1 \overset{\text{S}}{\equiv} r_2$, with r_1 and r_2 being relation instances in the BCDM. We also need the following result, which states that the temporal join and projection operators reduce to, or are natural generalizations of, their snapshot counterparts.

LEMMA: For bitemporal relation instances r and s as given in the definitions of temporal projection and join (in Section 5.2) and times t_1 and t_2, t_1 not exceeding the current time,

$$\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(\pi^{\text{B}}_D(r))) \overset{\text{S}}{\equiv} \pi_D(\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r)))$$

$$\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r \bowtie^{\text{B}} s)) \overset{\text{S}}{\equiv} \tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r)) \bowtie \tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(s))$$

PROOF: Proofs are omitted for brevity. A proof of a property similar to the second may be found elsewhere [21]. □

We can now prove the temporal equivalent of Theorem 4. As a result of this theorem, the algorithms for normal form decomposition in conventional relational databases are applicable to temporal databases as well.

Theorem 5 The decomposition X, Y of a temporal relation schema, R, with a set of temporal dependencies, F, is lossless (w.r.t. $\bowtie^{\mathrm{B}}$) if

$$X \cap Y \overset{\mathrm{T}}{\rightarrow} X \in F^{+} \text{ or } X \cap Y \overset{\mathrm{T}}{\rightarrow} Y \in F^{+}.$$

PROOF: Let r be an instance of R (i.e., that satisfies F). Showing that the definition of lossless holds is equivalent to showing that $r \overset{\mathrm{S}}{\equiv} \pi_X^{\mathrm{B}}(r) \bowtie^{\mathrm{B}} \pi_Y^{\mathrm{B}}(r)$, which, in turn, is equivalent to showing that for all times t_1 not exceeding the current time and times t_2,

$$\tau_{t_2}^{\mathrm{V}}(\rho_{t_1}^{\mathrm{B}}(r)) = \tau_{t_2}^{\mathrm{V}}(\rho_{t_1}^{\mathrm{B}}(\pi_X^{\mathrm{B}}(r) \bowtie^{\mathrm{B}} \pi_Y^{\mathrm{B}}(r))).$$

From the premise and the definition of temporal functional dependency, we have that $\tau_{t_2}^{\mathrm{V}}(\rho_{t_1}^{\mathrm{B}}(r))$ satisfies $X \cap Y \rightarrow X$ or $X \cap Y \rightarrow Y$. Theorem 4 then applies, meaning that $\tau_{t_2}^{\mathrm{V}}(\rho_{t_1}^{\mathrm{B}}(r)) = \pi_X(\tau_{t_2}^{\mathrm{V}}(\rho_{t_1}^{\mathrm{B}}(r))) \bowtie \pi_Y(\tau_{t_2}^{\mathrm{V}}(\rho_{t_1}^{\mathrm{B}}(r)))$. Next, it follows from the two properties stated in the previous lemma that this is equivalent to the right-hand side of the formula displayed above. All of this holds for arbitrary times t_1 and t_2, and the theorem follows. □

With the property of snapshot equivalence, we next define the property of snapshot subset.

Definition 28 A temporal relation instance r is a *snapshot subset* of a temporal relation instance s, $r \overset{\mathrm{S}}{\subseteq} s$, if $\tau_{t_2}^{\mathrm{V}}(\rho_{t_1}^{\mathrm{B}}(r)) \subseteq \tau_{t_2}^{\mathrm{V}}(\rho_{t_1}^{\mathrm{B}}(s))$ holds for all times t_1 not exceeding the current time and all times t_2. □

Unlike for snapshot equivalence, $r_1 \overset{\mathrm{S}}{\subseteq} r_2$ does not imply $r_1 \subseteq r_2$. For example, let $r_1 = \{(\text{Bill}|\{(5, 10)\})\}$ and $r_2 = \{(\text{Bill}|\{(5, 10), (5, 11)\})\}$. Then $r_1 \overset{\mathrm{S}}{\subseteq} r_2$, but $r_1 \not\subseteq r_2$ as the two relations contain distinct tuples.

The following theorem states three additional properties of the temporal natural join. The snapshot natural join has a parallel for each of these properties. For example, the first property states that in general, a decomposition is *lossy,* i.e., may produce additional, spurious tuples that makes it impossible to identify the true information.

