Extending Normal Forms to Temporal Relations

Christian S. Jensen[1]
Richard T. Snodgrass[2]
Michael D. Soo[2]

TR 92-17

July 29, 1992

Abstract

Normal forms play a central role in the design of relational databases. Recently several normal forms for temporal relational databases have been proposed. The result is a number of isolated and sometimes contradictory contributions that only apply within specialized settings.

This paper attempts to rectify this situation. We define a consistent framework of temporal equivalents of all the important conventional database design concepts: functional and multivalued dependencies, primary keys, and third, Boyce-Codd, and fourth 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.

We compare our approach with previously proposed definitions of temporal normal forms and temporal keys. To demonstrate the generality of our approach, we outline how normal forms and dependency theory can also be applied to spatial databases, as well as to spatial-temporal databases.

[1]Department of Mathematics and Computer Science
Aalborg University
Fredrik Bajers Vej 7E
DK-9220 Aalborg Ø, DENMARK
csj@iesd.auc.dk

[2]Department of Computer Science
University of Arizona
Tucson, AZ 85721
{rts,soo}@cs.arizona.edu

Extending Normal Forms to Temporal Relations

Contents

1 Introduction

The 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.

In this paper we describe how normal forms may be defined on *temporal* relations which record time-varying information. A confusing array of normal forms for temporal relations has been previously proposed, including *first temporal normal form* [Segev & Shoshani 88], *time normal form* [Navathe & Ahmed 89], and P and Q *normal forms* [Lorentzos & Kollias 89]. None of these definitions is truly an extension of conventional normal forms, for a variety of reasons that we detail in Section 3. Also, each definition is restricted to a specific data model, and inherits the peculiarities inherent in that model. It is not satisfactory to have to define all the normal forms anew for each of the two dozen existing temporal data models [Snodgrass 92].

We adopt a different tack. In defining our normal forms, we utilize a new data model, termed the *bitemporal conceptual data model*, that is in a sense the "largest common denominator" for existing models [Jensen et al. 92A]. Specifically, we have shown how to map relations and operations in several quite different temporal 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. Consequently, the temporal normal forms are defined in the context of this model.

In the next section, we survey conventional normal forms, highlighting the properties that should carry over to temporal normal forms. We then evaluate previously proposed definitions of temporal normal forms and temporal keys against these properties, demonstrating that, while all of these definitions satisfy some of the desiderata, no definition is an entirely natural extension of conventional normal forms.

We introduce in Section 4 a new data model, the *bitemporal conceptual data model*, in which to express normal forms. We then apply the conventional normal forms to this new model in such a way that virtually all of the theory behind the normal forms applies to the temporal analogues that we define. 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.

To demonstrate the generality of our approach, we outline how normal forms and dependency theory can also be applied to spatial databases, as well as to spatial-temporal databases.

2 Conventional Normal Forms

Normal forms are guidelines for what constitutes a good relation schema when designing the conceptual schema of a database. While temporal normal forms should be faithful extensions of these notions, some previously proposed temporal normal forms do not maintain such a correspondence. Hence, as a prelude to presenting our temporal normal forms, we survey existing standard relational normal forms, describing the kinds of problems addressed and emphasizing the aspects common to normal forms. Throughout we highlight the important concepts of dependency and normalization theory that we wish to preserve when defining temporal normal forms.

A normal form is an *intensional* property of a database schema that follows from a set of (functional or 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 three most important normal forms, third normal form [Codd 72], Boyce-Codd normal form [Codd 74] and fourth normal form [Fagin 77], as well as the concept of key, all of which rely on the concept of functional dependency [Codd 72B] or multivalued dependency [Zaniolo 76].

DEFINITION: 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

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

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 constrains the set of possible extensions of a relation. Which functional dependencies are applicable to a schema reflects the reality being modeled and the applications of the schema. Determining the relevant functional dependencies is a primary task of the database designer.

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

EXAMPLE: 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, but may have several phone numbers. Corresponding to these real-world constraints are the following functional dependencies.

$$\begin{aligned} \text{Name} &\rightarrow \text{Dept} \\ \{\text{Name, PhNo}\} &\rightarrow \{\text{Name, PhNo, Dept}\} \end{aligned}$$

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. □

DEFINITION: The *transitive closure* of a set of functional dependencies, F, with respect to some set of inference rules (equivalent to Armstrong's inference rules), is given by F^+. □

EXAMPLE: In the example database, the transitive closure of those two functional dependencies contains the following additional non-trivial dependences.

$$\begin{aligned} \{\text{Name, PhNo}\} &\rightarrow \text{Dept} \\ \{\text{Name, PhNo}\} &\rightarrow \{\text{Name, Dept}\} \\ \{\text{Name, PhNo}\} &\rightarrow \{\text{PhNo, Dept}\} \end{aligned}$$

□

DEFINITION: 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. □

DEFINITION: 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. □

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 such anomalies are avoided.

EXAMPLE: As 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 don't 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, for which the normals forms are satisfied. Such 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: Let X and Y be arbitrary sets of non time-stamp 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,

$$r = \pi_X(r) \bowtie \pi_Y(r).$$ □

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: 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 [Ullman 88, Korth & Silberschatz 86]. □

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: 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 [Ullman 88]. □

EXAMPLE: 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. This decomposition is both loss-less and dependency-preserving. □

Some decomposition algorithms can be proven to be dependency-preserving; others jettison this property in favor of more desirable ones, such as lossless-join.

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, this is generally not possible for BCNF. If a dependency preserving BCNF decomposition is not possible, 3NF is usually preferred, at the risk of added data redundancy.

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.

We define an additional higher level normal form, fourth normal form [Fagin 77], which relies on the concept of multivalued dependencies [Zaniolo 76].

DEFINITION: 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 *multivalue dependent* on X, denoted $X \twoheadrightarrow Y$, if for any two tuples s_1 and s_2 in any possible extension of R, where $s_1[X] = s_2[X]$, there exists tuples s_3, s_4 in that extension such that

$$\begin{aligned} s_1[X] &= s_2[X] = s_3[X] = s_4[X] \\ s_3[Y] &= s_1[Y] \\ s_3[R - (X \cup Y)] &= s_2[R - (X \cup Y)] \\ s_4[Y] &= s_2[Y] \\ s_4[R - (X \cup Y)] &= s_1[R - (X \cup Y)]. \end{aligned}$$ □

Intuitively, $X \twoheadrightarrow Y$ states that attribute values of X may determine multiple values of the attributes in Y, and that the relationship between X and Y is independent of the values of the remaining attributes in R.

