TERC+: A Temporal Conceptual Model

Esteban ZIMANYI, Christine PARENT, Stefano SPACCAPIETRA
Ecole Polytechnique Fédérale de Lausanne
Laboratoire de Bases de Données, IN-Ecublens, 1015 Lausanne, Switzerland
e-mail: {Christine.Parent,Stefano.Spaccapietra,Esteban.Zimanyi}@di.epfl.ch

Alain PIROTTE
Université catholique of Louvain,
IAG Management School,
1 place des Doyens, 1348 Louvain-la-Neuve, Belgium,
e-mail: pirotte@info.ucl.ac.be

Abstract

Much work has been done recently on the design of a temporal extension to SQL-92, called TSQL2. Similar work is being done to add temporality to the forthcoming SLQ3 standard. Still, TSQL2 is based on the relational model, which is hardly adequate for conceptual modeling of any but the simplest applications. This paper presents a temporal conceptual model, called TERC+, that can be used for developing applications with temporal features in a practical and realistic way. The model allows to describe application data structures independently of implementation concerns. We then define the semantics of TERC+ as a mapping to relational schema constructs.

Keywords: conceptual modeling, temporal applications, temporal database management, relational implementation

1 Introduction

More than thirty years of experience in database design have clearly shown that user requirements of complex applications are best captured in conceptual models. Conceptual modeling adds two significant advantages to straight relational or object-oriented design: (1) it allows designers to focus on their main problem (namely, the representation of application data and processes on the data) with minimal concern for and interference from technical constraints, and (2) its result (a conceptual model of data and processes) is more stable in time than implementation-oriented models that become obsolete with any change in the target implementation platform.

In addition, conceptual models provide a better support for visual user interfaces. Entity-relationship (ER) models [6, 29], for instance, enable users to easily visualize database schemas and thus master the information content of the database. Such models also support direct manipulation techniques (e.g., point, click and drag elements on a screen) to browse the database, and express queries and updates without the burden of the more elaborate syntax of a textual language [12].

Conceptual modeling also facilitates information exchange over the Internet and within heterogeneous distributed or federated databases. In such contexts, a conceptual model provides the best vehicle for a common understanding among partners with different technical and application backgrounds. Abstracting from technicalities greatly facilitates a global consistent description of all relevant information [25].

Many applications require data not only about the present but also about the past and the future. Modeling the temporal evolution of data often provides interesting insight into the dynamics of real-world phenomena.

Much work has recently been devoted to temporal databases, in particular to data modeling, query language design, standardization, and implementation issues. Many temporal models were proposed, mainly for relational and object-oriented models, although some work has also been done for entity-relationship models. Important bibliographies on temporal databases [1, 5, 16, 20, 28, 30, 34], surveys [4, 7, 21, 26], and books [10, 35] were published, witnessing an important research activity in the field.

International Symposium on Digital Media Information Base (DMIB'97), Nara, Japan, November 1997

A step towards the practical availability of temporal solutions was the design of a temporal extension to SQL-92, called TSQL2 [27]. Similar work is being done to add temporality to the forthcoming SLQ3 standard. Still, TSQL2 is based on the relational model, which is hardly adequate for conceptual modeling of any but the simplest applications. Thus temporal extensions of richer models are also needed. We have developed one such extension, called TERC+, of the ERC+ [29] entity-relationship model.

TERC+ is the kernel of a practical and realistic methodology for developing applications with temporal features. The methodology comprises three major steps:

(1) conceptual specification, where application requirements are described in TERC+, independently of implementation concerns;
(2) relational or object-oriented implementation of the conceptual schema;
(3) optimization and tuning of the relational representation, involving trade-off decisions between storage space and efficiency of query processing according to application-dependent criteria.

This paper focuses on the first step. Section 2 presents the TERC+ model and discusses temporal issues for conceptual data modeling in general. Section 3 defines the semantics of TERC+ schemas in terms of the semantics of the ERC+ model, by making explicit the information implied by the temporal features. As ERC+ is known to be translatable into relational specifications, an implementation of TERC+ on a traditional, atemporal relational DBMS is straightforward. Section 4 compares TERC+ with other temporal conceptual models. Finally, Section 5 concludes the paper and suggests directions for further research.

2 The TERC+ Conceptual Model

Conceptual models have been used for more than twenty years for expressing user requirements when developing database applications. Conceptual models allow users and analysts to concentrate on essential aspects of an application domain, without being bothered by constraints of a specific implementation platform. Thus, conceptual models provide descriptions that are closer to a human perception of events in the real world and that facilitate man-machine communication. Users need not transform their intuitive specifications to adjust to the technological constraints of specific systems.

As a formalism, or language, for expressing conceptual models, TERC+ aims at supporting the expression of information requirements independently of technological concepts. In addition, TERC+ aims at the following general modeling objectives:

- **completeness**: conceptual models must be able to describe all usual types of temporal applications; they must be able to integrate nontemporal and temporal data; temporality must support instantaneous events as well as facts valid for some period of time;
- **soundness**: concepts must rely on a formal definition, to avoid ambiguities due to incomplete or imprecise specifications;
- **user orientation**: conceptual models must allow easy communication with users, preferably supported by schema diagrams; they must be understandable with reasonable effort, through a reasonable number of intuitive concepts;
- **orthogonality** (or independence of concepts): constructs must be as independent as possible from one another, to make the modeling language easy to understand and use;
- **implementability**: the modeling language must be directly translatable into logical data models of existing DBMS's, so that conceptual models can effectively serve as common, pivot data schemas in a federation of heterogeneous systems;
- **full operationality**: the modeling formalism must include a data manipulation language, integrated in the paradigm of the data definition language, that can act as a common language for communicating between the various DBMS's in a federation.

The following subsections review the data modeling capabilities of TERC+ in terms of traditional and temporal features. Process modeling is out of the scope of the paper.

2.1 Traditional Structural Features

TERC+ supports a typical collection of basic concepts: entities, relationships, attributes, is-a links, aggregation links, and associated integrity constraints. These concepts can be defined briefly as follows:

- An **entity** represents an object of interest from the real world.
- An **entity type** represents a set of entities with similar structure and behavior.
- A **relationship** represents a link of interest between two or more entities, where each entity plays a given role.
- A **relationship type** represents a set of links with similar characteristics (linking entities of the same types, with the same roles, and similar properties).

- An **attribute** represents a real-world property of interest; both entity types and relationship types may have attributes. Attributes can be:
  - **simple** (with atomic values) or **complex** (or structured, i.e., composed of simple or complex attributes); the complex attribute is sometimes called composite, while its parts are called components, by analogy with aggregation (see below);
  - **monovalued** (with a single value) or **multivalued** (with a multiset value);
  - **mandatory** (with a value in every instance) or **optional** (with a value in some instances and no value in others).