Theorem 6 Let r be a bitemporal relation instance of a schema that includes the sets X and Y of non-timestamp attributes. Also let γ be an instance of a bitemporal relation schema with precisely the non-timestamp attributes X, and let λ be an arbitrary relation instance. The following three properties hold.

$$\begin{aligned} r &\overset{\text{S}}{\subseteq} \pi^{\text{B}}_X(r) \bowtie^{\text{B}} \pi^{\text{B}}_Y(r) \\ \pi^{\text{B}}_X(r) &\overset{\text{S}}{\subseteq} \pi^{\text{B}}_X(\pi^{\text{B}}_X(r) \bowtie^{\text{B}} \pi^{\text{B}}_Y(r)) \\ \pi_X(\gamma \bowtie^{\text{B}} \lambda) &\overset{\text{S}}{\subseteq} \gamma \end{aligned}$$

PROOF: We consider the first property only; the other properties may be proven in similar fashion. For any snapshot relation r^s, it is the case that $r^s \subseteq \pi_X(r^s) \bowtie \pi_Y(r^s)$. In particular, this holds for r^s equal to $\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r))$, for arbitrary times t_1 and t_2, with t_1 not exceeding the current time. From the lemma also used in the previous theorem, it follows that $\pi_X(\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r))) \bowtie \pi_Y(\tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(r))) = \tau^{\text{V}}_{t_2}(\rho^{\text{B}}_{t_1}(\pi^{\text{B}}_X(r) \bowtie^{\text{B}} \pi^{\text{B}}_Y(r)))$. The first property then follows from the definition of snapshot subset just given. □

In an entirely analogous way, by using the modified version of the relational operators given in Section 5.2 and the concept of snapshot equivalence, one can extend other properties of functional dependencies to hold for temporal functional dependencies. Also, every concept defined above is applicable, as special cases, to both valid-time and transaction-time relations, using the appropriate temporal operators.

Our approach uses the bitemporal conceptual data model, along with the timeslice operators ρ and τ, to define the notion of a temporal functional dependency. It is possible to map such dependencies into representational data models. Specifically, if appropriate valid and transaction timeslice operators are defined in the representational model, then the definition of temporal functional dependency and the various temporal normal forms apply directly to that model.

Elsewhere [21] we have provided timeslice operators for the popular data models of tuple timestamping (e.g., [4, 32, 35, 40, 44, 46]), backlogs (e.g., [23, 19]), and attribute value timestamping (e.g., [8, 49, 16, 25, 29, 30]). Those operators, combined with the definitions provided in this paper, enable model-specific definitions of temporal functional dependencies, keys, and normal forms.

The result is a consistent and wholesale application of existing dependency and normalization theory to valid-time, transaction-time, and bitemporal databases in a wide variety of temporal relational data models.

8 Application to Spatial Databases

The graphical representation of a bitemporal element as an area in the two-dimensional valid-time/transaction-time space (see Figure 1(b)) leads one to consider spatial databases, which are either two dimensional (e.g., index by latitude and longitude over the surface of the Earth, cf., [28]) or three dimensions (e.g., the third dimension being altitude or depth, cf., [22]). In fact, the entire discussion of generalizing normal form and dependency theory to accommodate time can be applied to space. In this section, we outline this correspondence.

Each relation in the spatial data model would be "space-stamped" with an implicit *spatial element* S, which is a set of n-dimensional *quanta* (the spatial analogue of the temporal "chronon"). For two-dimensional modeling, *bispatial elements* would be used; for three-dimensional modeling, *trispatial elements* would be used. Spatial extensions of the relational operators could be defined. For example, x-slice and y-slice operators, analogous to valid and transaction timeslice, could be defined. The *spaceslice* of a relation r, then, is a relation containing the tuples in r that apply to specified values of x and y.

The functional dependency $X \rightarrow Y$ can be generalized to a *spatial functional dependency*, denoted $X \overset{\text{SP}}{\rightarrow} Y$, by formalizing the dependency predicate to apply to all space slices of all possible extensions, as well as a *spatiotemporal functional dependency*, denoted $X \overset{\text{SP-T}}{\rightarrow} Y$, that would take all space slices and time snapshots of all possible extensions. The spatial and spatiotemporal functional dependencies introduced here are highly restricted, as they are defined in terms of single space slices and time snapshots.

Like the temporal functional dependency, these dependencies would be natural generalizations of the snapshot functional dependency. More specifically, the statement concerning temporal functional dependencies in Theorem 1 would also apply to these new dependencies.

Finally, it is possible to generalize all of the other dependency results, multivalued, fourth and fifth normal forms, etc. to the spatial and spatiotemporal regimes.

9 Recent Work

While the present paper was in review, there were two other extensions of existing dependency theory to temporal databases. Both built upon the work described here, and in fact referenced prior versions of this paper. We describe these briefly here.