EXAMPLE: Consider again the relation schema Emp = (Name, Dept, PhNo), and assume that in the reality being modeled, many employees can work for the same department and a single department may have many telephones, each of which is accessible to all members of the department.

Furthermore, assume that the functional dependency of the previous example, an employee may only work for one department, no longer applies. With these restrictions, the non-trivial multivalued dependencies Dept$\twoheadrightarrow$ Name, Dept$\twoheadrightarrow$ PhNo and {Dept, Name} $\twoheadrightarrow$ PhNo hold in the schema. □

Fourth normal form is a strict generalization of BCNF, i.e., any schema in fourth normal form must also be in BCNF, and there exist schemas obeying BCNF that are not in fourth normal form. By incorporating multivalued dependencies as well as functional dependencies in the schema decomposition process, fourth normal form reduces additional data redundancy that may be present in schemas developed using normal forms that rely solely on functional dependencies.

DEFINITION: *(4NF)* A relation schema R is in *fourth normal form* (4NF) with respect to a set F of functional and multivalued dependencies if for all non-trivial multivalued dependencies $X \twoheadrightarrow Y$ in F^+, X is a superkey of R. □

EXAMPLE: Since there are no non-trivial functional dependencies for the Emp schema, the schema is in BCNF. However, as Dept is not a superkey, 4NF is violated. □

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.

We should mention two other normal forms [Codd 72B]. *Second normal form* is weaker than third normal form, and is only of historical interest. *First normal form* (1NF) is unlike any of the other normal forms, in that it is not defined in terms of functional dependencies. Instead, it merely requires that attribute values be drawn from domains of atomic values, which do not have internal structure.

3 Review of Previous Proposals

In this section, we consider previous proposals of temporal normalization concepts. We examine in turn normal forms, functional dependencies and keys. We first explain each contribution, then evaluate whether it naturally extends conventional normal forms.

3.1 Functional Dependencies and Normal Forms

As a basis for the discussion, we summarize, from Section 2, the fundamental qualities of the (conventional) normal forms.

1. Functional dependencies and normal forms are intensional, not extensional, properties.
2. Normal forms are defined solely in terms of the dependencies that are satisfied.
3. Normal forms are properties of stored (base) relations only.
4. Functional dependencies and normal forms are defined independently of the representation of a relation.

We evaluate each of the previously proposed normal forms based on these desiderata.

3.1.1 First Temporal Normal Form

Segev and Shoshani define in their Temporal Data Model a normal form, 1TNF, for valid time relations [Segev & Shoshani 88]. 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 of 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: 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." [[Segev & Shoshani 88], 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 time stamping, respectively. For the interval based representation, the following definition is given.

DEFINITION: "A relation with a schema, $R(S, A_1, \ldots, A_n, T_s, T_e)$, is in 1TNF if there do not exist two tuples $r^1(s^1, a_1^1, \ldots, a_n^1, t_s^1, t_e^1)$ and $r^2(s^2, a_1^2, \ldots, a_n^2, t_s^2, t_e^2)$ such that $s^1 = s^2$ and the intervals $[t_s^1, t_e^1]$ and $[t_s^2, t_e^2]$ intersect." [[Segev & Shoshani 88], p. 17] □

According to the authors, 1TNF is required rather than desirable.

This normal form deviates from conventional normal forms in several respects. 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 4, as a conceptual definition is provided. However, 1TNF does not satisfy desiderata 1 (since normal forms are defined on relations, rather than relation schemas), 2 (though the conceptual definition does employ the notion of snapshot 1NF) or 3 (since operators are expected to preserve 1TNF, and are only defined over 1TNF relations).

3.1.2 Time Normal Form

This normal form for valid time relations also applies to an interval tuple timestamp representation, containing the attributes T_S (the starting valid time) and T_E (the ending valid time) [Navathe & Ahmed 89]. Unlike 1TNF, it is based on the notion of temporal dependency, defined as follows.

DEFINITION: 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 $t[A_j] = t'[A_j]$.

[[Navathe & Ahmed 89], p. 156 and [Ahmed 92]] □

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 A_k (k is i or j) but disagree on A_{i+j-k}.

DEFINITION: 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."
[[Navathe & Ahmed 89], p. 157] □

This definition satisfies desideratum 1. The definition of TNF does not satisfy desiderata 2 (rather, it requires a normal form to be violated), 3 (because operators are defined only on TNF relations) or 4.

Finally, snapshot normal forms, e.g., BCNF, and therefore snapshot functional dependencies are applied to the representations of valid time relations [Navathe & Ahmed 89], violating desideratum 4.

3.1.3 The HSQL Data Model

In the valid time data model associated with the query language HSQL [Sarda 90B], 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 $R = (A_1, \ldots, A_n, \text{PERIOD})$ [Sarda 90A]. It is claimed, but not demonstrated, that conventional normalization techniques apply to the design of a valid time database. Part of the purpose of this paper is to give a formal characterization of the sense in which this is true.

3.1.4 P and Q Normal Forms

The interval extended relational model (IXRM) [Lorentzos 91] integrates (n-dimensional) intervals into the snapshot relational model. The intervals, one per attribute, may be drawn from any data type, including time and space. Interval-valued attribute values are accommodated, and new operators that manipulate relations with interval attributes are defined.

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. Rather, such timestamps are more properly thought of as *user-defined time* [Snodgrass & Ahn 86]. IXRM is mentioned here because, while not a valid-time model, relations in this model may *represent* valid-time relations and some of the operators may be conveniently used for valid-time queries.

Lorentzos extended the notion of functional dependency to cover interval values as well as atomic values, resulting in an extended notion of key as well. Here the timestamp merely appears as one of the attributes in the functional dependency or key. The extensions are particular to the special interval attributes, and they have no counterparts in conventional valid-time relational models.

Lorentzos also defined P normal form. We give a simplied definition; the original definition ([Lorentzos 91], p. 49) used a rather complex algebraic operator.

DEFINITION: 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 1 and 3. However, as it does not apply at the conceptual level, but at the representation level, it does not satisfy desiderata 2 or 4.

A second normal form is also defined. It relies on the following concept.

DEFINITION: "Two or more attributes of a schema are *independent* if none of them is fully or partially dependent on any of the others." [[Lorentzos 91], p. 51] □

Here, "dependent" means point or interval dependent (which are generalizations of functionally dependent). We give a simplified version here also of this normal form ([Lorentzos 91], p. 51).