  The mono-/multivalued and optional/mandatory characteristics of an attribute are integrity constraints expressed with the **minimum/maximum cardinality** concept. Cardinalities also apply to roles in relationship types, to define how many (at least, at most) instances of a relationship type may link an entity of the associated type.
  The value of an attribute may be computable, i.e., automatically inferred by the system, from values in other attribute(s) that belong to the same entity or relationship, or to several entities or relationships. Such an attribute is called a **derived attribute**. The derivation function may invoke both computations and navigations through the database.
- An **is-a relationship**, or generalization, relates two entity types, a **supertype** and a **subtype**. An is-a link between two entities expresses that both entities represent the same real-world object, in such a way that an entity can exist in the population of the subtype only if it also belongs to the supertype. Inherited properties (attributes, relationships, temporal characteristics) can be refined: their domain in the subtype can be a subdomain of the corresponding property in the supertype. Additional constraints can be expressed on generalizations, as usual:
  - a generalization is said to be **exclusive** if an entity cannot belong to more than one subtype; otherwise it is **overlapping**;
  - a generalization is said to be **total** if every entity belonging to the supertype also belongs to a subtype; otherwise it is **partial**.
- An **aggregation** link is a special directed binary relationship whose semantics expresses that entities of a type, called **composite** entities, represent aggregates of entities of another type, called **component** entities. Alternative terms for aggregation in object-oriented models include **composition link** and **part-of relationship**. As for generic relationships, cardinalities are attached to the composite and component roles. Additional constraints can be expressed on aggregations [18], as follows:
  - an entity type in an aggregation (whether the composite or the component) is said to be **dependent** on the other entity type if, when one entity of the latter type ceases to exist, all entities of the former type to which it is related through the aggregation are also destroyed; otherwise it is **independent**;
  - a set of aggregations is said to be **exclusive** for a component entity type if each entity of that type can participate in at most one instance of aggregation from the set of aggregations; otherwise, the set of aggregations is **shared**; exclusiveness on a single aggregation is equivalent to a maximal cardinality of 1.

2.2 Temporal Features

There are two ways of adding time to information. The usual way consists in **timestamping** attributes, entities, and/or relationships. Thus, attribute values can have a period of validity, and life cycles of entities and relationships can be remembered (i.e., when they are created, deactivated, reactivated, and deleted). The other association of time with information concerns the **relative positioning of activities in time**, that models aspects of the inter-object dynamics of applications (e.g., that entity has created that other entity; that entity lived before that other one; this entity is a snapshot of that other one). Although such relationships are more rarely supported, they are important in particular for applications related to the management, the analysis and the understanding of natural and human phenomena. They are discussed in Section 2.7.

Three complementary and independent viewpoints on timestamping are customary. **Transaction time** consists of system-generated timestamps recording the point in time when a fact was introduced into or deleted from the database. Values for **valid time** are user-provided and represent the actual time of occurrence of an event or the period of time where a fact is valid in the real world. Valid time may span the past, the present, and the future. Valid time is the most common requirement for usual applications. **User-defined time** refers to a data type that also codes time values, e.g., a DATE domain, but whose temporal semantics is not supported by the DBMS.

The TERC+ approach to temporal modeling adheres to the following principles:

- focus is on valid time (but the approach is easily extendible to transaction time);
- both snapshot (nontemporal) and time-varying (temporal) data are supported;
- temporal facts include both instantaneous events and facts lasting over periods of time;
- orthogonality (or independence): temporality can be attached to any construct of the model, i.e., to entities, relationships, and attributes (at any level);
- consistency rules enforce a correct semantics of temporality;
- the TSQL2 approach is adopted whenever applicable and appropriate.

We adopt a linear discrete representation of time: the time axis is seen as a sequence of time points isomorphic to the set of natural numbers. Time points are called **instants**.

TERC+ supports valid time at different **granularities** (e.g., year, month, day, hour, minute, second). Temporal support requires functions to convert from one unit to another.

Temporal specifications distinguish instants, intervals, and temporal elements. An **instant** or **chronon** typically represents the time of occurrence of an event. An **interval** is a time slice, typically specifying the validity period of a fact. An interval is defined by a **start chronon** and an **end chronon**. An interval with the same start and end chronons defines an instant. A **temporal element** is the union of a set of disjoint intervals, representing the set of time slices where a fact is valid. As in TSQL2, temporal elements may be restricted to contain only instants (or event elements) or only intervals of duration greater than one chronon (or state elements).

One special chronon called *now* is useful to express the meaning "valid until further notice" [9]. Thus, a fact timestamped with the interval [12/95,*now*] will be valid until some undefined instant in the future.

Temporal elements may be compared using Allen's interval logic [2], which defines a complete set of basic Boolean operators for the relative positioning of time intervals (namely, *before, equals, meets, overlaps, during, starts,* and *finishes*). Most of these operators were included in TSQL2.

2.3 Temporal Attributes

Temporal (i.e., timestamped) attributes record the history of their values. For example, in Figure 1 (a), salary is a temporal attribute of the Employee entity type. For each employee, the DBMS keeps track of all values of salary that have been, are, or are planned to

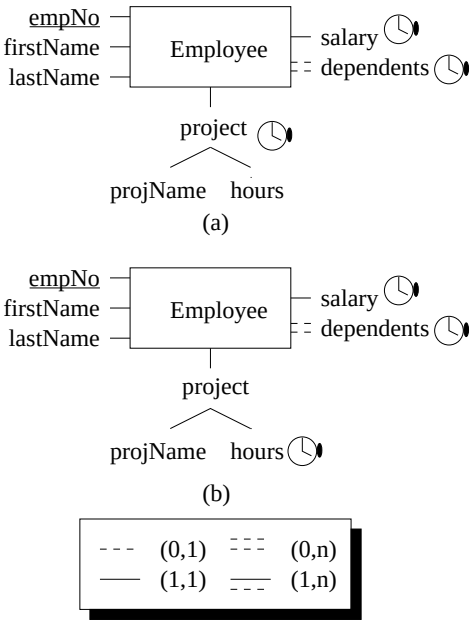

Figure 1: Temporal attributes.

be valid together with the time, or time interval, of validity. Thus, in general, each value of a temporal attribute is associated with a temporal element. When defining an attribute as temporal, its granularity must be specified.

For example, assuming that the granularity for salary is month, the salary of a given employee could have value {5000@[4/96,6/96], 6000@[9/96,11/96]} meaning that he/she earned 5000 from 4/96 to 6/96 and 6000 from 9/96 to 11/96. The value of an attribute is assumed to be undefined (unknown or inapplicable) for the points in time not included in its temporal element.