Wijsen and his colleagues have adapted his dependency theory to a relational model without object identity [57, 58, 59]. In his most recent work [59], he adapts his approach to the valid-time subset of the BCDM. His snapshot functional dependency captures the same notion as our temporal functional dependency, while being expressed in a different notation. The notions of keys and normal forms defined in terms of snapshot functional dependencies also closely track the ones defined here.

Wang and his colleagues [53] extended temporal functional dependencies to accommodate temporal granularities. Specifically, their dependency, also termed a *temporal functional dependency*, must hold for an entire granule, say a day or month. As such, it is more properly considered a interstate dependency.

10 Summary and Future Research

In this paper, we have defined consistent temporal extensions of the concepts of functional dependencies, keys, and normal forms. We briefly surveyed conventional normalization concepts and extracted desiderata enumerating those properties of conventional concepts that we would like temporal normalization concepts to also possess. We further conducted the first thorough survey of previous contributions related to temporal relational database design. In part, this was done in attempt to build maximally on existing contributions and to put our proposal into a proper perspective.

Our definitions were shown to be more natural extensions than those previously proposed, in the sense that they satisfied all desiderata. The generality of our approach was indicated by applying it to spatial databases. The result is a consistent and wholesale application of existing dependency and normalization theory to valid-time, transaction-time, bitemporal, spatial, and spatiotemporal databases, in a variety of existing temporal relational data models, allowing temporal and spatial database design to closely track conventional database design.

We emphasize here the three fundamental decisions that made this possible. First, we used snapshot equivalence of temporal relations (defined as having identical snapshots over all valid and transaction times) as a formalization of the notion of temporal relations having the same information content. Second, we focused on the semantics of temporal relations rather than their representation. Our use of snapshot equivalence on conjunction with the fact that query languages of representational models generally provide the means of computing snapshots enabled this conceptual focus. The concepts apply globally, across most if not all existing representational temporal data models. As a result, new concepts are not needed for each representational data model. Third, we chose a simple data model that has the important feature that relation instances with the same information content are identical.

Our normal forms do not address all the issues that come into play when the schema for a temporal database is being designed. First, the normal forms do not consider the semantics of time-varying attributes, such as whether they are continuously varying or are stepwise constant. Secondly, the normal forms do not consider important efficiency concerns. Specifically, synchronous attributes, as defined by Navathe [32], may be seen to affect the space efficiency of the storage of a temporal relation or the time efficiency of evaluating a temporal query, yet are not relevant to the *semantics* of the temporal relation. Finally, more general inter-state (and inter-slice) constraints such as "No employee can later return to the same department," or "No employee can be assigned to departments in geographically separate plants" should be explored.

A fully articulated design methodology utilizing the normal forms presented here and taking into account the time semantics of tuples and attributes and efficiency concerns is still needed.

Acknowledgements

This work was conducted while the first author visited the University of Arizona. Funding was provided in part by NSF grants ISI-8902707 and ISI-9302244, by IBM contract #1124, and by an AT&T Foundation grant. In addition, the first author was supported in part by the Danish Natural Science Research Council through grants 11–1089–1 SE, 11–9675–1 SE, and 11–0061–1 SE. We very much appreciate insightful comments from Curtis E. Dyreson, Nick Kline, Jef Wijsen, and the reviewers on a previous draft.

References

[1] R. Ahmed. Personal Communication, March 1992.

[2] K. K. Al-Taha, R. T. Snodgrass, and M. D. Soo. Bibliography on Spatiotemporal Databases. *SIGMOD Record*, 22(1):59–67, March 1993.

[3] W. Armstrong. Dependency Structures of Data Base Relationships. In *Proceedings of the IFIP Congress*, 1974.

[4] J. Ben-Zvi. *The Time Relational Model*. PhD thesis, Computer Science Department, UCLA, 1982.

[5] M. Böhlen. Valid Time Integrity Constraints. Technical Report, University of Arizona, Department of Computer Science, TR 94-30, November, 1994. 22 pages.

[6] I. Bracken and C. Webster. Towards a Typology of Geographical Information Systems. *Int. Journal of Geographical Information Systems*, 3(2):137–152, 1989.

[7] M. Casanova, R. Fagin, and C. Papadimitriou. Inclusion Dependencies and Their Interaction with Functional Dependencies. *Journal of Computer and System Sciences*, 28(1):29–59, 1984.

[8] J. Clifford and A. Croker. The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans. In *Proceedings of the International Conference on Data Engineering*, pages 528–537, Los Angeles, CA, February 1987. IEEE Computer Society, IEEE Computer Society Press.

[9] J. Clifford and D. S. Warren. Formal Semantics for Time in Databases. *ACM Transactions on Database Systems*, 8(2):214–254, June 1983.