DEFINITION: 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 all time invariant attributes (i.e., all attributes excluding the valid time interval attribute and the time-invariant key) are not independent. □

This normal form also satisfies desiderata 1 and 3. It does not satisfy desiderata 2 (because it relies on PNF) or 4.

3.2 Keys

We now turn to the related topic of defining keys for a temporal data model. We first list the properties held by the definition of snapshot keys.

1. Keys are intensional.
2. Relation keys are defined soley in terms of functional dependencies.
3. Keys are properties of stored (base) relations only.
4. Particular attributes are not a priori designated as keys.
5. Keys are independent of the representation.
6. Primary keys are minimal.

Next we examine various proposals for temporal keys.

3.2.1 The HQL Data Model

In the data model associated with the query language HQL [Sadeghi et al. 87], 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, apart from being unclear, is representation-dependent. In summary, this definition appears to satisfies desiderata 1–4, but does not satisfy desiderata 5 or 6.

3.2.2 The HSQL Data Model

In the data model associated with the query language HSQL [Sarda 90B], a key is defined as follows.

DEFINITION: A set of attributes X is a key of a valid-time relation schema $\overline{R} = (A_1, \ldots, A_n)$ if "it has unique values across all tuples at any point in time." [[Sarda 90A], p. 13] □

This definition satisfies desiderata 3, 4 and 5; it does not satisfy desiderata 1 or 2; primary keys are not defined.

3.2.3 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 [Lorentzos 91]. 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. A key is required to be minimal. As Lorentzos mentions "the" key, one could assume that only the primary key was being defined. This definition of key satisfies all but desideratum 5.

3.2.4 The TempSQL Data Model

Gadia and Nair define a special notion of key in the data model associated with the query language TempSQL [Gadia 92, Nair & Gadia 92]. To understand this concept, the type of relation employed must be understood first.

EXAMPLE: Consider the following relation 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). 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: "A *relation* r 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." [[Gadia 92] 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.

This concept of key presents a major departure from conventional normalization theory and has no counterpart there. First, it appears that a key is a property of a relational instance, making the definition extensional. Second, 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 4 and 5; it does not satisfy desiderata 1, 2, or 3 (because operators can change the key of a relation); and primary keys are not defined in the model.

3.3 Summary

None of the five temporal normal forms proposed previously are a natural extension of conventional normal forms; none satisfied all the desiderata. For the few temporal data models in which keys were discussed (in the majority of the two dozen temporal data models, keys were not discussed), none include an extension of the concept of snapshot key satisfying all six desiderata. Finally, all of these definitions were *model-specific*—it is generally not possible to apply in a straightforward fashion particular definitions to other data models.

4 A Bitemporal Conceptual Data Model

We feel that the reason why so may 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 most 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, in many situations, to presentation and storage, those representations can be employed for those purposes, while retaining the semantics of the conceptual data model.

We first informally 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 [Jensen & Snodgrass 92]. Finally, we outline a few of the *representational* data models in which instances and operators can be mapped to and from the BCDM.

4.1 Objects in the Model

The primary reason behind the success of the relational model is its simplicity. A bitemporal relation is necessarily more complex [Jensen et al. 92]. Not only must it associate values with facts, as does the relational model, it must also specify *when* the facts were valid in reality, as well as *when* the facts were current in the database. Since our emphasis is on semantic clarity, we will extend the conventional relational model as small an extent as necessary to capture this additional information.

Tuples in a bitemporal 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 part of reality being modeled, and transaction time models the update activity of the relation [Snodgrass & Ahn 86]. For both domains, we assume that the database system has limited precision, and we term the smallest time unit a *chronon* [Dyreson & Snodgrass 92]. As we can number the chronons, the domains are isomorphic to the domain of natural numbers.

In general, the schema of a bitemporal relation, R, consists of an arbitrary number of explicit attributes, $A_1, \ldots, A_n$, encoding some fact (possibly composite) and an implicit timestamp attribute, T. Thus, a tuple $s = (a_1, a_2, \ldots, a_n|t)$, in a bitemporal relation instance $r(R)$, consists of a number of attribute values associated with a timestamp value. Note that, as in the conventional relational model, the attributes $A_1, \ldots, A_n$ must each be atomic-valued, meeting the requirement that relations be in 1NF. Finally, 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.

Associated with a tuple is a set of *bitemporal chronons* (rectangles) in the two-dimensional space spanned by valid time and transaction time. Such a set is termed a *bitemporal element*[1]. 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 an arbitrary 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.

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.

EXAMPLE: Consider the normalized relation schema EmpDept = (Name, Dept) used in the example of Section 3.2.4. This relation records information such as "Bill works for the shipping department." The BCDM schema would be (Name, Dept | T). We assume that the granularity of chronons is one day for both valid time and transaction time, and the period of interest is the month of June 1992.

Figure 1 shows how the bitemporal element in an employee's department tuple changes. The x-axis denotes transaction time, and the y-axis denotes valid time. Employee Bill was hired by the company as temporary help in the shipping department for the interval from June 10th to June 15th, and this fact is recorded in the database proactively on June 5th. This is shown in Figure 1(a). The arrows pointing to the right signify that the tuple has not been logically deleted; it continues through to the transaction time NOW. On June 10th, the personnel department discovers an error. Bill had really been hired for the valid time interval from June 5th to June 20th. The database is corrected on June 10th, and the updated bitemporal element is shown in Figure 1(b). On June 15th, the personnel department is informed that the correction was itself incorrect; Bill really was hired for the original time interval, June 10th to June 15th, and the database is corrected the same day. This is shown in Figure 1(c). Lastly, Figure 1(d) shows the result of three updates to the relation, all of which take place on June 20th. While the the period of validity was correct, it was discovered that Bill was not in the shipping department, but in the loading department. Consequently, the fact (Bill, Ship) is removed from the current state and the fact (Bill, Load) is inserted. A new employee, Al, is hired for the shipping department for the interval from June 25th to June 30th.

We note that the number of bitemporal chronons in a given bitemporal element is the area enclosed by the bitemporal element. The bitemporal element for (Bill, Ship) contains 140 bitemporal chronons.

[1]This term is a generalization of the term *temporal element*, used previously to denote a set of single dimensional chronons [Gadia 88].