Multivalued attributes may also be temporal, like attribute dependents of Employee. Here a temporal element is associated with each set of values of the attribute. For example, the dependents of employee Fred could be {John,Mary}@[9/96,12/96] and {John,Mary,Peter}@[1/97,7/97]. Alternative (and, in our opinion, less intuitive) representations of the association between a multivalued attribute and a time element include the following:

- possibly overlapping time intervals, each associated with a simple value (e.g., John@[9/96,7/97], Mary@[9/96,7/97], Peter@[1/97,7/97]);
- sets of simple values associated with the time interval (or temporal element) that describes the validity of each set (e.g., {John,Mary}@[9/96,7/97], {Peter}@[1/97,7/97]).

Temporality can be attached to any level in a com-

plex attribute. For example, project in Figure 1 (a) is a temporal complex attribute of Employee composed of projName and hours, where the latter is the average number of hours spent per week by the employee on the project. A value of project is a set of pairs ⟨projName, hours⟩, with each pair carrying a temporal element. If only the component attribute hours is temporal, as in Figure 1 (b), then a value for project will contain one value for projName and a set of values for hours, each with an associated temporal element.

A complex temporal attribute may itself have temporal components. For example, both project and hours could be defined as temporal. This would allow to express, e.g., that employee John worked on project Super-G from 2/95 to 12/95, and that he worked on that project 20 hours from 2/95 to 6/95 and 36 hours from 7/95 to 12/95.

Constraints on complex attributes may refer to the temporal elements associated with components and composite. They can be constructed from Allen's operators. Special constraints, called inclusion, covering, and equality, were found particularly useful in real-world modeling.

The **inclusion** constraint states that the temporal element of a value of either the composite or the component must be included in the temporal element of the associated value of the other attribute. For example, if both project and hours are temporal, an inclusion of hours into project could require that the temporal element of each value of hours be included in the temporal element of the associated project.

The **covering** constraint states that the temporal elements associated to the values of one attribute must be included in the *union* of temporal elements associated to the values of the other attribute. For example, a covering of project by hours will ensure that at any instant included in the temporal element of a project, there is an associated value of attribute hours.

The **equality** constraint is equivalent to an inclusion and a covering constraint holding together on the same attributes: it states that the union of the temporal elements associated to the values of one attribute must be equal to the union of temporal elements associated to the values of the other attribute.

As application needs can be very different, no constraint is specified by default between the temporal elements of the composite and component values. As shown in later sections, the same constraints can be defined (1) between a temporal entity or relationship and its attributes; and (2) between an aggregated entity and its components.

Cardinalities attached to attributes are interpreted as **static**, i.e., they define the number of possible values of the attribute at any point in time. A **temporal cardinality**, written $h(max)$, may be defined to constrain the maximum number of value changes of the attribute over the life cycle of its entity.

For example, attribute project has static cardinality (1,1), meaning that, at any point in time, an employee is attached to exactly one project. A temporal cardinality $h(3)$ states that an employee will be attached to at most 3 projects altogether during his/her life cycle.

For the sake of simplicity, temporal cardinalities are allowed only on temporal attributes. Also, by default, the temporal cardinality is $h(n)$.

Temporality of attributes governs their updating. Consider, for Figure 1 (b), an employee e having as value of project ⟨proj1, {20@[9/95, 6/96], 25@[7/96, 9/98]}⟩. Updates of project include the following:

- change the name of the project: the update concerns projName and the values of hours are not affected;
- introduce information for another project for employee e, say, ⟨proj2, {35@[7/97, now]}⟩, meaning that, from 7/97 on, employee e has been attached to project proj2 for 35 hours per week; as complex attribute project is nontemporal, the information about proj1 will be erased by the update, as only the latest information is kept for a nontemporal attribute.

2.4 Temporal Entities

When associated to entity types, temporality concerns the existence of the entities within their types rather than values. Entities are created as instances of an entity type, but they can migrate to another entity type, be temporarily suspended as instances of their entity type, be resumed in their membership of the type, and eventually be deleted.

Defining an entity type as temporal instructs the DBMS to keep track of the life cycle of its instances, as affected by creation, suspension, reactivation, and deletion events. For each temporal entity, this information remains available after its deletion. So does the latest value of each attribute, in order to allow users to associate life-cycle information to the corresponding real-world object.

More precisely, the life cycle of a temporal entity comprises three states (active, suspended, and dead[1]),

[1]The dead status is associated with a degenerated temporal

each associated with a temporal element. In particular, the active periods of an entity is the temporal element associated to the active status in the life-cycle information.

For example, specifying an entity type Employee as temporal requires to keep track of the periods at which each employee has worked and will work in the company, as well as of the periods where the employee's activity was or will be suspended (e.g., for a temporary leave of absence).

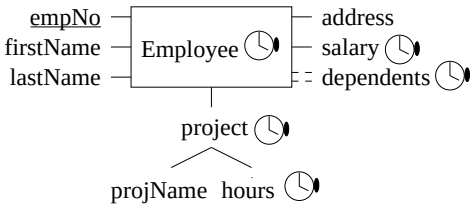

Figure 2: Temporal entities.

Defining an entity type as temporal is independent of defining some of its attributes as temporal. For temporal entity types without temporal attributes, like Employee in Figure 2 (a), the life cycle of each entity is kept together with one value per attribute. Nontemporal attributes may either be constant over time (like, e.g., firstName, lastName), or may change (like, e.g., address, salary, or dependents), without the applications being interested in keeping track of past values. Thus, it is possible to record, e.g., that employee John Smith was in the company from 6/90 to 1/97, while retaining only the latest known values for address, salary, and dependents.

On the other hand, for temporal entity types with temporal attributes, like Employee in Figure 2 (b), in addition to recording the life cycle of each entity, the history of attribute values is recorded for each temporal attribute (e.g., salary, dependents).

As for complex temporal attributes, several constraints (like inclusion, covering, or equality) may be defined between the temporal elements associated to a temporal entity and a temporal attribute. These constraints would allow to constrain, e.g., the valid time associated with values for salary with respect to the life cycle of its associated employee.

The notion of identifier must be revisited for temporal entity types. In Figure 2, empNo is an identifier of Employee. Two interpretations are possible: (1) at any point in time, an empNo value designates a unique employee, but the same empNo may designate different employees at different points in time (e.g., if the empNo of a former employee is reused for a new one); (2) an empNo value will only ever be associated with a single employee; this interpretation holds by default.

For currently valid temporal entities, two different types of deletions can be considered. Physical deletion typically concerns errors (e.g., when an entity was erroneously introduced in the database). Conceptual deletions, on the contrary, correspond for example to a temporal entity ceasing to belong to an entity type. This amounts to modifying the temporal element of the entity, to make currently invalid its membership in the entity type.