[10] E. F. Codd. *Further Normalization of the Data Base Relational Model*, Volume 6 of *Courant Computer Symposia Series*. Prentice Hall, Englewood Cliffs, N.J., 1972.

[11] E. F. Codd. *Relational Completeness of Data Base Sublanguages*, Volume 6 of *Courant Computer Symposia Series*, pages 65–98. Prentice Hall, Englewood Cliffs, N.J., 1972.

[12] E. F. Codd. Recent Investigations in Relational Database Systems. In *Proceedings of the IFIP Congress*, 1974.

[13] R. Fagin. Multivalued Dependencies and a New Normal Form for Relational Databases. *ACM Transactions on Database Systems*, 2(3):262–278, September 1977.

[14] R. Fagin. Normal Forms and Relational Database Operators. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1979.

[15] R. Fagin. A Normal Form for Relational Databases that is Based on Domains and Keys. *ACM Transactions on Database Systems*, 6(3):387–415, September 1981.

[16] S. K. Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, 13(4):418–448, December 1988.

[17] S. K. Gadia. A Seamless Generic Extension of SQL for Querying Temporal Data. Technical Report TR-92-02, Computer Science Department, Iowa State University, May 1992.

[18] C. S. Jensen, L. Mark, and N. Roussopoulos. Incremental Implementation Model for Relational Databases with Transaction Time. *IEEE Transactions on Knowledge and Data Engineering*, 3(4):461–473, December 1991.

[19] C. S. Jensen, L. Mark, N. Roussopoulos, and T. Sellis. Using Caching, Cache Indexing, and Differential Techniques to Efficiently Support Transaction Time. *VLDB Journal*, 2(1):75–111, 1992.

[20] C. S. Jensen, J. Clifford, R. Elmasri, S. K. Gadia, P. Hayes and S. Jajodia (eds). A Glossary of Temporal Database Concepts. *ACM SIGMOD Record*, 23(1):52–64, March 1994.

[21] C. S. Jensen, M. D. Soo, and R. T. Snodgrass. Unifying Temporal Data Models via a Conceptual Model. *Information Systems*, 19(7):513–547, December 1994.

[22] C. B. Jones. Data Structures for Three-Dimensional Spatial Information Systems in Geology. *International Journal of Geographical Information Systems*, 3(1):15–31, 1989.

[23] K. A. Kimball. The Data System. Master's thesis, University of Pennsylvania, 1978.

[24] H. F. Korth and A. Silberschatz. *Database System Concepts*. McGraw-Hill Advanced Computer Science Series. McGraw-Hill Book Company, 1986.

[25] N. A. Lorentzos and R. G. Johnson. Extending Relational Algebra to Manipulate Temporal Data. *Information Systems*, 13(3):289–296, 1988.

[26] N. A. Lorentzos and V. Kollias. The Handling of Depth and Time Intervals in Soil-Information Systems. *Computers and Geosciences*, 15(3):395–401, 1989.

[27] N. A. Lorentzos. Management of Intervals and Temporal Data in the Relational Model. Technical Report 49, Agricultural University of Athens, 1991.

[28] D. M. Mark, J. P. Lauzon, and J. A. Cebrian. A Review of Quadtree-Based Strategies for Interfacing Coverage Data with Digital Elevation Models in Grid Form. *International Journal of Geographical Information Systems*, 3(1):3–14, 1989.

[29] E. McKenzie. *An Algebraic Language for Query and Update of Temporal Databases*. PhD thesis, Computer Science Department, University of North Carolina, 1988.

[30] E. McKenzie and R. T. Snodgrass. Supporting Valid Time in an Historical Relational Algebra: Proofs and Extensions. Technical Report TR–91–15, Department of Computer Science, University of Arizona, August 1991.

[31] S. Nair and S. K. Gadia. Algebraic Optimization in a Relational Model for Temporal Databases. Technical Report TR-92-03, Computer Science Department, Iowa State University, May 1992.

[32] S. B. Navathe and R. Ahmed. A Temporal Relational Model and a Query Language. *Information Sciences*, 49:147–175, 1989.

[33] Z. M. Özsoyoğlu and L.-Y. Yuan A New Normal Form for Nested Relations *ACM Transactions on Database Systems*, 12(1):111–136, March 1987.

[34] J. Rissanen. Independent Components of Relations. *ACM Transactions on Database Systems*, 2(4):317–325, December 1977.

[35] R. Sadeghi. *A Database Query Language for Operations on Historical Data*. PhD thesis, Dundee College of Technology, Dundee, Scotland, December 1987.

