PROJECT REPORT

Temporal aspects in database modelling for medical knowledge production

Aida Kamišalić, David Riaño, John Bohada, Tatjana Welzer

Tarragona, June 2005

Table of Contents

Table of Figures

1 Introduction

Time-varying information is managed by a wide range of database applications such as accounting, banking, record-keeping, travel and medicine applications. Handling time concepts is crucial in medicine [1]. Medical assistance consists on diagnosis, treatment and prognosis processes. Diagnosis is the process by which a patient is assigned a primary disease that will direct the physician activities for cure. Treatment is about the definition of the most convenient plan to fight the patient disease, and prognosis is the ability to predict the consequences of a treatment in a particular case. Such processes are closely related and capturing temporal aspects is extremely important.
Existing temporal database applications employ the Entity-Relationship (ER) model for database modelling. The model is easy to understand and use, and it provides good overview of a database modelling. It has been recognized that although temporal aspects are prevalent and important for most applications, they are also difficult to capture using the ER model. When fully modelling the temporal aspects, they tend to obscure and clutter otherwise intuitive and easy-to-comprehend diagrams. The research community's response has been to develop temporally enhanced ER models. These temporal ER models are developed in an attempt to provide modelling constructs that more naturally and elegantly permit the designer to capture temporal aspects.
Basically, there exist two approaches of adding temporal support into the ER model. The temporal ER models generally either change the semantics of the existing ER model constructs or introduce new constructs to the model. The first approach is adopted in developing Extended ER model known as EER [14] and in Temporal EER model known as TempEER [18]. The second one is adopted in developing Temporal extension of ERC+, known as TERC+ [8], Time extended EER model known as TIMEER [9], Temporal ER model known as TER [2], etc. Each of these models supports some temporal aspects. We decided to use the TIMEER for modelling temporal aspects of our database for medical knowledge production. It is the most complex model of all known, giving the support for transaction and valid time, lifespan and user-defined time, and is giving support to all temporal aspects important for solving our particular problem.

The document gives an overview of existing temporal aspects and their implementation in TIMEER model considering their usage over specific constructs (entities, attributes and relationships). Section 2 is introducing database modelling constructs and time aspects. Preferences of using temporal ER models are presented in section 3. The following section is explaining different approaches for introducing temporal support to ER models. Specification of concepts in query language TSQL2 is presented in section 5. Section 6 is reserved for detailed explanation of TIMEER model, which was chosen for modelling our specific problem. Different database models are presented in section 7, considering different capturing of temporal aspects. The document ends with final conclusions and acknowledgements.

2 DB modelling constructs and time aspects

An **entity** represents an object of interest from the real world [8, 13]. The time during which an entity exists is called existence time of the entity. A set of entities with similar structure and behaviour is represented by **entity type**. The existence of an entity type may depend on the presence of another entity type. This type of entity type is referred to as a **weak entity type** [18]. An entity is characterized by its properties, modelled by **attributes** [9]. At any

given point in time, an entity has a value for each of its attributes. The values of some attribute remain unchanged over time while others vary, that is, at different points in time, the values of an attribute for an entity may be different. A **relationship** represents a link of interest between two or more entities [8]. A **relationship type** among some entity types defines a set of relations among entities of these types. Each relationship relates exactly one entity from each of the entity types that the relationship type is defined over [12].
Another type of relationships exists, namely the superclass/subclass relationships that classifies entities of a superclass into different subclasses, e.g. employees may be divided into secretaries, engineers and technicians. This type of relationship is known as **is-a relationship**, or **generalization** [8]. Is-a relationship represents inheritance hierarchies rather than relate entities. The entities of the subclasses inherit all the properties of entities of the superclass. It is not possible in subclasses to delete or modify the inherited properties, but it is possible to add new properties [9]. Additional constraints can be expressed on generalizations, such as:

- A generalization is said to be **exclusive** if an entity cannot belong to more than one subclass; otherwise is **overlapping**;
- A generalization is said to be **total** if every entity belongs to the superclass; otherwise it is **partial** [8].

An **aggregation link** is a special directed binary relationship whose semantics expresses that entity types, called composite entities, represent aggregates of another entity types, called component entities. Alternative terms for aggregation are composition link or part-of relationship. Additional constraints can be expressed on aggregations as follows:

- An entity type in an aggregation is said to be **dependent** on the other entity type if, when one entity of the later type stop to exist, all entities of the former type to which it is related through 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 [8].

A data model should make it possible to conveniently and concisely capture all the information that is meaningful and relevant for the application at hand. This implies that the database designers should have the ability to indicate, using the conceptual data model, whether or not they want to register these periods of time for the entities.

Several types of temporal aspects of information have been discussed over the years. We will focus on four distinct types of temporal aspects that are specific to and widely used within temporal databases, namely lifespan, transaction time, valid time and user-defined time [6].
The **lifespan** of an entity captures the existence time of the entity. If relationships are regarded as having existence in their own right, the concept of lifespan is also applicable to relationships, with the same meaning as for entities. Capturing lifespans of entities and relationships is important in many applications, because entities and relationships may exist beyond the times when their attributes have values associated with the entities or relationships.
The **transaction time** of database fact is the time when the fact is current in the database and may be retrieved [6]. With transaction time captured, past states of a database are retained, which is essential in applications with accountability or trace-ability requirements. It is impossible to change the past, so the transaction times cannot be changed. Transaction time may be associated with any element stored in a database, no only with facts. Thus, all database elements have a transaction time aspect [9].

The **valid time** of a fact is the time when the fact is true in the modelled reality [6]. Any fact in the database may be associated with a valid time. Because attributes are the modelling constructs used to capture facts at the conceptual level, a temporal ER model should support the possibility to register valid time for attributes. Support for valid time is important because it is fundamentally important in a large class of applications to know at what times the facts recorded in the database are true. However, the valid time may or may not be captured explicitly in the database. Three different cases arise in connection with the recording (or non recording) of the valid time of an attribute. First, if we record the valid time, this implies that we obtain the ability to capture all the values that have ever been valid for the attribute for a particular entity. Second, if we do not register the valid time of the attribute, this may be because the value of the attribute either never changes or because we are only interested in the current value of the attribute. Third, it could be that we do not know the valid time of the attribute – we know the valid value, but not the time when it is valid [5].
Valid and transaction time are general – rather than application specific – aspects of all database facts. Lifespan and transaction time are general aspect of entities [9]. All above mentioned temporal aspects have duration.
User-defined time is supported when time-valued attributes are available in the data model [16]. They are already available in the ER model and should also be supported in a temporally extended ER model. Valid and transaction times have special query language support [17], while user-defined time has not. It may be used for attributes such as "birth day", "hiring date", etc. Table 1 is giving an overview of database modelling construct with supported temporal aspects.