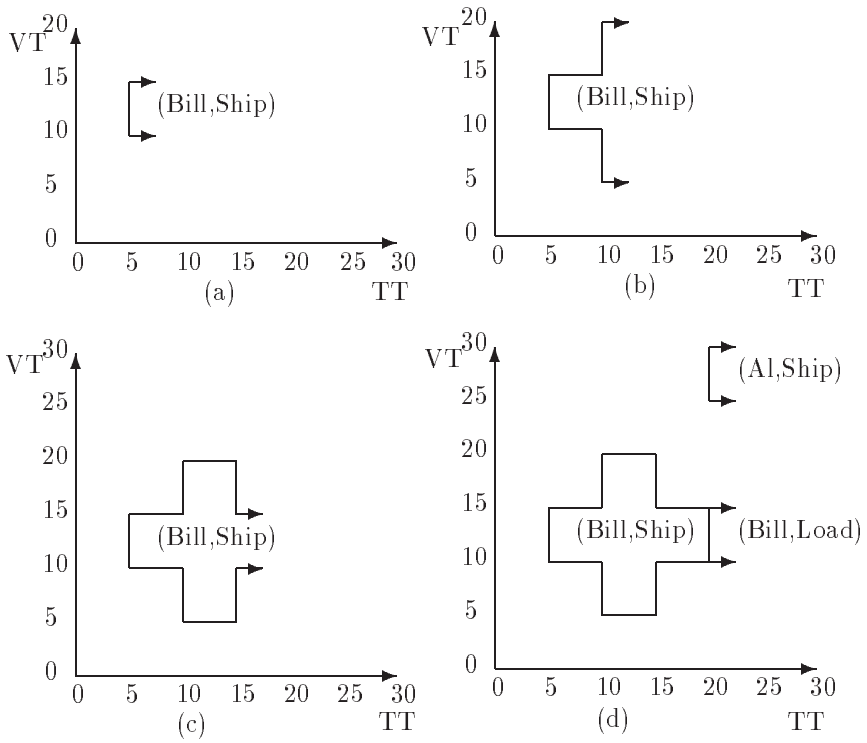

Figure 1: Bitemporal Elements

The example illustrates how transaction time and valid time are handled. As time passes, i.e., as the computer's internal clock advances, the bitemporal elements associated with current facts are updated. For example, when (Bill, Ship) was first inserted, the six valid time chronons from 10 to 15 had associated the transaction time chronon NOW. At time 5, the six new bitemporal chronons, $(5,10),\ldots,(5,15)$, were appended. This continued until time 9, after which the valid time was updated. Thus, starting at time 10, 16 bitemporal chronons are added at every clock tick.

The actual bitemporal relation corresponding to the graphical representation in Figure 1(d) is shown below. This relation contains three facts. The implicit timestamp attribute T shows each transaction time chronon associated with each valid time chronon as a set of ordered pairs.

Emp	Dept	T
Bill	Ship	$\{(5,10),\ldots,(5,15),\ldots,(9,10),\ldots,(9,15),$ $(10,5),\ldots,(10,20),\ldots,(14,5),\ldots,(14,20),$ $(15,10),\ldots,(15,15)\ldots,(19,10),\ldots,(19,15)\}$
Bill	Load	$\{(NOW,10),\ldots,(NOW,15)\}$
Al	Ship	$\{(NOW,25),\ldots,(NOW,30)\}$

The relation is created by the following sequence of commands, described in detail elsewhere [Jensen et al. 92A], invoked at the indicated transaction times.

Command	*Transaction Time*
`insert(dept,("Bill","Ship"),[6/10,6/15])`	6/5
`modify(dept,("Bill","Ship"),[6/5,6/20])`	6/10
`modify(dept,("Bill","Ship"),[6/10,6/15])`	6/15
`delete(dept,("Bill","Ship"))`	6/20
`insert(dept,("Bill","Load"),[6/10,6/15])`	6/20
`insert(dept,("Al","Ship"),[6/25,6/30])`	6/20

□

Depending on the extent of decomposition, a tuple in a bitemporal relation represents an atomic or a composite fact. Thus, we will use the terminology that a tuple stores 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 and transaction time, respectively. Thus a valid-time tuple has an associated set of valid time chronons (termed a *valid-time element*), and a transaction-time tuple has an associated set of transaction time chronons (termed a *transaction-time element*). For clarity, we use the term *snapshot relation* for a conventional relation. Snapshot relations support neither valid time nor transaction time.

Time-invariant relationships can be accommodated in this model. Note that such relationships are almost always valid-time invariant; they generally vary over transaction time.

EXAMPLE: The relation ParentOf(Parent, Child) can be modeled with a valid-time element covering all of time, from beginning to forever. If it was later determined that someone else was the parent of a particular person, this would cause the present tuple associated with that person to be terminated at that transaction time. □

The bitemporal conceptual data model timestamps tuples, as does the Time Relational Model [Ben-Zvi 82], the Historical Data Model [Clifford & Warren 83], the Temporal Relational Model [Navathe & Ahmed 89], the Temporal Data Model [Segev & Shoshani 87] and the data models associated with Legol 2.0 [Jones et al. 79], HQL [Sadeghi et al. 87], HSQL [Sarda 90B] and TQuel [Snodgrass 87]. The timestamps are temporal elements, as in the Historical Relational Data Model [Clifford & Croker 87], the Homogeneous Relational Model [Gadia 88] and the data model associated with TempSQL [Gadia 92]. Attributes are atomic, as in most of the temporal data models proposed to date [Snodgrass 92].

4.2 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.

We first define bitemporal analogues of some of the snapshot relational operators, to be denoted with the superscript "$_{\mathrm{B}}$".

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.

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

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, $\sigma_P^{\mathrm{B}}(r)$, is defined as follows.

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

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.

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

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 denote an arbitrary time value and let t' denote a time not exceeding NOW. The *valid timeslice* operator (τ^{B}) yields a relation timestamped with transaction-time elements; the *transaction timeslice* operator (ρ^{B}) evaluates to a relation timestamped with valid-time elements[2].

$$\begin{aligned} \tau_t^{\mathrm{B}}(r) &= \{z^{(n+1)} \mid \exists x \in r\ (z[A] = x[A] \wedge t' \in \mathit{elem\_1}(x[\mathrm{T}]) \wedge z[\mathrm{T}_v] = \mathit{elem\_2}(x[\mathrm{T}]))\} \\ \rho_{t'}^{\mathrm{B}}(r) &= \{z^{(n+1)} \mid \exists x \in r\ (z[A] = x[A] \wedge t \in \mathit{elem\_2}(x[\mathrm{T}]) \wedge z[\mathrm{T}_t] = \mathit{elem\_1}(x[\mathrm{T}]))\} \end{aligned}$$

Here, *elem_1* selects all the transaction-time chronons from a bitemporal element, and *elem_2* selects all the valid-time chronons.