2.5 Temporal Entities and Generalization

Entity types (whether temporal or not) can be connected by generalization links. The semantics of generalization has to be adapted when it relates temporal and nontemporal entity types.

Consider a nontemporal entity type Employee and a temporal subtype Manager, as in Figure 3 (a). The intended semantics is that we want to keep information only about current employees, but we want to keep information about past, present, and future managers. More specifically, as shown in Figure 3 (b), Manager inherits a set of attributes from Employee in addition to defining its own attributes. Therefore, for past, present, and future managers, it is necessary to keep the values for all own and inherited attributes.

As shown in Figure 3 (c), a temporal entity type cannot have a nontemporal subtype. The reason is that, in a generalization link, a supertype entity can always be replaced by a subtype entity (substitution principle).

Entity types can take part in generalization hierarchies like that of Figure 4. Two parallel generalizations are defined according to qualification and type of contract. In this case, the life cycle of an entity in any subtype must be included in that of its supertype(s). The exclusive/overlapping and total/partial constraints defined on generalizations involving temporal entity types imply that:

- for exclusive (resp., overlapping) generalizations, the active periods of an entity in a group of related subclasses must be disjoint (resp., may intersect);
- for total (resp., partial) generalizations, the union of active periods of an entity in a group of related subclasses must equal (resp., include) the active periods in its superclass.

As shown in Section 2.7, objects can migrate along such generalization hierarchies. The transition dynamic relationship allows to model such changes.

element, which is either empty or an interval whose end point is infinitely remote in the future.

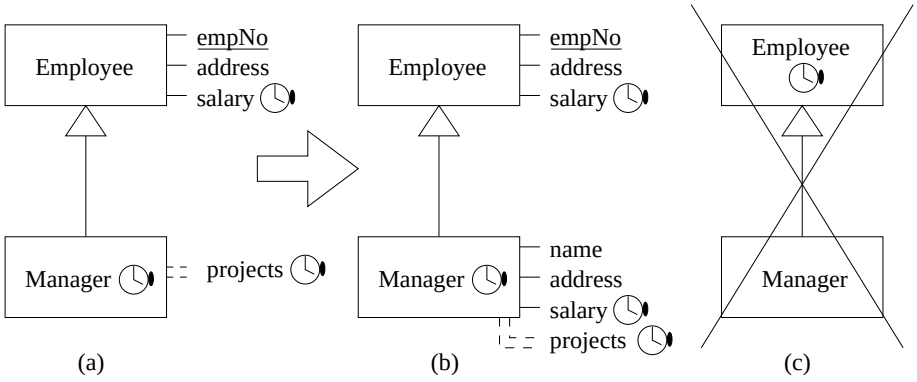

Figure 3: Temporal Entities and Generalization.

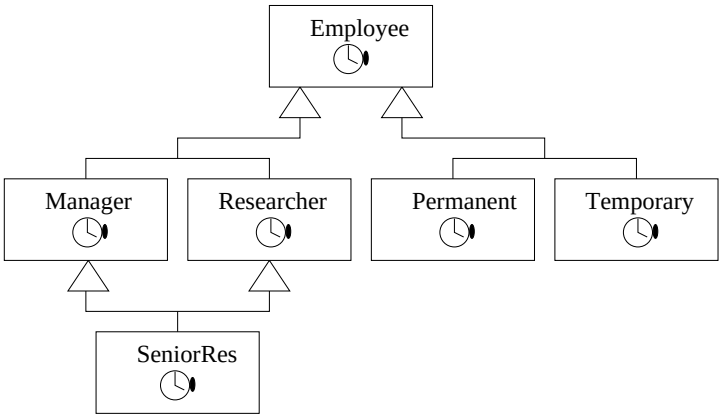

Figure 4: Temporal Generalization Hierarchies.

2.6 Temporal Relationships

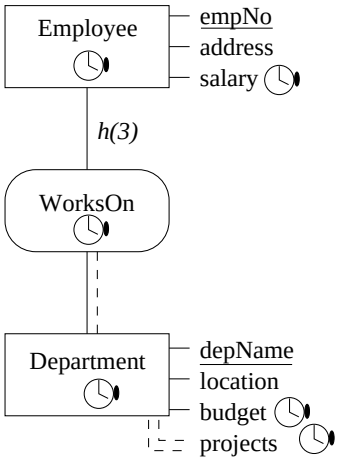

Figure 5: Temporal relationships.

As for temporal entities, temporality of relationship types (in the timestamping sense) does not concern values. Instead, temporality of a relationship type involves keeping track of the life cycle of its instances. Relationship instances can be created, suspended, resumed, and deleted.

For example, in Figure 5, a temporal relationship type WorksOn links the Employee and Department entity types. Past, present, and future ⟨employee, department⟩ pairs associated in the relationship are kept in the temporal database, i.e., which employees were, are, and will be attached to which departments.

We assume, for simplicity, that a temporal relationship type can only link temporal entity types. This simplifies the management of constraints preventing dangling references, i.e., a relationship instance that would attempt to link entities about which information is no longer present in the database. If relationship WorksOn was temporal and entity type Employee nontemporal, the deletion of an employee emp1 would require the deletion of the instance of WorksOn concerning emp1. That would affect the very notion of temporality of WorksOn, that could no longer strictly record the history of the relationship.

For temporal relationships on temporal entities,

the active periods of each relationship instance must be included in the intersection of the active periods of the participating entities.

As for attributes, cardinalities linking temporal entity types to temporal relationship types are interpreted as **static**, i.e., they define the number of possible links from an entity playing the role in the relationship at any point in time. **Temporal cardinalities**, written $h(max)$, define the maximum number of possible links from an entity playing the role in the relationship over the life cycle of the entity. By default, the temporal cardinality on a role is $h(n)$.

For example, in Figure 5, Employee is related to relationship WorksOn with static cardinality (1,1), meaning that at any point in time an employee is related to exactly one department. The temporal cardinality $h(3)$ means that an employee can be attached to at most 3 different departments along its life cycle.

Consider now nontemporal relationships linking temporal entity types. Suppose that, in Figure 5, Employee and Department are temporal entity types while relationship WorksOn is nontemporal. The intended semantics is that, although the database keeps track of past, present, and future employees and departments, it only keeps track of currently valid $\langle \mathsf{employee}, \mathsf{department} \rangle$ pairs in relationship WorksOn. Therefore, the database must ensure that employees and departments participating in the relationship are currently valid.

As with temporal attributes, semantically different updates can affect temporal relationships, depending on the intended application semantics. Consider an instance $\langle \mathsf{emp1}, \mathsf{dep1} \rangle @ [9/95, 11/96]$ of relationship WorksOn. One possible update could link the same employee with another department while keeping the same instance in the relationship (e.g., if the information was inaccurate). Another update could relate the employee to a new department and add a tuple like $\langle \mathsf{emp1}, \mathsf{dep2} \rangle @ [12/96, \mathsf{now}]$.