	Entity types	Attributes	Relationship types
Lifespan	+	-	+
Transaction time	+	+	+
Valid time	-	+	+
User-defined time	-	+	-

Table 1: DB modelling constructs and supported aspects of time

Different time data types may be used for capturing the temporal aspects of database modeling constructs, such as instants, chronons, intervals and temporal elements.
An **instant** is a time point on an underlying time axis [6]. Various models of time have been proposed in different time literature. The time can be viewed as discrete, dense or continuous. The instants in a discrete model of time are isomorphic to the natural numbers: every instant has a unique successor. Instants in the dense model of time are isomorphic to the rational numbers: between any to instants there is always another. Continuous models of time are isomorphic to the real numbers [6].
In a data model that supports a time line using chronons, an instant is represented by a chronon. A single **chronon** may therefore represent multiple instants. In a data model, a one-dimensional chronon is a non-decomposable time interval of some fixed, minimal duration. An n-dimensional chronon is a non-decomposable region in n-dimensional time. Important special types of chronons include transaction time, valid time and bitemporal chronons.
A time **interval** is the time between two instants. In a system that supports a time line composed of chronons, an interval may be represented by a set of contiguous chronons.
The **temporal element** is a finite union of n-dimensional time intervals. Special cases of temporal elements include valid time elements, transaction time elements and bitemporal elements. They are finite unions of valid time intervals, transaction time intervals and bitemporal intervals, respectively. A **bitemporal interval** is a region in two-space of

transaction time and valid time. When a fact has bitemporal interval, it means that the fact was true during the specified interval of valid time and was logically in the database during the specified interval of transaction time [6]. Temporal elements are often used as timestamps.

3 Usability of temporal ER models

The different temporal aspects may or may not be captured in the database, depending on the application requirements. Therefore, the support for these aspects should be **user-specifiable** and maximally **flexible**. This is achieved if the temporal ER model permits the database designer to decide which temporal aspects to capture of the different database elements. For example, the designer may capture lifespan for the entity type while capturing both transaction time and valid time for some of the attributes of the entity type.

Different **time data types** may be used for capturing the temporal aspects of database constructs, including instants, time intervals, chronons and temporal elements [4, 11]. A temporal ER model may provide the database designer with a choice of data types, thereby increasing the utility of the model. All above mentioned time data types may be used for encoding durations. The importance of the availability of time data types is dependent on whether the model is used solely as a design model or is also used as an implementation model, complete with database instances and a query language.

It is possible to capture temporal variability of different objects using times of different **granularities** [3, 4, 11]. The timestamp granularity is the size of each chronon in a timestamp interpretation. For example, if the timestamp granularity is one second, then the duration of each chronon in the timestamp interpretation is one second [6]. It should be possible to capture the variability of the different objects in the database using different granularities. For example, the granularity of a minute may be used when recording the actual working hours of employees, while the granularity of a day may be used when recording the assignment of employees to projects.

The temporal variability of different objects may be known with different **precisions** and although some imprecision may be captured using multiple granularities; it does not provide a general solution. For example, the variability of an attribute may be recorded using timestamps with granularity of a second, but the varying values may only be known to the precision of ±5 seconds of recorded time. This phenomenon may be prevalent and important to capture in scientific and monitoring applications that store measurements made by instruments. Thus the usability of a temporal ER model would be increased if support for temporal precision is provided [10].

To increase the usability of a new ER model, it is very important that legacy ER models remain correct in the new model. This property is known as **upward compatibility**. A temporal ER model is upward compatible with respect to a conventional ER model if any legacy conventional ER diagram is also legal ER diagram in the temporal model and if the semantics of the diagrams in the two models are the same. Upward compatibility protects investments in legacy systems and provides the basis for a smooth transition from conventional ER model to a temporally enhanced ER model [10].

4 Temporal support

The approaches taken to add temporal support into ER model are quite different. The temporal ER models generally either change the semantics of the existing ER model constructs or introduce new constructs to the model. The first one is to devise new notational shorthands that replace some of the patterns that occur frequently in ER models when temporal aspects are being modelled. The second one is to change the semantics of the existing ER model constructs, making them temporal. This approach does not result in any new syntactical constructs. All the original constructs have become temporal.

The approach where all existing ER model constructs are given temporal semantics has been used in some of the existing temporal models [14, 18]. Database designers are likely to be familiar with the existing ER constructs. After understanding the principle of making these constructs temporal, the designers are ready to work with and benefit from using the temporal ER model. However, this approach is not totally without problems. For example, it is not possible to design databases with non-temporal parts. Another problem is that old diagrams are no longer correct, while their syntax is legal, their semantics have changed, and they therefore no longer describe the underlying relational database schemas.

It is also possible to retain the existing ER constructs with their usual semantics while achieving temporal support. This is accomplished by adding new temporal constructs to the model that provide the support. It is used in most of the existing extensions of ER models [2, 9, 10]. The extent of the changes made to the ER model may range from minor changes to a total redefinition of the model.

Two types of new temporal constructs may be distinguished. With implicit temporal support, the timestamp attributes used for capturing a temporal aspect are ‘hidden’ in the new modelling constructs. In contrast, with explicit temporal support, timestamp attributes are explicit and the semantics of the existing ER constructs are retained. Any new modelling constructs are notational shorthands for elements of regular ER models, introduced to make the modelling of temporal aspects more convenient. The models that retain the existing constructs with their old semantics and introduce new temporal constructs also have problems. If their extensions are comprehensive, they are likely to be more difficult for the database designers to learn and understand. The larger initial investment in training that this induces may prevent a model from being accepted in industry. On the other hand, this approach avoid the problem of legacy diagrams not describing the underlying database, since the semantics of the existing ER constructs are retained.

There also exist two approaches considering temporal ER model supporting query language [2,15] or providing algorithms that map ER models to implementation platforms [2, 9, 14, 18]. A mapping algorithm translates a temporal ER diagram into a corresponding database schema in another data model. The most obvious possible target data models are the relational model, some temporal relational model and the conventional ER model. The algorithm may map temporal ER models directly to relational database schemas or a two-phase approach may be adopted, where temporal ER models are first mapped to conventional ER diagrams and then mapped to relational database schemas, reusing mappings from the conventional ER model to the relational model. For minor extensions of the ER model, the reuse in the two-phase approach may be attractive. However, the two-phase translation yields less control over what relational schemas result from the combined mapping. With this

approach, which is the one assumed in most existing temporal ER models, the ER model is a design model that is used solely for database modelling and has no directly associated database instance. In industry, the ER model is generally used as a design model, with variations of the relational model being the most popular target models.
As an alternative to mapping ER diagrams to the schema of a separate implementation platform, another approach is to assume a system that implements the ER model directly. With this approach, a mapping to an implementation platform is not required. Instead, a query language should be available for querying ER databases [10].

There exist many evaluation criteria that indicate that none model is entirely satisfactory. For example, only two models support the transaction time aspect of data, which is important to many applications. Rather than being systematically founded on an analysis of general concepts and temporal aspects, the designs of the models are often ad hoc. For example, the design of one model is the result of the need for the modelling of temporal aspects in a specific application. We have decided to use TIMEER model for database modelling for medical knowledge production and it will be detail explained in section 6. The design of the model was based on an ontology, which defines database objects, fundamental aspects of time and indicates which aspects of time may be associated meaningfully with which database object. It was also provided a formal semantics based on denotational semantics.
Research community has provided temporal query language known as TSQL2. It has been standardised and is an extension of commonly used SQL-92. The following section is making an overview of the concepts introduced to TSQL2.