There also exist variants that extract a snapshot relation from a valid-time relation (τ^{V}) and that extract a snapshot relation from a transaction-time relation (ρ^{T}). To extract from r the tuples valid at time t and current in the database during t' (termed a *snapshot* of r), either $\tau_t^{\mathrm{V}}(\rho_{t'}^{\mathrm{B}}(r))$ or $\rho_{t'}^{\mathrm{T}}(\tau_t^{\mathrm{B}}(r))$ may be used; these two expressions evaluate to the same snapshot relation.

Note that since relations in the data model are *homogeneous*, i.e., all attributes in a tuple are associated with the same timestamp [Gadia 88], the valid or transaction timeslice of a relation will not introduce any nulls into the resulting relation.

4.3 Associated Representational Data Models

The objectives of the bitemporal conceptual data model are narrow in scope. This data model is designed capture the time-varying semantics of the data, without concern for ease of implementation, convenience of presentation, or amenability to query optimization. For these aspects, we rely on existing *representational* data model(s).

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

We have demonstrated equivalence mappings between the conceptual model and several representational models [Jensen et al. 92A]. This equivalence is based on *snapshot equivalence*, defined in Section 5.4. Snapshot equivalence formalizes the notion that two temporal relations have the same information contents and provides a natural means of comparing rather disparate representations. Specifically, we have shown that conceptual instances could be mapped into instances of three data models: a 1NF tuple timestamped data model [Snodgrass 87], a data model based on 1NF timestamped change requests recorded in backlog relations [Jensen et al. 91], and a non-1NF data model in which attribute values were stamped with rectangles in transaction-time/valid-time space [Gadia 92]. We also showed how the relational algebraic operators defined in the previous section could be mapped to analogous operators in the representational models.

5 Generalizing Dependency and Normal Form Theory

The bitemporal conceptual data model just defined provides the means to generalize standard relational dependency and normal form theory to accommodate temporal relations. In this section we do just that, generalizing in turn the concepts of functional dependencies, multivalued dependencies, keys, the normal forms themselves, and lossless join decomposition.

5.1 Temporal Dependencies

Functional dependencies are *intensional*, i.e., they apply to every possible extension. This intuitive notion already encompasses time, for it 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 $R = (A_1, \ldots, A_n|\mathrm{T})$, and a parallel snapshot relation r' with the same schema (but without the implicit timestamp attribute): $R' = (A_1, \ldots, A_n)$. The current state of r, denoted by $\rho^{\mathrm{T}}_{NOW}(r)$, will faithfully track the current state of r'. The information in a past state of r' will be retained in r, and can be extracted via $\rho^{\mathrm{T}}_{t'}(r)$. 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. The same argument can be applied to valid-time relations and to bitemporal relations, yielding the following characterization.

DEFINITION: Let X and Y be sets of non time-stamp attributes of a temporal relation schema, R. A *temporal functional dependency*, denoted $X \overset{\mathrm{T}}{\rightarrow} Y$, exists on R if

$$\forall r(R)\ \forall t, t'\ \forall s_1, s_2 \in \tau_t(\rho_{t'}(r))\ (s_1[X] = s_2[X] \Rightarrow s_1[Y] = s_2[Y]).$$ □

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 operator variants with the superscripts "$_{\mathrm{V}}$" (for valid-time), "$_{\mathrm{T}}$" (for transaction-time) and "$_{\mathrm{B}}$" (for bitemporal), the temporal functional dependency is generic, applying to all forms of temporal relations, with the appropriate operator variants coming into play. The "$_{\mathrm{T}}$" designation in a temporal functional dependency refers to the generic adjective "temporal", not the specific adjective "transaction-time."

EXAMPLE: Consider again the database recording 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 constraints as temporal dependencies, we have Name$\stackrel{\mathrm{T}}{\rightarrow}$Dept and {Name, PhNo}$\stackrel{\mathrm{T}}{\rightarrow}${Name, PhNo, Dept}. □

Functional dependencies on snapshot schemas generalize to analogous temporal functional dependencies, as illustrated in the previous example. Specifically, the following holds.

$$X \rightarrow Y \Leftrightarrow X \stackrel{\mathrm{T}}{\rightarrow} Y.$$

Note that both $X \rightarrow Y$ and $X \stackrel{\mathrm{T}}{\rightarrow} Y$ impose constraints derived from the reality being modeled on possible instances in the data model (intuitively, the same real world constraints generate both $X \rightarrow Y$ and $X \stackrel{\mathrm{T}}{\rightarrow} Y$). It is also important to note that two separate data models are involved here. The dependency $X \rightarrow Y$ applies to the *snapshot* data model only, whereas $X \stackrel{\mathrm{T}}{\rightarrow} Y$ applies to temporal data models: valid-time, transaction-time, and bitemporal data models.

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 [Sarda 90A]. Assume that $(A_1, \ldots, A_n|\mathrm{T})$ is the schema for a temporal relation R. An instance of R 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 $(A_1, \ldots, A_n, T)$, where T is simply another explicit attribute. We can compare functional dependencies in the two interpretations.

THEOREM: With X and Y denoting arbitrary non-timestamp attributes of a relation schema,

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

PROOF: The following instance satisfies Emp $\cup \{\mathrm{T}\} \rightarrow$ Dept but not Emp$\stackrel{\mathrm{T}}{\rightarrow}$Dept.

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

Note also 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 that the converse *does* hold.

THEOREM: Letting X and Y be sets of non-timestamp attributes of a relation schema,

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

PROOF: Assume that $X \stackrel{\mathrm{T}}{\rightarrow} Y$ holds in R and let r be an arbitrary instance of R. 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 such that $s_1[X \cup \{\mathrm{T}\}] = s_2[X \cup \{\mathrm{T}\}]$ but $s_1[Y] \neq s_2[Y]$. Let (t, t') 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, $s_1', s_2' \in \tau_t(\rho_{t'}(r))$. However, $s_1'[X] = s_2'[X]$ and by assumption $s_1[Y] \neq s_2[Y]$, and hence $X \stackrel{\mathrm{T}}{\rightarrow} Y$ is not satisfied, and the implication of the theorem is true by contradiction. □

Multivalued temporal dependencies may be defined using the same template as that used for defining temporal functional dependencies. Specifically, for each universal qualification of possible instances of R, $\forall r(R) \forall s \in r$, in the original definition, substitute universal quantification over all possible snapshots of instances of R, $\forall r(R) \forall t, t' \; \forall s \in \tau_t(\rho_{t'}(r))$.