[36] R. Sadeghi, W. B. Samson, and S. M. Deen. HQL—A Historical Query Language. Technical report, Dundee College of Technology, Dundee, Scotland, September 1987.

[37] F. Sadri. *Data Dependencies in the Relational Model of Data: A Generalization*. PhD thesis, Princeton University, October 1980.

[38] F. Sadri and J. Ullman. Template Dependencies: A Large Class of Dependencies in Relational Databases and Its Complete Axiomatization. *Journal of the Association for Computing Machinery*, 29(2), April 1982.

[39] N. Sarda. Algebra and Query Language for a Historical Data Model. *The Computer Journal*, 33(1):11–18, February 1990.

[40] N. Sarda. Extensions to SQL for Historical Databases. *IEEE Transactions on Knowledge and Data Engineering*, 2(2):220–230, June 1990.

[41] A. Segev and A. Shoshani. Logical Modeling of Temporal Data. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 454–466, May 1987.

[42] A. Segev and A. Shoshani. The Representation of a Temporal Data Model in the Relational Environment. In *Proceeding of the 4th International Conference on Statistical and Scientific Database Management*, 1988.

[43] R. T. Snodgrass and I. Ahn. Temporal Databases. *IEEE Computer*, 19(9):35–42, September 1986.

[44] R. T. Snodgrass. The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

[45] R. T. Snodgrass. Temporal Databases. In *Proceedings of the International Conference on GIS: From Space to Territory*, Volume 639 of *Lecture Notes in Computer Science*, pages 22–64. Springer-Verlag, September 1992.

[46] R. T. Snodgrass. An Overview of TQuel. Chapter 6 of *Temporal Databases: Theory, Design, and Implementation.* Benjamin/Cummings, 1993.

[47] R. T. Snodgrass (ed.), I. Ahn, G. Ariav, D. S. Batory, J. Clifford, C. E. Dyreson, R. Elmasri, F. Grandi, C. S. Jensen, W. Kafer, N. Kline, K. Kulkanri, T. Y. C. Leung, N. Lorentzos, J. F. Roddick, A. Segev, M. D. Soo, and S. M. Sripada. *The TSQL2 Temporal Query Language*. Kluwer Academic Press, 1995.

[48] M. D. Soo, C. S. Jensen, and R. T. Snodgrass. An Algebra for TSQL2. Chapter 27 of *The TSQL2 Temporal Query Language.* Kluwer Academic Press, 1995, pp. 505–546.

[49] A. U. Tansel. Adding Time Dimension to Relational Model and Extending Relational Algebra. *Information Systems*, 11(4):343–355, 1986.

[50] A. U. Tansel and L. Garnett. Nested Historical Relations. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 284–293, 1989.

[51] J. D. Ullman. *Principles of Database and Knowledge-Base Systems,* Volume 1. Computer Science Press, Potomac, Maryland, 1988.

[52] V. Vianu. Dynamic functional dependencies and database aging. *JACM*, 34(1):28–59, January 1987.

[53] X. Wang, C. Bettini, A. Brodsky and S. Jajodia. Logical Design for Temporal Databases with Multiple Granularities. Technical Report ISSE-TR 94-111, ISSE Department, George Mason University, 1994.

[54] J. Wijsen. A Theory of Keys for Temporal Databases. In *Actes 9èmes Journèes Bases de Données*, pages 35–54, Toulouse, France, 1993.

[55] J. Wijsen, J. Vandenbulcke, and H. Olivé. Functional Dependencies Generalized for Temporal Databases that Include Object-Identity. In *Proceedings of the International Conference on the Entity-Relationship Approach*, pages 100-114, Arlington, Texas, 1993.

[56] J. Wijsen, J. Vandenbulcke, and H. Olivé. Database Design with Temporal Dependencies. Technical Report, Departement Toegepast Economische Wetenschappen, 1993.

[57] J. Wijsen, J. Vandenbulcke, and H. Olivé. Temporal Dependencies in Relational Database Design. In *Actes 10èmes Journées Bases de Données*, pages 157–169, Clermont-Ferrand, France, 1994.

[58] J. Wijsen, J. Vandenbulcke, and H. Olivié. On time-invariance and synchronism in valid-time relational databases. *Journal of Computing and Information*, 1(1), 1994.

[59] J. Wijsen. Design of Temporal Relational Databases Based on Dynamic and Temporal Functional Dependencies. In *Recent Advances in Temporal Databases*, Springer-Verlag, September, 1995, pp. 61–76.

[60] C. Zaniolo. *Analysis and Design of Relational Schemata for Database Systems*. PhD thesis, UCLA, July 1976.