5 TSQL2

TSQL2 is a fully upwardly compatible extension of SQL-92. The functionality of user-defined time support in SQL-92 is enhanced. Support for transaction and valid time has been added. The TSQL2 model of time is bounded on both ends. The model refrains from deciding whether time is ultimately continuous, dense, or discrete. TSQL2 does not allow the user to ask a question that will differentiate the alternatives. Instead, the model accommodates all three alternatives by assuming that an instant on a time-line is much smaller than a chronon, which is the smallest entity that a timestamp can represent exactly (the size of a chronon is implementation-dependent). An instant can only be approximately represented. A discrete image of the represented times emerges at run-time as timestamps are scaled to user-specified (or default) granularities and as operations on those timestamps are performed to the given scale. An instant is modelled by a timestamp coupled with an associated scale (e.g., day, year, month). A period is modelled by the composition of two instant timestamps and the constraint that the instant timestamp that starts the period equals or precedes (in the given scale) the instant timestamp that terminates the period.

TSQL2 includes the concept of a baseline clock, which provides the semantics of timestamps. The baseline clock relates each second to physical phenomena. The baseline clock partitions the time line into a set of contiguous periods. Each period runs on a different clock. A synchronization point delimits a period boundary. The baseline clock and its representation are independent of any calendar [17].
In SQL-92 date, time and interval data types are augmented with a period data time, of specifiable range and precision. The range and precision can be expressed as an integer (e.g., a precision of 3 fractional digits) or as an interval (e.g., a precision of a week). Operators are available to compare timestamps and to compute new timestamps, with a user-specified precision. Temporal values can be input and output in user-specifiable formats, in a variety of

natural languages. Calendars and calendric systems permit the application-dependent semantics of time to be incorporated.
A surrogate data is introduced in TSQL2. Surrogates are unique identifiers that can be compared for equality, but the values of which cannot be seen by the users. In this sense, a surrogate does not describe a property (i.e., it has no observable value). Surrogates are useful in identifying objects associated with time-varying attributes, but are not a replacement for keys.

Three time-lines are supported in TSQL2: transaction time, valid time and user-defined time. Transaction-time is bounded by initiation, the time when the database was created, and until changed. In addition, valid time and user-defined have two special values, beginning and forever, where are the least and greatest values in the ordering. Transaction time has the special value until changed. Valid and user-defined data types can be temporally indeterminate. In temporal indeterminacy, it is known that an event stored in a temporal database did in fact occur, but it is not known exactly when that event occurred. An instant (interval, period) can be specified as determinate or indeterminate.

The snapshot tables currently supported by SQL-92 continue to be available in TSQL2.
TSQL2 also allows state tables to be specified. In such tables, each tuple is timestamped with a temporal element, which is a union of periods. The timestamp is implicitly associated with each tuple; it is not another column in the table. The range, precision and indeterminacy of the timestamps within the temporal element can be specified. TSQL2 also allows event tables to be specified. In such tables, each tuple is timestamped with an instant set.
Transaction time can be associated with tables. The transaction time of a tuple, which is a temporal element, specifies when that tuple was considered to be logically stored in the database. The transaction timestamps have an implementation-dependent range and precision, and are determinate.
In summary, there are six kinds of tables: snapshot (no temporal support beyond user-defined time), valid-time state tables (consisting of sets of tuples timestamped with valid-time elements), valid-time event tables (timestamped with valid-time instant sets), transaction-time tables (timestamped with transaction-time elements), bitemporal state tables (timestamped with bitemporal elements), and bitemporal event tables (timestamped with bitemporal instant sets).

The CREATE TABLE and ALTER statements were extended to allow specification of the transaction- and valid-time aspects of temporal tables. The scale and precision of the valid timestamps can also be specified and later altered.
The FROM clause in TSQL2 allows tables to be restructured so that the temporal elements associated with tuples with identical values on a subset of the columns are coalesced.
Restructuring can also involve partitioning of the temporal element or instant set into its constituent maximal periods or instants, respectively. Many queries refer to a continuous property, in which maximal periods are relevant.
The valid-time timestamp of a table may participate in predicates in the WHERE clause by via VALID() applied to the table (or correlation variable) name. The transaction-time of a table can be accessed via TRANSACTION(). The operators have been extended to take temporal elements and instant sets as arguments.
Conventional snapshot tables, as well as valid-time tables, can be derived from underlying snapshot or valid-time tables. An optional VALID or VALID INTERSECT clause is used to specify the timestamp of the derived tuple. The transaction time of an appended or modified tuple is supplied by the DBMS [17].

All aspects of TSQL2 are extensions of SQL-92. The user-defined time in TSQL2 is a consistent replacement for that of SQL-92. This was done to permit support of multiple calendars and literal representations. Legacy applications can be supported through a default SQL92 calendric system. The defaults for the new clauses used to support temporal tables were designed to satisfy snapshot reducibility, thereby ensuring that these extensions constitute a strict superset of SQL-92.

6 TIMEER

The Time Extended ER model, known as TIMEER, is presented in this chapter. The ER model has been extended in order to capture superclass/subclass relationships and complex entity types and is known as the EER model. EER model presented by Elmasri and Navathe [15] was chosen as the basic model of TIMEER [10]. There were introduced new temporal constructs and provided implicit temporal support for the model. It means that the TIMEER is upward compatible with the ER model it extends. It means that existing legacy ER diagrams are valid temporal ER diagrams and retain their legacy meaning in the new model. It also retains the ability to design non-temporal databases as well as databases where some parts are non-temporal and others are temporal. TIMEER supports the time data types 'instant' and 'temporal element'.
The TIMEER model has implicit temporal support, and the existing EER constructs and their semantics are retained. It means that new notation with implicit temporal support is added to the EER model to reach the TIMEER model. TIMEER extends the EER model to include temporal support for entities, attributes, relationships and superclasses/subclasses.

6.1 TIMEER modelling constructs

A regular **entity type** is represented by a rectangle. Since all entities represented by an entity type have existence time, modelled by lifespans in the database, and a transaction time aspect, the TIMEER model offers support for lifespans and transaction time for entity types. For each entity type in a TIMEER diagram, the database designer must decide whether or not to capture these temporal aspects of the entities in the database.
If the lifespan or the transaction time of an entity type is to be captured, this is indicated by placing an LS (Lifespan) or a TT (Transaction time) in the upper right corner of the rectangle, respectively. If both lifespan and the transaction time are captured, an LT (Lifespan and Transaction time) is placed as presented in Figure 1. Entity types that capture at least one temporal aspect are known as temporal entity types; otherwise, they are non-temporal.

Figure 1: Temporal entity type capturing lifespan and transaction time

The participation constraint of an entity type E with respect to a relationship type R is represented by placing *min* and *max* values in parentheses by the line connecting entity type E with relationship type R (as presented in Figure 2). If $min = 0$ then the participation of the entities of E is optional; if $min \geq 1$ then the participation is total (mandatory). If $max = 1$, this means that the entities of E cannot participate in more than one relationship at a time, whereas a $max = n$, with $n > 1$ means that E entities can participate in n relationships at a

time. At any point in time, each instance e of the entity type E will participate in at least *min* and at most *max* instances r of R.

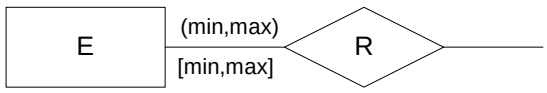

Figure 2: Representation of participation constraints in TIMEER

It is also useful to be able to describe the participation of an entity in a relationship over the entire existence time of the entity. In the TIMEER it is known as lifespan participation constraint. The lifespan participation constraint of entity type E with respect to relationship type R is represented by placing *min* and *max* values in square brackets by the line connecting entity type E with relationship type R (as presented in Figure 2). It means that over all of time, any instance e of the entity type E must participate in at least *min* and at most *max* instances r of R.