An example of temporal aggregation is shown in Figure 6, modeling the administrative partitioning of a country (e.g., Switzerland) as a set of counties, each decomposed into municipalities, in turn decomposed into districts. The three entity types and the aggregation links are modeled as temporal to keep track of history (e.g., new districts or municipalities are created by splitting larger units, while others disappear by fusion; the membership of municipalities in counties varies with time, e.g., as a result of a referendum).

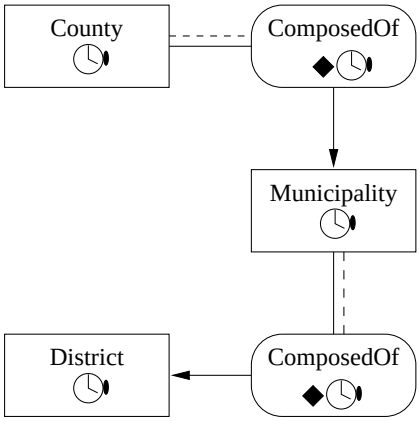

Figure 6: Temporal aggregation.

2.7 Dynamic Relationship Modeling

Dynamic aspects of real-world application objects are crucial for conceptual modeling. This section deals with the description of inter-object dynamics where time plays an essential role. Temporal links are expressed as relationship types. Four kinds of relationships are proposed, each representing a specific dynamic phenomenon:

- **transition relationships** express the behavior of entities changing their classification. For example, upon graduation, a person ceases to be an instance of the student subtype and becomes an instance of the alumnus subtype;
- **generation relationships** express generation of entities by other entities. For example, in a land management application, a parcel can be split creating several smaller ones, or, alternatively, several parcels can be merged into a new larger one;
- **timing relationships** describe temporal relationships between entities, e.g., before/after or during. For example, a storm preceded a landslide;
- **time-based aggregations** link temporal entities to their snapshots. This can be useful when integrating a temporal database and a snapshot one, describing the same entities.

These four dynamic relationship types are now discussed in more detail. Like any relationhip type, each may be named, have attributes, be involved in derivation formulas and integrity constraints.

2.7.1 Transition relationships

An object undergoes a transition as instance of a source class to become an instance of a target class.

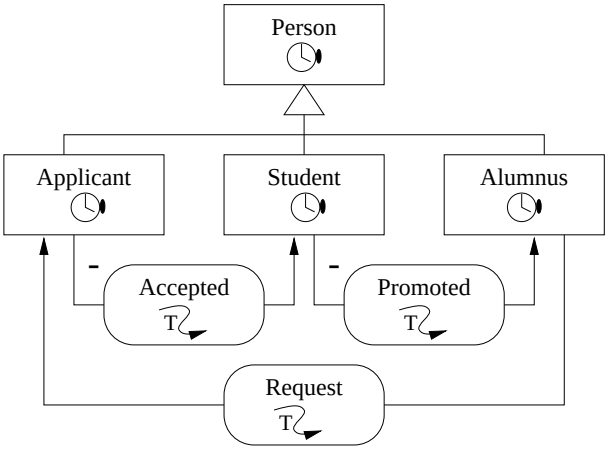

Figure 7: Transition dynamic relationships.

The transition abstraction represents a *becomes-a* relationship. It is a dynamic identity relationship, i.e., a dynamic link between objects that share the same identity. This requires that the source class and the target class be members of the same generalization hierarchy.

There are two types of transition, based on whether or not the object undergoing the transition is preserved as an instance of the source class. An **evolution** occurs when the transition entity ceases to be an instance of the source class. An **extension** occurs when the entity remains an instance of the source class. In both cases, a transition of an entity from one class to another is realized either when the user explicitly requests it, or when it can be deduced automatically, for example, in the case of specializations defined by predicates on the values of an attribute.

Figure 7 shows several examples of the transition relationship. There are two evolutions: from Applicant to Student and from Student to Alumnus. The minus sign on the source side indicates that the entity ceases to be an instance of the source class. Also, there is an extension from Alumnus to Applicant, meaning that an alumnus may become an applicant while still belonging to the Alumnus class.

2.7.2 Generation relationships

Generation relationships represent processes that lead to the emergence of new objects: an instance (or a set of instances) of a source class generate(s) an instance (or a set of instances) of a target class. Generations represent a *yields-a* relationships. It is useful for modeling causal and temporal relationships involved in the appeareance and disappearance of objects in the world.

As with transition, two different types of generation can be distinguished, depending on whether or not the source instances are preserved. A **transformation** occurs when the instance of the source class is consumed in the generation process. A **production** takes place when the source instance survives the generation process. Notice that for M:N generations, i.e., M objects which interact to result into N other objects, some input objects may be consumed while others are not.

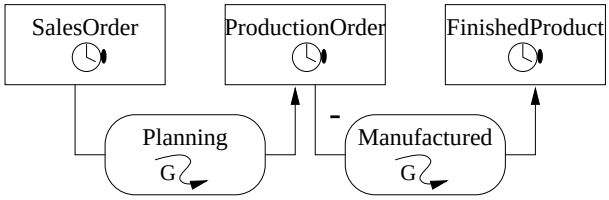

Figure 8: Generation dynamic relationships.

In Figure 8, sale orders generate production orders, but they are preserved after the generation. On the contrary, production orders generate finished products and are consumed in the process.

2.7.3 Timing relationships

We propose this binary relationship to specify constraints on the temporal elements of the participating entities. It is associated to the temporal operators *before, equals, meets, overlaps, during, starts,* or *finishes*. For example, relationship Causes could be defined between temporal entity types Landslide and Typhoon (Figure 9). This relationship can be described as a timing relationship of type *during*: it allows to state that instances of Landslide may be valid only during the validity period of the associated instance of Typhoon.

Allen's 13 operators define all basic temporal relationships between two intervals [2]. Many applications need less precise relationships, like the derived *dur* operator whose meaning is *during, starts or finishes*. Moreover life cycles are not mere intervals, but comprise three states (active, suspended, and dead), each associated with a temporal element. Allen's operators can be extended to the comparaison of life cycles in two main ways: comparing the whole duration of life cycles (i.e., timestamps of birth and death) or comparing active periods. For example, during$(e1, e2)$ can mean either:

- the birth of $e1$ takes place after the birth of $e2$, and the death of $e1$ takes place before the death of $e2$, or
- the temporal element of the active state of $e1$ is included in the temporal element of the active state

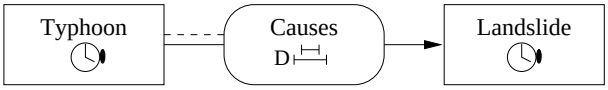

Figure 9: Example of temporal relationship Causes associated with operator during.