5.2 Temporal Keys

Since temporal functional dependencies are such a natural extension of conventional functional dependencies, extension of the concepts of temporal key, temporal transitive closure, and temporal normal forms are straightforward.

DEFINITION: A set of attributes X of a temporal relation schema is a *temporal superkey* of R if $X \xrightarrow{T} R$. The *primary temporal key* is a minimal temporal superkey. □

EXAMPLE: As before, 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 $X \cup \{T\}$ is the (snapshot) primary key of the representation of the temporal relation, it may not be the case that X is a temporal key.

5.3 Temporal Normal Forms

We can now generalize snapshot normal forms in a manner similar to generalizing keys. The comments made in connection with dependencies in Section 5.1 about the inadequacy of using snapshot definitions incorporating the timestamp attribute apply here as well.

DEFINITION: A pair (R, F) of a temporal relation schema R and a set of associated temporal functional dependencies F is in *Boyce-Codd temporal normal form* (BCTNF) if

$$\forall\, X \xrightarrow{T} Y \;\in F^+ \; (Y \subseteq X \vee X \xrightarrow{T} R).$$ □

DEFINITION: A pair (R, F) of a temporal relation schema R and a set of associated temporal functional dependencies F is in *third temporal normal form* (3TNF) if for all non-trivial temporal functional dependencies $X \xrightarrow{T} Y$ in F^+, X is a temporal superkey for R or each attribute of Y is part of a minimal temporal key of R. □

EXAMPLE: The relational schema Emp = (Name, Dept, PhNo |T) violates both 3TNF and BCTNF. □

Temporal versions of other conventional normal forms based on functional and multivalued dependencies may be expressed analogously, e.g., 2NF, 4NF.

5.4 Properties of Temporal Normal Forms

During database design, relation schemas are brought to satisfy normal forms by means of decomposition. Such decompositions must be reversible. We utilize the temporal natural join operator to identify such loss-less join decompositions.

DEFINITION: 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 $\bowtie^B$ if for all $r(R)$ that satisfy the set of temporal functional dependencies on R,

$$r = \pi^B_X(r) \bowtie^B \pi^B_Y(r).$$ □

In order to prove the theorems stated here, a few auxiliary definitions are practical.

DEFINITION: Two bitemporal relation instances, r and s, are *snapshot equivalent*, $r \overset{S}{\equiv} s$, if for all times t and for all times t' not exceeding NOW,

$$\tau_t^V(\rho_{t'}^B(r)) = \tau_t^V(\rho_{t'}^B(s)).$$ □

Observe that bitemporal relations in the BCDM have the property that

$$r_1 = r_2 \Leftrightarrow r_1 \overset{S}{\equiv} r_2 \ .$$

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

DEFINITION: A temporal relation instance, r, is a *snapshot subset* of a temporal relation instance, s, $r \overset{S}{\subseteq} s$, if for all times t_1 not exceeding $NOW$and all times t_2,

$$\tau_{t_2}^V(\rho_{t_1}^B(r)) \subseteq \tau_{t_2}^V(\rho_{t_1}^B(s)).$$ □

Here, the similar property does not apply. Specifically, $r_1 \overset{S}{\subseteq} r_2$ does not imply $r_1 \subseteq r_2$.

LEMMA: For two bitemporal relation instances r and s over the same schema,

$$r \overset{S}{\subseteq} s \text{ and } s \overset{S}{\subseteq} r \Leftrightarrow r \overset{S}{\equiv} s.$$ □

We now show that algorithms for normal form decomposition in conventional relational databases are applicable to temporal databases as well.

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

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

PROOF: Assume that $X \cap Y \overset{T}{\rightarrow} X$ holds on R and let r be an arbitrary instance of R. Let A, B, and C partition the attributes of R so that $A = X \cap Y$, $B = X - Y$, and $C = Y - X$. Showing that the definition of lossless holds is equivalent to showing that $r \overset{S}{\equiv} \pi_X^B(r) \bowtie^B \pi_Y^B(r)$, which in turn is equivalent to showing each of

$$r \overset{S}{\subseteq} \pi_X^B(r) \bowtie^B \pi_Y^B(r) \text{ and } r \overset{S}{\supseteq} \pi_X^B(r) \bowtie^B \pi_Y^B(r) \ .$$

To show the first inclusion, let $u = (u_A, u_B, u_C, t) \in r$. By definition of π^B, $u_1 = (u_A, u_B, t_1) \in \pi_X^B(r)$ with $t_1 \supseteq t$. Also $u_2 = (u_A, u_C, t_2) \in \pi_Y^B(r)$ with $t_2 \supseteq t$. By the definition of $\bowtie^B$, $u' = (u_A, u_B, u_C, t') \in r$ with $t' = t_1 \cap t_2 \supseteq t$. Without use of the premise, this proves the inclusion.

To prove the second inclusion, pick two arbitrary tuples in the joining relations on the right hand side. Let $u_1 = (u_A^1, u_B, t_1) \in \pi_X^B(r)$ and $u_2 = (u_A^2, u_C, t_2) \in \pi_Y^B(r)$. The inclusion follows if we can show that the result of $u_1 \bowtie^B u_2 (= u')$ is a snapshot subset of r. If $u_A^1 \neq u_A^2$ or $t' = t_1 \cap t_2 = \emptyset$, the result of $u_1 \bowtie^B u_2$ is empty, and the relationship holds.

Otherwise, $u' = (u_A, u_B, u_C, t')$ where $u_A = u_A^1 = u_A^2$. Now, for each chronon $e \in t'$ we need to show that there is a tuple $u \in r$ with $u = (u_A, u_B, u_C, t)$ and $e \in t$. Tuple u_1 must be (partially) derived from a tuple $u_e \in r$ for which $u_e[A] = u_A$ and $u_e[B] = u_B$ and $e \in u_e[T]$ (since $t' \subseteq t_1$). Similarly, because $e \in t_2$, tuple u_2 must be (partially) derived from a tuple $u'_e \in r$ with $u'_e[A] = u_A$ and $u'_e[C] = u_C$ and $e \in u'_e[T]$. From the assumption, we know that $A \overset{T}{\rightarrow} X$, so $A \overset{T}{\rightarrow} B$. This temporal functional dependency, along with the presence of u_e in r, implies that $u'_e[B] = u_e[B] = u_B$ (since $u'_e[A] = u_e[A]$). This u'_e in r is the tuple u that we were looking for, proving the second inclusion.