Weak entity types are represented by double rectangles and are used to represent entities that their existence depend on specific entities of another entity type and that cannot by themselves be lexically uniquely identified. A weak entity type must therefore be related via an identifying relationship type (represented by a double diamond) to at least one regular entity type that is then the owner of the weak entity type.

Entities are characterized by their **attributes**. A single-valued attribute is represented by an oval (Figure 3); a multi-valued attribute is represented by a double oval, and a composite attribute is represented by an oval connected directly to other ovals representing the component attributes of the composite attribute.
All facts, modelled by attributes, have a transaction time and a valid time aspect. The TIMEER model offers support for transaction time and valid time for all attribute types. If the database designer decides to capture the transaction time of an attribute, a TT is placed to the right in the oval; if valid time is captured, a VT is placed as before. If both the transaction time and the valid time are captured, a BT (Bitemporal) is used (presented in Figure 3). The components of a temporal composite attribute inherit the temporal specification for the composite attribute. Temporal support cannot be added separately to the components of a composite attribute. If no temporal aspects of an attribute are captured, it is the non-temporal attribute; otherwise, it is temporal.
To indicate that a set of attributes represent the key of an entity type, the attribute names of the involved attributes are underlined. Key attributes of an entity type can be specified as temporal or non-temporal. Simple and composite attributes may be specified as key attributes.

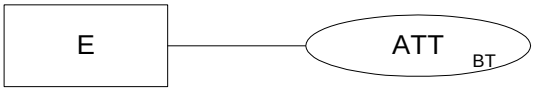

Figure 3: Temporal single-valued attribute capturing transaction and valid time

A **relationship type** is represented by a diamond. The model offers support for lifespans, transaction and valid time for relationship types. The reason for offering support for both lifespans and valid time is that relationships can be perceived as either attributes of the participating entities, or as things that exist in their own right. For each relationship type, it has to be decided by the database designer whether or not to capture the temporal aspects for the relationship type. If some temporal aspect is captured for a relationship type, it is

temporal; otherwise, it is non-temporal. If the relationship is perceived as an attribute, the possible temporal aspects are as for attributes, and the indication is placed in the lower corner of the diamond (presented in Figure 4). When relationships are perceived as things that exist in their own right, the temporal aspects supported are lifespans and transaction time.

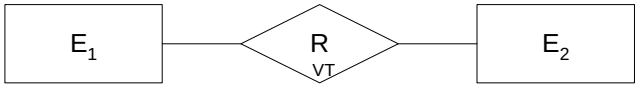

Figure 4: Temporal relationship type capturing valid time

TIMEER offers support for specifying **superclass/subclass relationships**. The syntax is as in the EER model. All subclasses inherit the attributes of their superclasses, and just as inherited attributes cannot be given new data types, it is not possible to change the temporal support given in the superclasses to the inherited attributes. But it is possible to add temporal and non-temporal attributes in the subclasses.

6.2 Usability of TIMEER model

It is provided support for capturing lifespans and transaction time for entities and relationships. This is achieved by making it possible for the database designer to specify for every entity type and relationship type whether or not to capture the temporal aspects of the modelling constructs. Similarly, support for capturing transaction time and valid time for attributes and relationships is provided. Finally, user-defined time attributes are available.

Because the database designer is able for each modelling construct to specify whether or not to capture each meaningful temporal aspect of the construct, TIMEER provides maximally meaningful and flexible temporal support. The model has optional use of the temporal constructs, providing the database designer with the possibility of mixing temporal and non-temporal constructs in the same diagram.
TIMEER supports time data types for the modelling instantaneous events and phenomena that persist in time, namely the 'instant' and 'temporal element' types. The model does not support 'interval' data type.
The time granularities supported by TIMEER are second, minute, hour, day, week, month, and year. The model does not support temporal imprecision.
TIMEER is upward compatible with respect to the EER model [14] because it is extended with new temporal constructs while retaining all original EER constructs, with their original syntax and semantics.

6.3 Transformation TIMEER → EER

We present all possible transformations from TIMEER to EER model considering all database modelling constructs supported by above mentioned temporal aspects.
Transformation of entity types capturing lifespan is presented in Figure 5; capturing transaction time is presented in Figure 6 and capturing both, lifespan and transaction time, is presented in Figure 7.

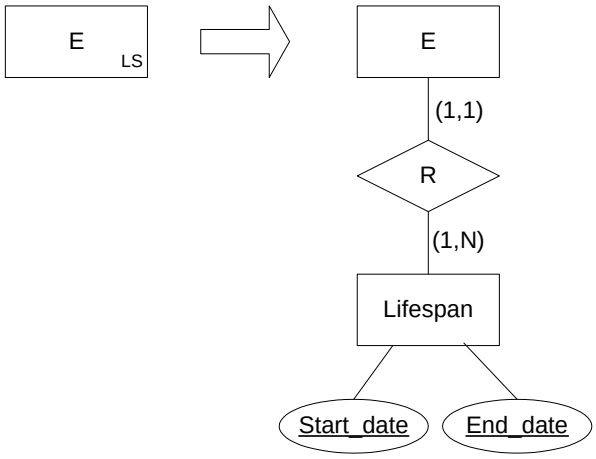

Figure 5: Transformation of entity type (capturing LS) from TIMEER to EER

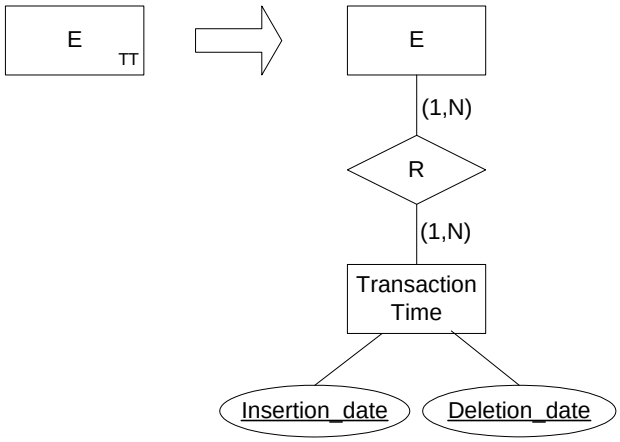

Figure 6: Transformation of entity type (capturing TT) from TIMEER to EER

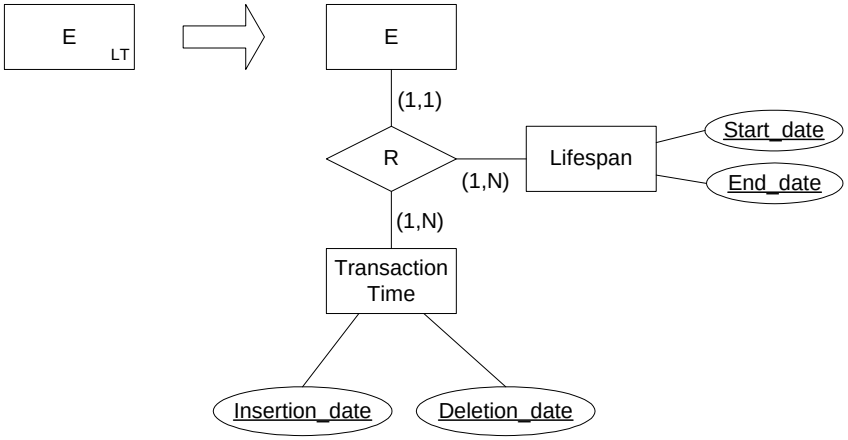

Figure 7: Transformation of entity type (capturing LT) from TIMEER to EER

Transformation of attributes capturing transaction time is presented in Figure 8; capturing valid time is presented in Figure 9 and capturing both, transaction and valid time, is presented in Figure 10.