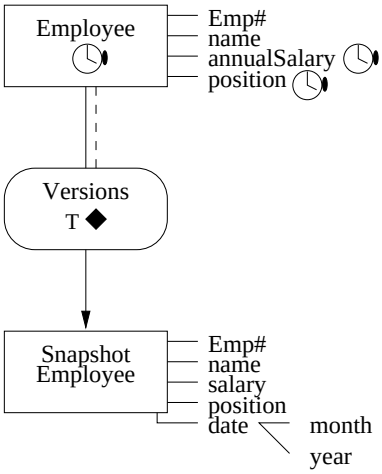

Figure 10: Time-based aggregation.

of $e2$; i.e. whenever $e1$ is active, so is $e2$.

We propose to allow the designer to define the exact meaning of each of the timing relationships needed by the application through temporal formulas. A formal language will be provided allowing to express any condition on the life cycles and the temporal elements.

Timing relationships may be computed from the life cycles recorded into the database. However, it is useful for some applications to stress these timing relationships, especially when the participating entities represent processes. Also, attributes may need to be attached to such relationships.

2.7.4 Time-based aggregations

Time-based aggregation is a special case of aggregation, where the composite entity type is temporal while the component is nontemporal, with the meaning that composite entities are snapshots of the component entity. A derivation formula specifies how the life cycle (and probably attributes) of the composite entity is derived from the component attributes, as in Figure 10 where, for an instance e of Employee, the life cycle and attributes annualSalary and position of e are derived from the instances of Snapshot Employee related to e by the time-based aggregation Versions.

3 Semantics of TERC+

The semantics of TERC+ can be expressed in terms of the semantics of the ERC+ model by making explicit the information implied by the temporal features.

Figure 11 shows the transformation that allows to replace temporal attributes. Generally speaking, a temporal attribute a at any level, with cardinalities (i, j) and $h(k)$, is replaced by a complex attribute a with cardinality (i, k), composed of component attributes value $(1, j)$ and valid $(1, k)$. The valid attribute represents the temporal element associated to the value; it is a complex multivalued attribute composed of from $(1, 1)$ and to $(1, 1)$.

Components from and to should be defined according to the granularity of the original a attribute. Notice that constraints on values of from and to cannot be explicitly represented in the corresponding ERC+ schema and should be enforced separately. Such constraints include:

- the values of valid.from and valid.to for each attribute define disjoint intervals;
- for mandatory attributes, such as address and projects, the intervals specified for an attribute should be contiguous; and
- inclusion or covering constraints over complex temporal attributes composed of other temporal attributes, if any.

Figure 12 shows the transformation that allows to replace temporal entity and relationship types. In both cases, a complex multivalued attribute lifecycle is added with components status, whose domain is {active, suspended, dead}, and valid. The valid attribute represents the temporal element associated to the status; it is a complex multivalued attribute composed of from and to. As before, additional constraints should be enforced by the DBMS for constraining the values of attributes valid.from and valid.to of each entity and relationship type, as well as constraining the life cycle of the relationship instances with respect to the life cycle of the participating entities.

In the corresponding ERC+ schema, cardinalities (0,1) and (1,1) of the temporal relationship become (0,n) and (1,n), respectively. For example, in Figure 12 (b), Employee participates in relationship WorksOn with cardinality (1,n). Indeed the cardinality (1,1) in Figure 12 (a) means that, at any point in time, each employee works in exactly one department. However, since WorksOn is a temporal relationship, the database must keep track of all past, present, and future instances of the relationship. Therefore, in Figure 12 (b), an additional constraint is needed stating that relationship WorksOn satisfies the functional

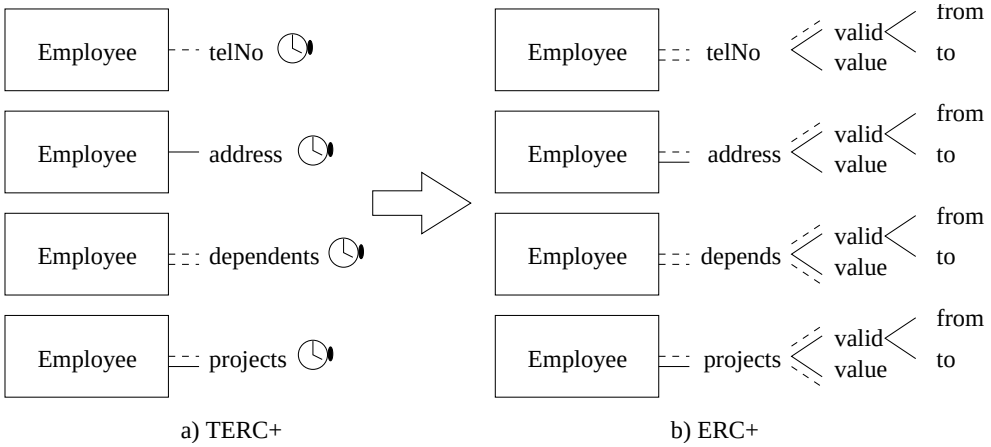

Figure 11: Semantics of temporal attributes.

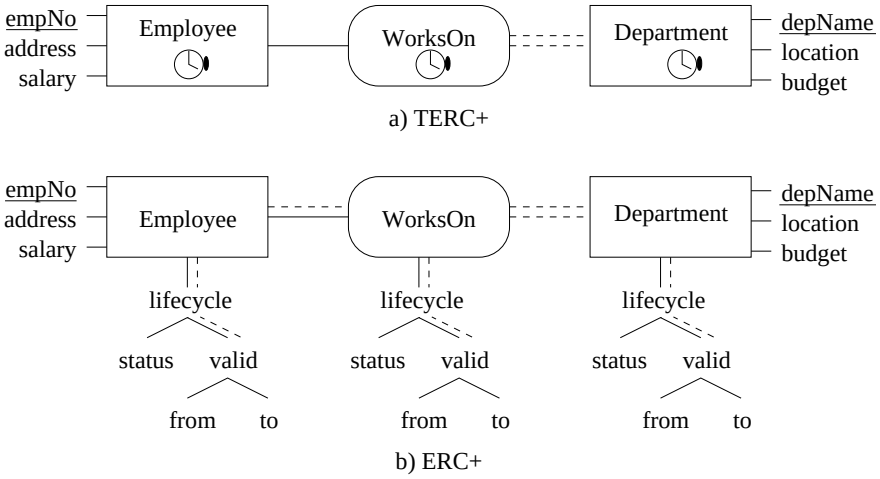

Figure 12: Semantics of temporal entities and relationships.

dependency (employee, valid.from) → department.