Finally, assuming that $X \cap Y \overset{T}{\rightarrow} Y$ (instead of $X \cap Y \overset{T}{\rightarrow} X$) leads to a similar proof. □

Every concept defined above is applicable, as special cases, to both valid-time and transaction-time relations, using the appropriate temporal operators.

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: 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 instances 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: The proof of the first property follows from the first half of the previous proof.

The proof of the second property follows from the first property and the fact that if $r \overset{\text{S}}{\subseteq} r'$ are arbitrary instances of a relation schema R then $\pi_X(r) \overset{\text{S}}{\subseteq} \pi_X(r')$. To see this is true, let x^s_X be an arbitrary tuple in a snapshot at transaction time t' and valid time t of the left hand side relation. This means that there exists a tuple x_X with $(t', t) \in x_X[T]$ in the left hand side. Then, by the definition of projection, $\exists x \in r\ (x[X] = x_X \wedge (t', t) \in x[T])$. By the assumption, $\exists x' \in r'\ (x'[X] = x[X] \wedge (t', t) \in x'[T])$. Thus, applying the definition of projection, x^s_X is in the snapshot at transaction time t and valid time t' of the right hand side relation.

Finally, to prove the third property, assume that a tuple x^s_X belongs to $\tau^{\text{V}}_t(\rho^{\text{B}}_{t'}(\pi_X(\gamma \bowtie^{\text{B}} \lambda)))$. Then there is an x in $\pi_X(\gamma \bowtie^{\text{B}} \lambda)$ such that $x[X] = x^s_X \wedge (t', t) \in x[T]$. By the definition of $\bowtie^{\text{B}}$, $\exists x' \in \gamma(x'[X] = x[X] \wedge x[T] \subseteq x'[T])$. The existence of tuple x' means that $x^s_X \in \tau^{\text{V}}_t(\rho^{\text{B}}_{t'}(\gamma))$. □

In an entirely analogous way, by using the modified version of the relational operators given in Section 4.2 and the concept of snapshot equivalence, one can extend all the other properties of functional dependencies to hold for temporal functional dependencies. One can also define temporal variants of join dependencies [Rissanen 77], fifth normal form (also called project-join normal form) [Fagin 79], embedded join dependencies [Fagin 77], inclusion dependencies [Casanova et al. 81], template dependencies [Sadri & Ullman 82], domain-key normal form [Fagin 81], and generalized functional dependencies [Sadri 80]. 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. The result is a consistent and wholesale application of existing dependency and normalization theory to valid-time, transaction-time, and bitemporal databases.

5.5 Evaluation

It should be clear from the discussion in Section 5.3 that the bitemporal conceptual data model, with its associated definitions of functional dependency and normal forms, satisfies all five desiderata listed in Section 3.1. It should also be evident that the definition of key in this model satisfies all six desiderata listed in Section 3.2.

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

To define first temporal normal form (1TNF), a special valid timeslice operator was defined

that relies on the presence of the surrogate attribute [Segev & Shoshani 88]. The timeslice operators defined in Section 4.2 do not rely on any distinguished attribute.

Informally, 1TNF was defined to ensure that applications of the special valid timeslice result in a 1NF relation. Our valid timeslice operator (τ^{B}) always returns a 1NF relation when applied to a valid time relation. 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., $S \xrightarrow{T} R$. Thus, 1TNF is simply 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 [Navathe & Ahmed 89]. This aspect is not accommodated in our definitions of temporal normal forms.

Using our definition of temporal key, P Normal Form (PNF) [Lorentzos 91] will automatically be satisfied. Thus PNF is also simply an application of the concept of a key. Q Normal Form appears to have similarities with Navathe's concept of synchronicity, 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 [Navathe & Ahmed 89, Sarda 90A, Lorentzos 91]. Our approach provides a more natural, representation-independent extension of the snapshot normal forms.

Concerning keys, we formalized and extended the notions present in the HQL [Sadeghi et al. 87], HSQL [Sarda 90A], and IXRM [Lorentzos 91] data models, using the more general concept of temporal dependency. The concept of key in the TempSQL data model [Gadia 92] appears to be entirely representational, does not reflect the semantics of the time-varying data, and is not consistent with the concept of a snapshot key.

6 Application to Spatial Databases

The graphical representation of a bitemporal element as an area in the two-dimensional valid-time/transaction-time space (c.f., Figure 1) leads one to consider spatial databases, which are either two dimensional (e.g., index by x and y axes, say latitude and longitude over the surface of the Earth) (c.f., [Mark et al. 89]) or three dimensions (e.g., the third dimension being altitude or depth) (c.f., [Jones 89]). 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. In the following, we will focus on geographical applications that require modeling the surface of the Earth in two dimensions [Bracken & Webster 89].

Relations can be of several general types.

Space and time invariant — modeled with a valid-time element of "all time" (from beginning to forever) and a bispatial element of "everywhere." An example is the relation ParentOf = (Parent, Child).

Space invariant, time varying — modeled with a bispatial element of everywhere. An example is the Emp = (Name, Dept, PhNo | T) relation discussed in detail earlier.

Space varying, time invariant — modeled with a valid-time element of all time. An example is the relation Elevation = (Value | S), with a value at each point on the Earth (geographical databases do not consider geological processes that could change the elevation, so for all practical purposes it is time-invariant).

Space and time varying — modeled with an arbitrary bitemporal/bispatial element, a set of 4-dimensional quanta. An example is the Emp = (Name, Dept, PhNo | S-T) relation modeling employees that commute between two (or more) different company plants and who are possibly in different departments in different locations.

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 $X \rightarrow Y$ is intensional: it applies to all possible instances of the relation at all times and at all points in space. We can define 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 *spatial-temporal functional dependency*, denoted $X \overset{\text{SP,T}}{\rightarrow} Y$, that would take all space slices and time snapshots of all possible extensions. These would be natural extensions of snapshot dependencies, in that the following would hold.

$$X \rightarrow Y \Leftrightarrow X \overset{\text{T}}{\rightarrow} Y \Leftrightarrow X \overset{\text{SP}}{\rightarrow} Y \Leftrightarrow X \overset{\text{SP,T}}{\rightarrow} Y$$

Note that these various dependencies are being applied in the context of different data models, specifically the snapshot, bitemporal conceptual, spatial conceptual, and bitemporal spatial conceptual data models, respectively.