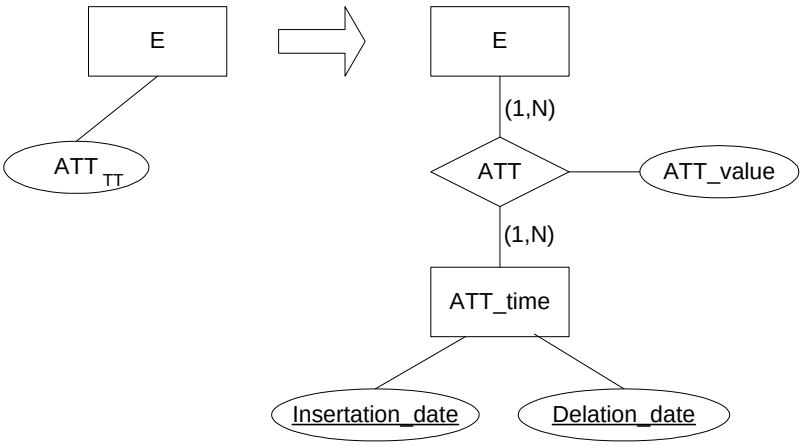

Figure 8: Transformation of attribute (capturing TT) from TIMEER to EER

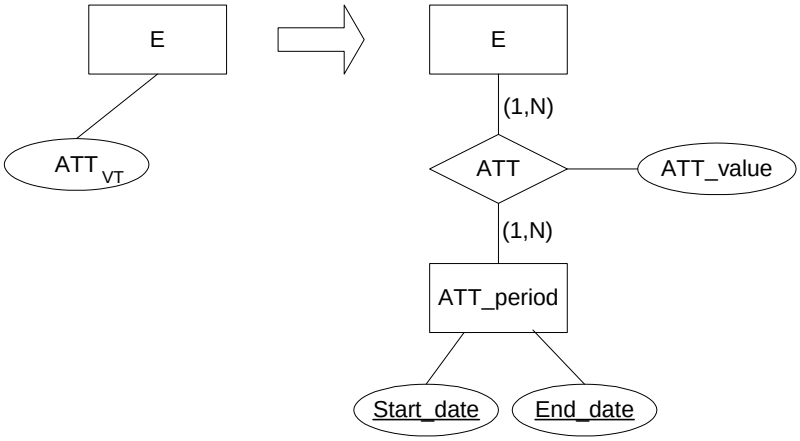

Figure 9: Transformation of attribute (capturing VT) from TIMEER to EER

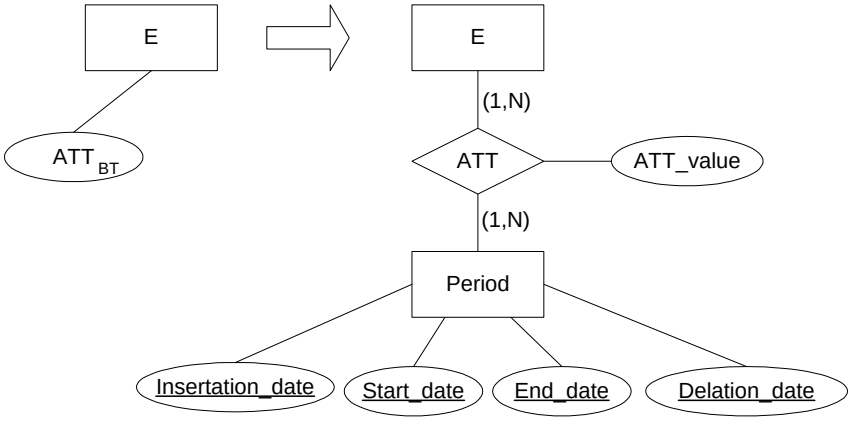

Figure 10: Transformation of attribute (capturing BT) from TIMEER to EER

Transformation of relationship types capturing lifespan is presented in Figure 11; capturing transaction time is presented in Figure 12; capturing valid time is presented in Figure 13; capturing lifespan and transaction time is presented in Figure 14, and capturing transaction and valid time, is presented in Figure 15.