Dynamic relationships are translated into regular relationships, with a system attribute TYPE, whose value defines the type of the relationship: transition, generation, one of the predefined timing relationships (or the formula given by the designer to define his own timing relationship), or a time-based aggregation. Constraints implement the specifics of the relationship. For example, a *during* relationship generates a temporal integrity constraint specifying that the temporal element associated to the active status of the first entity is included in that of the second entity.

4 Related work

While the literature is overabundant on temporal databases, most of the work reported deals with implementation-oriented data models, namely the relational model and, more recently, object-oriented models. The discussion of issues and solutions is therefore most often biased by the specific characteristics of the model used. Work on relational tuple timestamping is, for example, inspired by the need to obey relational first normal form. At an other extreme, the proposal for a temporal object-oriented model in Chimera [3] is hardly transposable to an object-oriented system with different assumptions on the class and type concepts. This general tendency to deal with implementation contexts, staying away from users' needs, may explain the comment in [23] that "there seems to be a gap between the goals assumed by the temporal database community and the needs" of users' communities looking for temporal support. More focus on conceptual modeling should help fill the gap.

In the literature on temporal conceptual models, extensions of the entity-relationship model are predominant. Apart from a few isolated contributions [11, 17],

major research efforts have resulted in the ERT and TEER proposals. The ERT (ER Time) model [31] has been specified as the kernel for a larger ESPRIT project to develop a set of integrated tools including temporal facilities [32, 33]. While very significant in terms of system development, the data modeling part of the project relies on a rather weak version of the ER model which does not include direct support for complex data structures and only provides binary relationships. This results in modeling concepts and design practices which would be confusing for the inexperienced designer. To this extent, ERT does not match user orientation criteria required of a genuine conceptual model.

The second major proposal, the TEER (Temporal Extended ER) model [13] focused on query facilities rather than on data modeling. The outcome is a rather simple temporal model where, for example, time elements for entities are restricted to a single interval (i.e., no suspension, reactivation), and the validity period for a composite attribute is by definition the union of the validity periods of its component attributes. Although intuitively appealing, this assumption proves too restrictive both in terms of modeling needs and data manipulation (e.g., structural join operations generate results that are inconsistent with the modeling rules and it may not be possible to formulate cyclic queries). Also, TEER does not support dynamic relationships.

Keeping the life cycle of objects. i.e., the creation, suspension, resuming, and killing of objects, is also introduced in [8]. However, one disadvantage with respect to TERC+ is the absence of available information about the past of an object after it is destroyed.

There is very little on dynamic relationships in the literature. Previous work on transition and generation relationships may be found in [14, 15], where the motivation is not related to temporal databases. These proposals are somehow less general than those discussed in this paper. Time-based aggregation has also been proposed elsewhere, not for temporal databases but within the context of object-versioning facilities for object-oriented DBMSs. Finally, to the best of our knowledge, there has been no contribution suggesting that time-based relationships be explicitly describable as part of the normal schema design process.

We now comment two representative temporal object-oriented models.

In [24], a bitemporal object-oriented database model is described that supports both valid time and transaction time. Like our approach, it allows parts of a database schema to be temporal and other parts to be nontemporal. As in TERC+, temporality can be attached to a class (recording the life cycle of the entities) and to the attributes (recording the different attribute values) in an orthogonal way, e.g., temporal entities with nontemporal attributes, and vice versa. That work also considers the possibility of adding temporality to both complex attributes and their components. However, nontemporal subclasses of temporal classes are allowed, although, as seen in Section 2.5, this contradicts the substitutability principle.

The paper also discusses semantic issues about queries and updates and it defines an intensional interpretation of the database where each class and each attribute are treated as bitemporal. However, some basic postulates underlying this interpretation contradict our semantics of temporal features in TERC+.

A formal definition of a valid-time temporal OODB model is presented in [3]. A temporal type constructor allows to introduce temporality for any static type. A life cycle is associated to each class, representing the time interval during which the class has existed; it is assumed to be contiguous. Each class maintains the history of its direct instances and of all its instances over time. Each object has a life cycle associated to it, which, unlike TERC+, is assumed to be contiguous. Each temporal object keeps the history of the most specific class to which it belongs during its life cycle, while a nontemporal object only retains the most specific class to which it currently belongs.

Several invariants are defined on the model. For example, information about the historical extent of a class must be consistent with the class histories of the objects in the database. Also, the life cycle of an object can be partitioned in a set of intervals in which the object belongs to a most specific class. Consistency of an object states that, at each instant t of its life cycle, the object must contain a value for each temporal attribute of the class to which it belongs at time t, and this value must be of the correct type.

For generalization, it is required that there be, for each object and each instant, a unique most specific class it belongs to, and also that objects of subclasses be valid at that instant in the superclasses. Since nontemporal attributes can be refined to temporal attributes in subclasses, for ensuring substitutability the redefined attribute in the superclass is "coerced" to its value at the instant *now*.

In addition to the classical notions of equality by identity and by value, two further notions of value equality are defined. Two objects are instantaneously value equal if there exists an instant t in which their attributes have the same values. They are weakly value equal if their attributes have ever had the same values, maybe in different instants.

5 Conclusions

This paper presents a temporal conceptual model, TERC+, that can be used for developing applications with temporal features in a practical and realistic way. TERC+ is a conceptual model in that it allows to describe applications independently of implementation concerns, which is not the case for relational or object-oriented data models. TERC+ is an entity-relationship model extended with temporal features and it relies on a formal definition. TERC+ is an orthogonal language, and it is directly translatable into logical models of existing DBMS's.

The TERC+ language has been used in the design of a practical and realistic methodology for developing applications with temporal features. The methodology consists of the following steps:

(1) conceptual specification of an application with a temporal conceptual model;
(2) implementation of the conceptual schema, for a relational or object-oriented DBMS;
(3) optimization and tuning of the relational representation.

We are working on a prototype implementation in Java in which the user can draw the schema of the application and the system will generate the corresponding relational or object-oriented schema. We are also currently working on defining a visual data manipulation language for TERC+ to be incorporated into our system.

This methodology was applied in a broader context where we defined a conceptual model for spatiotemporal data, called MADS, which is used for geographical information systems (GIS) [22]. MADS offers an object-based modeling of data structures enriched with spatial features (a rich variety of geometries), explicit description of topological relationships (whose scope has been extended to apply to entities with complex geometries) and temporal specifications. MADS is supported by formal definitions, establishing a theoretical basis to build manipulation operations, and is being implemented as a visual interface independent from any underlying GIS.

References

[1] K. Al-Taha, R. Snodgrass, and M. Soo. Bibliography on spatiotemporal databases. *Int. Journal of Geographical Information Systems*, 8(1):195–203, Jan.–Feb. 1994.