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

7 Conclusion and Future Research

In this paper, we have defined an consistent temporal extension of the concepts of functional dependencies, keys and normal forms. We demonstrated that our definitions are more natural extensions than those previously proposed. 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 spatial-temporal databases, allowing temporal and spatial database design to closely track conventional database design.

We want to emphasize the 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. Secondly, we focused on the semantics of temporal relations rather than their representation. The concepts apply globally, across most if not all existing representational temporal data models. Our use of snapshot equivalence makes this possible as representational models provide means of computing snapshots. As a result, new concepts are not needed for each representational data model. Finally, we chose a simple data model that has no intentions of being suitable as a basis for display or implementation (existing models may be used for these important tasks). The model has the important feature that relation instances with the same information content are identical. Our approach effectively moves the distinction between the various data models from a semantic basis to a physical, performance-relevant basis.

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 [Navathe & Ahmed 89], 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.

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 grant ISI-8902707 and IBM contract #1124. In addition, the first author was supported by Danish Research Academy grant 11–8696–1 SE and 11–9675–1 SE. We appreciate the helpful comments from Curtis E. Dyreson and Nick Kline on a previous draft.

References

[Ahmed 92] R. Ahmed. Personal Communication, March 1992.

[Bracken & Webster 89] I. Bracken and C. Webster. Towards a Typology of Geographical Information Systems. *International Journal of Geographical Information Systems*, 3(2):137–152, 1989.

[Ben-Zvi 82] J. Ben-Zvi. *The Time Relational Model*. PhD thesis, Computer Science Department, UCLA, 1982.

[Casanova et al. 81] M. Casanova, R. Fagin, and C. Papadimitriou. Inclusion Dependencies and Their Interaction with Functional Dependencies. In *Proceedings of the Conference on Principles of Database Systems*, 1981.

[Clifford & Croker 87] 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.

[Clifford & Warren 83] J. Clifford and D.S. Warren. Formal Semantics for Time in Databases. *ACM Transactions on Database Systems*, 8(2):214–254, June 1983.

[Codd 72] 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.

[Codd 72B] 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.

[Codd 74] E. F. Codd. Recent Investigations in Relational Database Systems. In *Proceedings of the IFIP Congress*, 1974.

[Dyreson & Snodgrass 92] C. E. Dyreson and R. T. Snodgrass. Time-stamp Semantics and Representation. TempIS Technical Report 33, Computer Science Department, University of Arizona, February 1992.

[Fagin 77] R. Fagin. Multivalued Dependencies and a New Normal Form for Relational Databases. *ACM Transactions on Database Systems*, 2(3), September 1977.

[Fagin 79] R. Fagin. Normal Forms and Relational Database Operators. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1979.

[Fagin 81] R. Fagin. A Normal Form for Relational Databases that is Based on Domains and Keys. *ACM Transactions on Database Systems*, 6(3), September 1981.

[Gadia 88] S.K. Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, 13(4):418–448, December 1988.

[Gadia 92] 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.

[Jensen et al. 91] 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.

[Jensen & Snodgrass 92] C. S. Jensen and R. T. Snodgrass. Proposal of a Data Model for the Temporal Structured Query Language TempIS Technical Report No. 37, Computer Science Department, University of Arizona, July 1992.

[Jensen et al. 92] C.S. Jensen, J. Clifford, S.K. Gadia, A. Segev, and R.T. Snodgrass. A Glossary of Temporal Database Concepts. *SIGMOD Record*, 21(3), September 1992.

[Jensen et al. 92A] C. S. Jensen, M. D. Soo, and R. T. Snodgrass. Unification of Temporal Relations. Technical Report 92-15, Computer Science Department, University of Arizona, July 1992.

[Jones et al. 79] S. Jones, P. Mason, and R. Stamper. Legol 2.0: A Relational Specification Language for Complex Rules. *Information Systems*, 4(4):293–305, November 1979.

[Jones 89] C.B. Jones. Data Structures for Three-dimensional Spatial Information Systems in Geology. *International Journal of Geographical Information Systems*, 3(1):15–31, 1989.

[Korth & Silberschatz 86] H. F. Korth and A. Silberschatz. *Database System Concepts*. McGraw-Hill Advanced Computer Science Series. McGraw-Hill Book Company, 1986.

[Lorentzos & Kollias 89] N. Lorentzos and V. Kollias. The Handling of Depth and Time Intervals in Soil-information Systems. *Computers and Geosciences*, 15(3):395–401, 1989.

[Lorentzos 91] N. Lorentzos. Management of Intervals and Temporal Data in the Relational Model. Technical Report 49, Agricultural University of Athens, 1991.

[Mark et al. 89] 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.

[Navathe & Ahmed 89] S. B. Navathe and R. Ahmed. A Temporal Relational Model and a Query Language. *Information Sciences*, 49:147–175, 1989.

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

[Rissanen 77] J. Rissanen. Independent Components of Relations. *ACM Transactions on Database Systems*, 2(4), December 1977.

[Sadeghi et al. 87] R. Sadeghi, W.B. Samson, and S.M. Deen. HQL — A Historical Query Language. Technical report, Dundee College of Technology, Dundee, Scotland, September 1987.

[Sadri 80] F. Sadri. *Data Dependencies in the Relational Model of Data: A Generalization.* PhD thesis, Princeton University, October 1980.

[Sadri & Ullman 82] 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.

[Sarda 90A] N. Sarda. Algebra and Query Language for a Historical Data Model. *The Computer Journal*, 33(1):11–18, February 1990.

[Sarda 90B] N. Sarda. Extensions to SQL for Historical databases. *IEEE Transactions on Knowledge and Data Engineering*, 2(2):220–230, June 1990.

[Snodgrass & Ahn 86] R. Snodgrass and I. Ahn. Temporal Databases. *IEEE Computer*, 19(9):35–42, September 1986.

[Segev & Shoshani 87] A. Segev and A. Shoshani. Logical Modeling of Temporal Data. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 454–466, San Francisco, CA, May 1987.

[Segev & Shoshani 88] 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.

[Snodgrass 87] R. T. Snodgrass. The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

[Snodgrass 92] R.T. Snodgrass. Temporal Databases. In *Proceedings of the International Conference on GIS: From Space to Territory*, Pisa, September 1992. Springer-Verlag.

[Ullman 88] J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, Volume 1. Computer Science Press, Potomac, Maryland, 1988.

[Zaniolo 76] C. Zaniolo. *Analysis and Design of Relational Schemata for Database Systems.* PhD thesis, UCLA, July 1976.