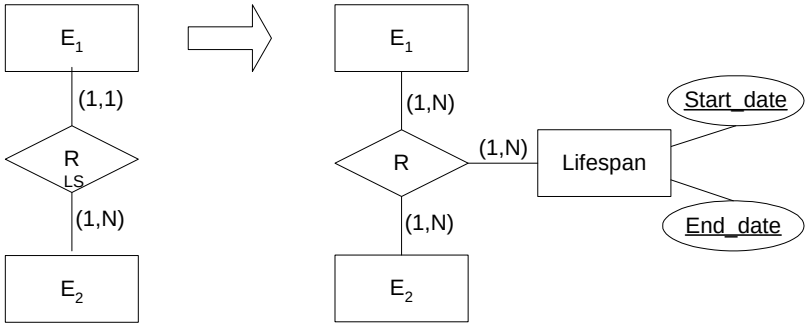

Figure 11: Transformation of relationship type (capturing LS) from TIMEER to EER

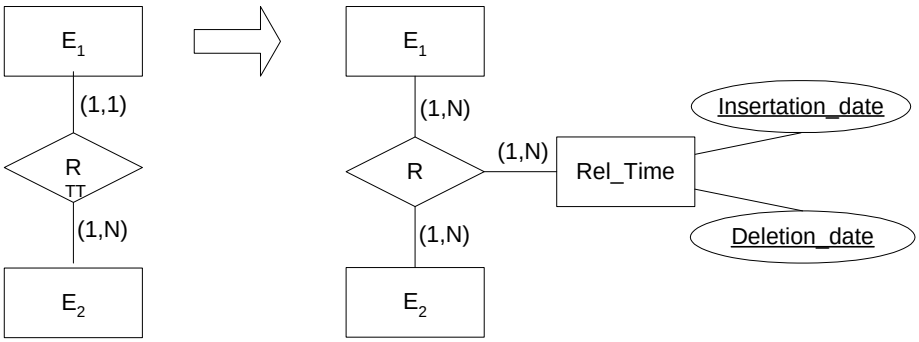

Figure 12: Transformation of relationship type (capturing TT) from TIMEER to EER

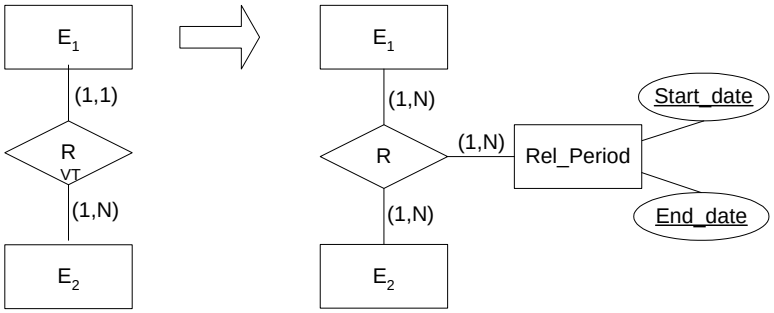

Figure 13: Transformation of relationship type (capturing VT) from TIMEER to EER

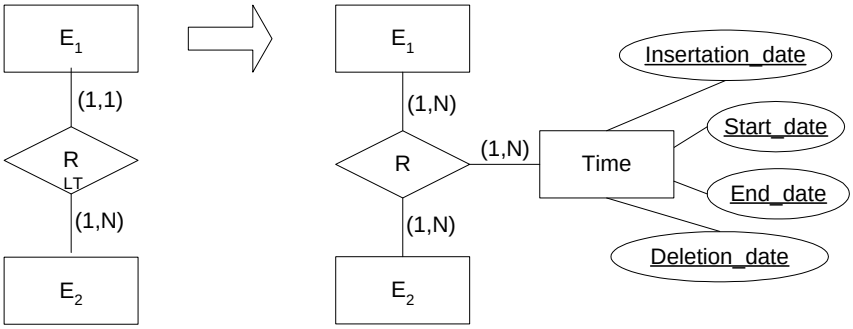

Figure 14: Transformation of relationship type (capturing LT) from TIMEER to EER

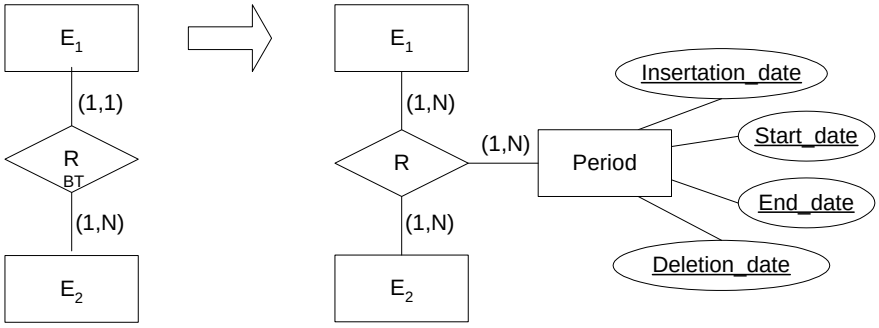

Figure 15: Transformation of relationship type (capturing BT) from TIMEER to EER

7 DB models for medical knowledge production

Processes of medical assistance are closely related. When a patient arrives to a hospital the symptoms and the signs, together with the patient antecedents (electronic healthcare record), help the physician to fix one (or several) possible diagnostics. Then, the data of the diagnosed disease (background knowledge) and the rest of the patient data are used by the physician to construct one o several possible medical treatments. These treatments are temporal plans that contain drugs prescriptions and medical procedures to be applied. Finally the physician can use the treatment and the patient state medical description to make a prognosis.

Based on the minimum basic data set (MBDS) concept [7], the diagnosis process gathers data about the *patient,* the *state,* the *primarydiagnosis,* and the *secondarydiagnoses.* Where, *Patient* represents demographic data about the patient as *name, surname, birth date,* and *sex*; *state* (S_i) is a list of pairs *(sign&symptom, value);* each *Sign&symptom* is *(ICD9-SS, description)* and *value* is the value that the sign or symptom takes (e.g. 38.5 ºC for temperature). Finally, *primarydiagnosis* represents the patient main disease and s*econdarydiagnosis* the list of secondary diseases of the patient. A *primarydiagnosis* or a *secondarydiagnosis* is a tuple *(ICD9-disease, name, description)*, where *ICD9-disease* stands for the ICD9 code [10] of the disease, *name* is the name of the disease, and *description* is free text optionally describing the disease.

The treatment of a patient is preceded by a diagnosis chapter in which the patient disease is identified. The diagnosis of a patient is conditioned by all the medical acts of the treatment. Each stage of a treatment is called a clinical *episode.* Then a *treatment* is a sequence of clinical episodes (T_i) each one containing *(patient, Primarydiagnosis, State, MedicalOrder, Duration).* Where, *Patient* represents demographic data of patient as *name, surname, age, sex. Primarydiagnosis* contains *(ICD9-disease, name, description), State* (S_i) is a list of *(sign&symptom, value), Sign&symptom contains (ICD9-SS, description)* and *Value* is the value that the sign or symptom takes. *MedicalOrder* contains *(prescription and/or test), Prescription* contains *(drugID, dose, frequency), Test* contains *(ICD9-procedure, description)* and *Duration* (t_i) is the time of a clinical episode (T_i).

Figure 16 shows the temporal representation of the clinical episodes in a patient treatment. In Figure 16(a) each clinical episode is represented by T_i, which indicates the patient evolution from state S_i to S_{i+1} (i=1..n-1). Each T_i is associated a time t_i that represents the duration of this clinical episode. The state S_n indicates the end of a particular treatment. Figure 16(b) shows a feasible case where a patient remains in the same state (S_1) for several consecutive episodes of the same treatment. In this case, if the clinical episodes T_i and T_{i+1} are equal (i.e. the same treatment is provided), it is considered as a single episode whose duration is t_i+t_{i+1},

but if something in the treatment has been changed after the appointment, this is interpreted as a change of the physician's mind after observing that the previous treatment step caused the patient to remain in the same state. In this case, T_i is equivalent to a fruitless attempt and it is removed from the global treatment. The duration of the remaining clinical episode is then t_{i+1}.

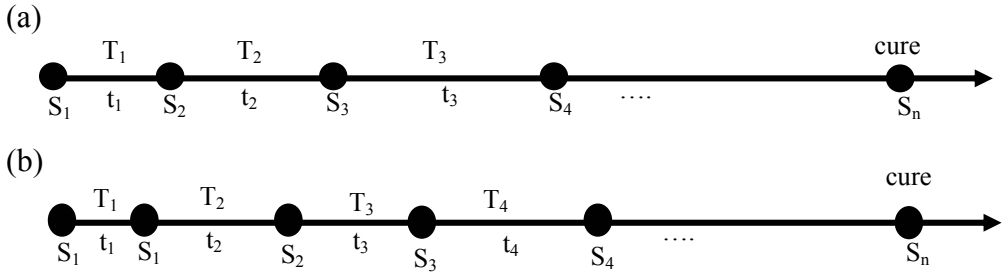

Figure 16: Temporal representation of the episodes in the treatment of patients

Finally, prognosis is the statistical analysis about the expected success of a concrete treatment when it is applied to a particular patient. In this case, prognosis can be interpreted as a graph where nodes represent patient states and edges represent the evolution of the patients from one state to another (i.e. patient improvement or worsening).

7.1 ER model description

The objective of this work is to propose a temporal data model to capture the data generated in health care process and use it to generate new knowledge from medical experiences. Considering the objective we have decided to support diagnosis and treatment processes with our database. Also we have decided to capture valid and user-define times. It would be also possible to introduce transaction time and by this support administrative level of a system. But considering our work objectives, we have decided to capture only valid time. In our case it is important to have knowledge of when some fact is valid and at this level it is not necessary to keep track of insertion or deletion time of the fact.