[2] J. Allen. Maintaining knowledge about temporal intervals. *Comm. of the Assoc. for Computing Machinery*, 26(11):832–843, Nov. 1983.

[3] E. Bertino, E. Ferrari, and G. Guerrini. A formal temporal object-oriented data model. In P. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Proc. of the 5th Int. Conf. on Extending Database Technology, EDBT'96*, LNCS 1057, pages 342–356, Avignon, France, 1996. Springer-Verlag.

[4] M. Böhlen. Temporal database system implementations. *SIGMOD Record*, 24(4):53–60, Dec. 1995.

[5] A. Bolour, T. Anderson, L. Dekeyser, and H. Wong. The role of time in information processing: A survey. *SigArt Newsletter*, 80:28–48, Apr. 1982.

[6] P. Chen. The entity-relationship model : towards an unified view of data. *ACM Trans. on Database Systems*, 1(1):9–36, 1976.

[7] J. Chomicki. Temporal query languages: A survey. In D. Gabbay and H. Ohlbach, editors, *Proc. of the 1st Int. Conf. on Temporal Logic*, LNAI 827, pages 506–534. Springer-Verlag, July 1994.

[8] J. Clifford and A. Croker. Objects in time. *IEEE Data Engineering*, 7(4):189–196, Dec. 1988.

[9] J. Clifford, C. Dyreson, T. Isakowitz, C. Jensen, and R. Snodgrass. On the semantics of "Now". *ACM Trans. on Database Systems*, 22(2):171–214, June 1997.

[10] J. Clifford and A. Tuzhilin. *Recent Advances in Temporal Databases. Proc. of the Int. Workshop on Temporal Databases.* Springer-Verlag, Zurich, Switzerland, 1995.

[11] V. DeAntonellis, A. Degli, G. Mauri, and B. Zonta. Extending the entity-relationship approach to take into account historical aspects of systems. In P. Chen, editor, *Proc. of the Int. Conf. on the E-R Approach to Systems Analysis and Design.* North-Holland, 1979.

[12] Y. Dennebouy, C. Parent, S. S. Dennebouy, M. Andersson, A. Auddino, Y. Dupont, E. Fontana, M. Gentile, and S. Spaccapietra. SUPER: Visual interfaces for object + relationship data models. *Journal of Visual Languages and Computing*, 6(1):73–99, 1995.

[13] R. Elmasri, G. Wuu, and V. Kouramajian. A temporal model and query language for EER databases. In *Temporal databases. Theory, design, and implementation* [35], chapter 9, pages 212–229.

[14] R. Gupta and G. Hall. An abstraction mechanism for modeling generation. In F. Golshani, editor, *Proc. of the 8th Int. Conf. on Data Engineering, ICDE'92*, pages 650–658, Tempe, Arizona, Feb. 1992. IEEE Computer Society.

[15] G. Hall and R. Gupta. Modeling transition. In *Proc. of the 7th Int. Conf. on Data Engineering, ICDE'91*, pages 540–549, Kobe, Japan, Apr. 1991. IEEE Computer Society.

[16] N. Kline. An update of the temporal database bibliography. *SIGMOD Record*, 22(4):66–80, Dec. 1993.

[17] M. Klopprogge. TERM: An approach to include the time dimension in the entity-relationship model. In *Proc. of the 2nd Int. Conf. on the Entity Relationship Approach*, pages 477–512, Washington, DC, Oct. 1981.

[18] M. Kolp and A. Pirotte. An aggregation model and its C++ implementation. In *Proc. of the 4th Int. Conf. on Object-Oriented Information Systems*, Brisbane, Australia, 1997. To appear.

[19] P. Loucopoulos and R. Zicari, editors. *Conceptual modeling, databases, and CASE.* John Wiley & Sons, 1992.

[20] E. McKenzie. Bibliography: Temporal databases. *SIGMOD Record*, 15(4):40–52, Dec. 1986.

[21] G. Ozsoyoglu and R. Snodgrass. Temporal and real-time databases: A survey. *IEEE Trans. on Knowledge and Data Engineering*, 7(4):513–532, Aug. 1995.

[22] C. Parent, S. Spaccapietra, and E. Zimányi. Conceptual modeling for federated GIS over the web. In *Proc. of the Int. Symp. on Information Systems and Technologies for Network Society*, Fukuoka, Japan, 1997.

[23] N. Pissinou, R. Snodgrass, R. Elmasri, I. Mumick, M. Özsu, B. Pernici, A. Segev, and B. Theodoulidis. Towards an infrastructure for temporal databases: Report of an invitational arpa/nsf workshop. *SIGMOD Record*, 23(1):35–51, Mar. 1994.

[24] H. Riedel. On consistency in temporal object bases. In S. Conrad, H.-J. Klein, and K.-D. Schewe, editors, *Integrity in Databases - Proc. of the 6th Int. Workshop on Foundations of Models and Languages for Data and Objects*, pages 149–156. University of Magdeburg, Faculty of Computer Science, Preprint No. 4, 1996, 1996. http://wwwiti.cs.uni-magdeburg.de/ conrad/IDB96/Proceedings.html.

[25] F. Saltor, M. Castellanos, and M. García-Solaco. Suitability of data models as canonical models for federated databases. *SIGMOD Record*, 20(4):44–48, 1991.

[26] R. Snodgrass. Temporal object oriented databases: A critical comparison. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability, and Beyond*, chapter 19, pages 386–408. Addison-Wesley, 1995.

[27] R. Snodgrass, editor. *The TSQL2 Temporal Query Language.* Kluwer Academic Publishers, 1995.

[28] M. Soo. Bibliography on temporal databases. *SIGMOD Record*, 20(1):14–23, Mar. 1991.

[29] S. Spaccapietra and C. Parent. ERC+: an object based entity relationship approach. In Loucopoulos and Zicari [19], chapter 3, pages 69–86.

[30] R. Stam and R. Snodgrass. A bibliography on temporal databases. *Database Engineering*, 7(4):231–239, Dec. 1988.

[31] B. Theodoulidis, P. Loucopoulos, and B. Wangler. The entity relationship time model. In Loucopoulos and Zicari [19], chapter 4, pages 87–116.

[32] C. Theodoulidis and P. Loucopoulos. The time dimension in conceptual modelling. *Information Systems*, 16(3):273–300, 1991.

[33] C. Theodoulidis, P. Loucopoulos, and B. Wangler. A conceptual modelling formalism for temporal database applications. *Information Systems*, 16(4):401–416, 1991.

[34] V. Tsotras and A. Kumar. Temporal database bibliography update. *SIGMOD Record*, 25(1):41–51, Mar. 1996.

[35] A. Uz Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal databases. Theory, design, and implementation.* Benjamin/Cummings, 1993.