Representing medical processes DISEASE and PATIENT are the two main entities used for diagnosis and treatment. They store the main data about each patient and about all the ICD9 codified diseases [10]. The important aspect of time is the moment in which the data about symptoms are produced and the moment in which diagnosis of disease for particular patient is made. These are the times which are obligatory supported in our database (presented in Figure 17).
Important aspect of time is also when the treatment starts. During the treatment, the physician can propose some tests (e.g. blood analysis, X ray, etc.) or give some prescriptions. At least one of these (test or prescription) has to exist and therefore also the time aspect of both. Drug prescription consists of prescription period, doses, frequency and duration. Here we can find three types of temporal data. Prescription period and duration are attributes that we consider capturing with valid time. Frequency can be introduced as user-defined time.
If we want to keep record of new appointments assigned to each patient, we have to introduce another valid time. In Figure 17 is presented our model which allows also keeping records of new appointments. In order to compare, the same model is presented also in EER notation in Figure 19. Figure 18 is presenting a model which excludes the above mentioned possibility and only obligatory temporal aspects are kept.

Besides two above possibilities (to keep record of all the temporal aspects or just of obligatory ones) we have introduced the third possibility, which is managing temporal aspects using only user-defined time. It is supported in each ER model and for this there is no need of introducing temporal extension of ER model. In Figure 20 we present our model using only user defined times.

Finally, the prognosis process is described as the probability a patient can change from one current state to another (the one prognosed). Meanwhile diagnosis and treatment processes are medical activities that require capturing information, prognosis is a medical process that uses the experience of past cases stored in the database to predict about the patient under study. Storing the predictions of the physicians before they are confirmed (prospective approach) can be of introducing wrong information in the data base, if the prediction fails. For this reason, we propose a retrospective-based prognostic model in which the current states of the patients are considered the correct predictions of those patients in the past. This way, the database will only contain confirmed true data.

7.2 A scenario for using the model

The introduced database model can be useful for complex medical cares. So, when a patient arrives to a hospital, a physician looks for signs and symptoms to identify the patient's primary and secondary diseases. These diseases can be easily identified by the expert physician, but in some particular cases as doctors facing a rare disease or a combination of diseases or a disease not in the doctor's specialty or novel doctors may require a confirmation before concluding about a diagnosis. In these cases, the physician can use the database to ask for a suggestion that may confirm his/her suspicions or discard them or propose alternatives.

Once the diagnostic or diagnoses are confirmed, the patient has to start the treatment that the physician proposes. Every time the patient has an appointment, the physician has a chance to observe the signs and symptoms of the patient and make a decision that can be: continue with the same treatment, change medication, or order some procedure or test. All these decisions are stored in the database for later exploitation. Past episodes of the patient can be retrieved if they are needed, but in the case that a treatment or a set of treatments do not improve the state of the patient, a doctor can have doubts about how to proceed in a different way. Since the database gathers the information about the treatments given by all the physicians in the hospital, any practitioner can profit from the experiences or the different approaches to medicine of other colleagues.

Finally, the database is designed to permit the physicians to make adherence studies. These studies are useful to emphasize the differences between the treatments s/he is given and the treatments stored in the data base. So, s/he can generate medical knowledge showing how a particular sort of patient or disease is being treated in the hospital, compare these general guidelines with his/her approach and adapt.

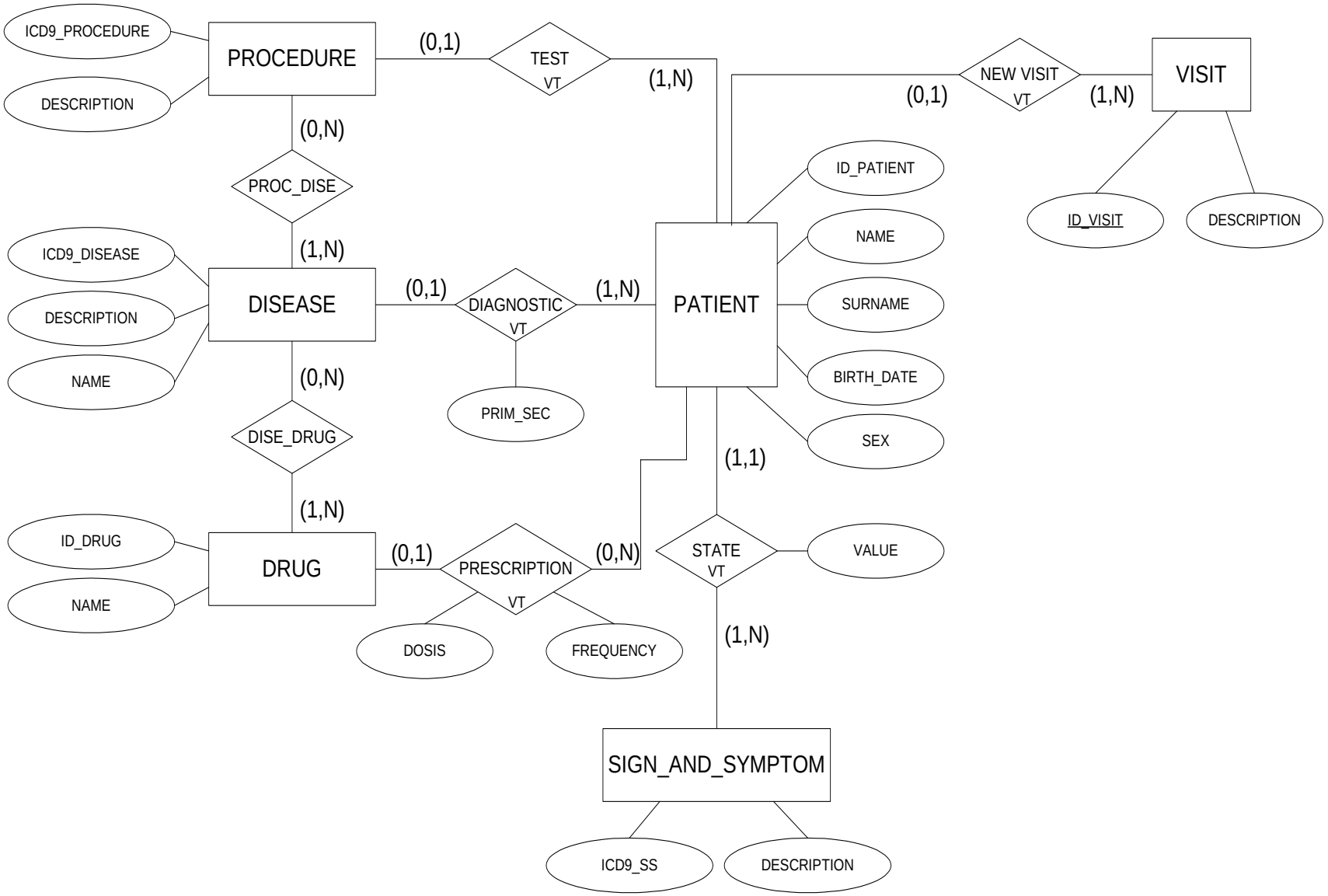

Figure 17: TIMEER model for medical knowledge production including all temporal aspects

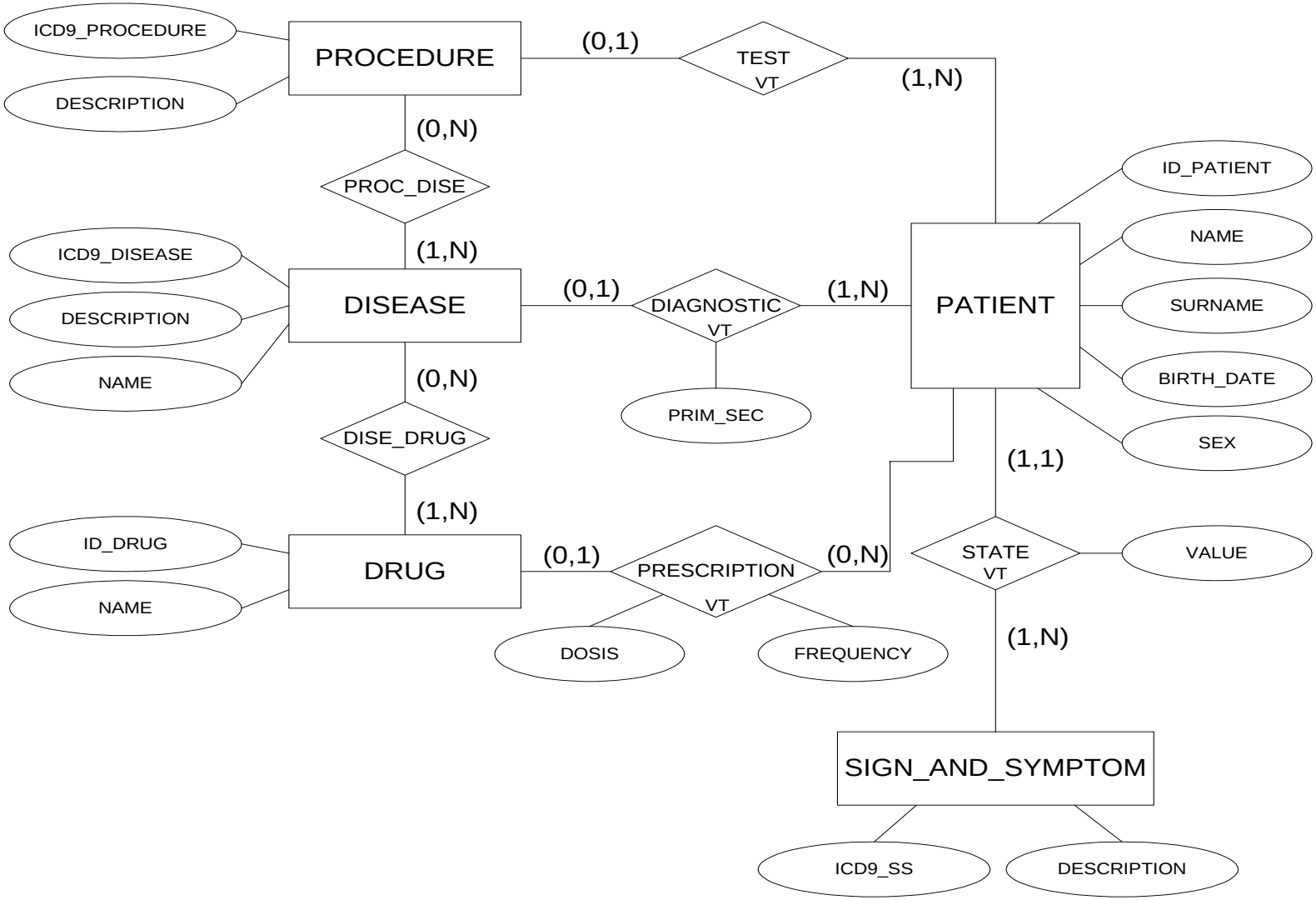

Figure 18: TIMEER model for medical knowledge production including obligatory temporal aspects

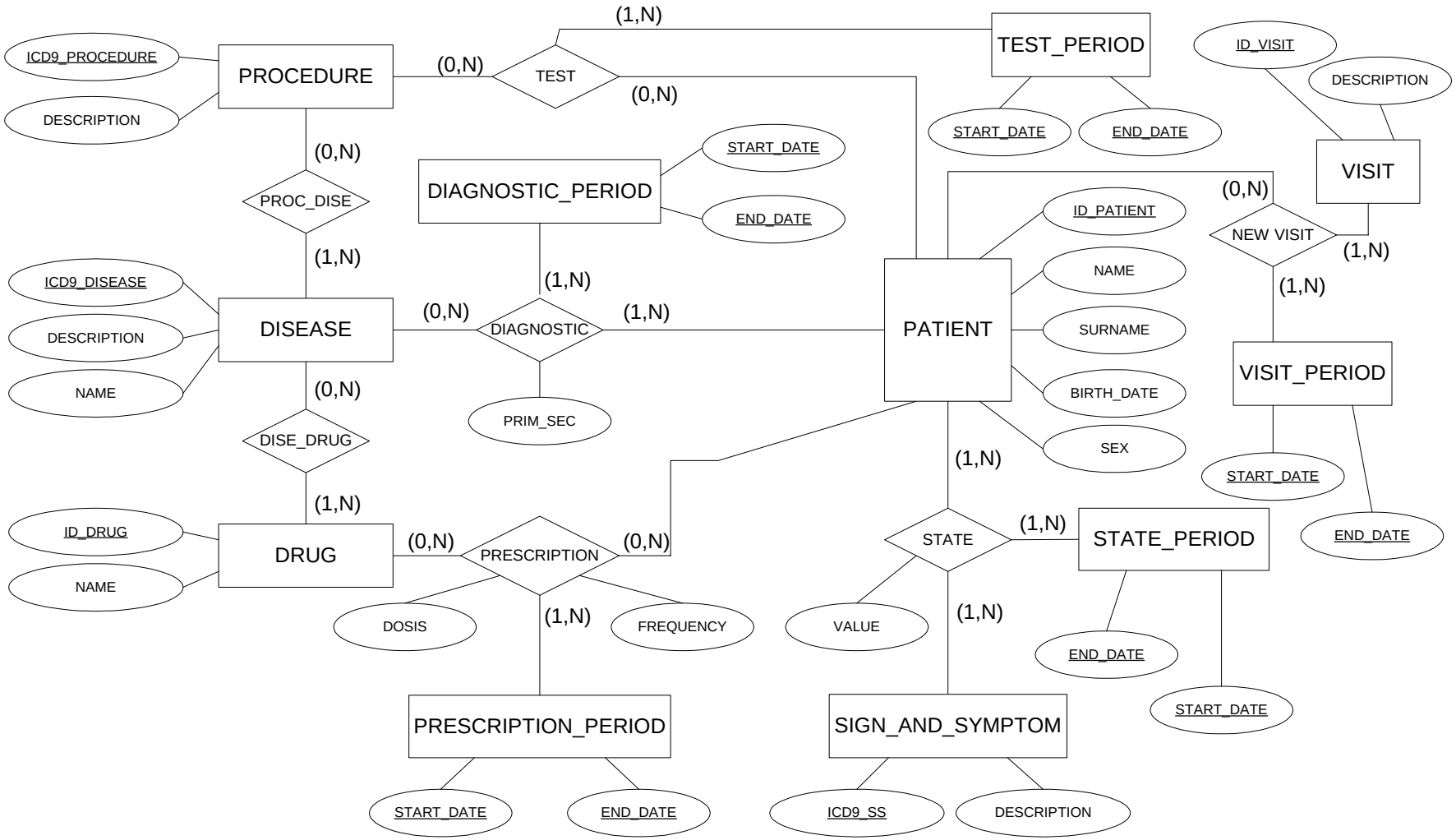

Figure 19: EER model for medical knowledge production including all temporal aspects

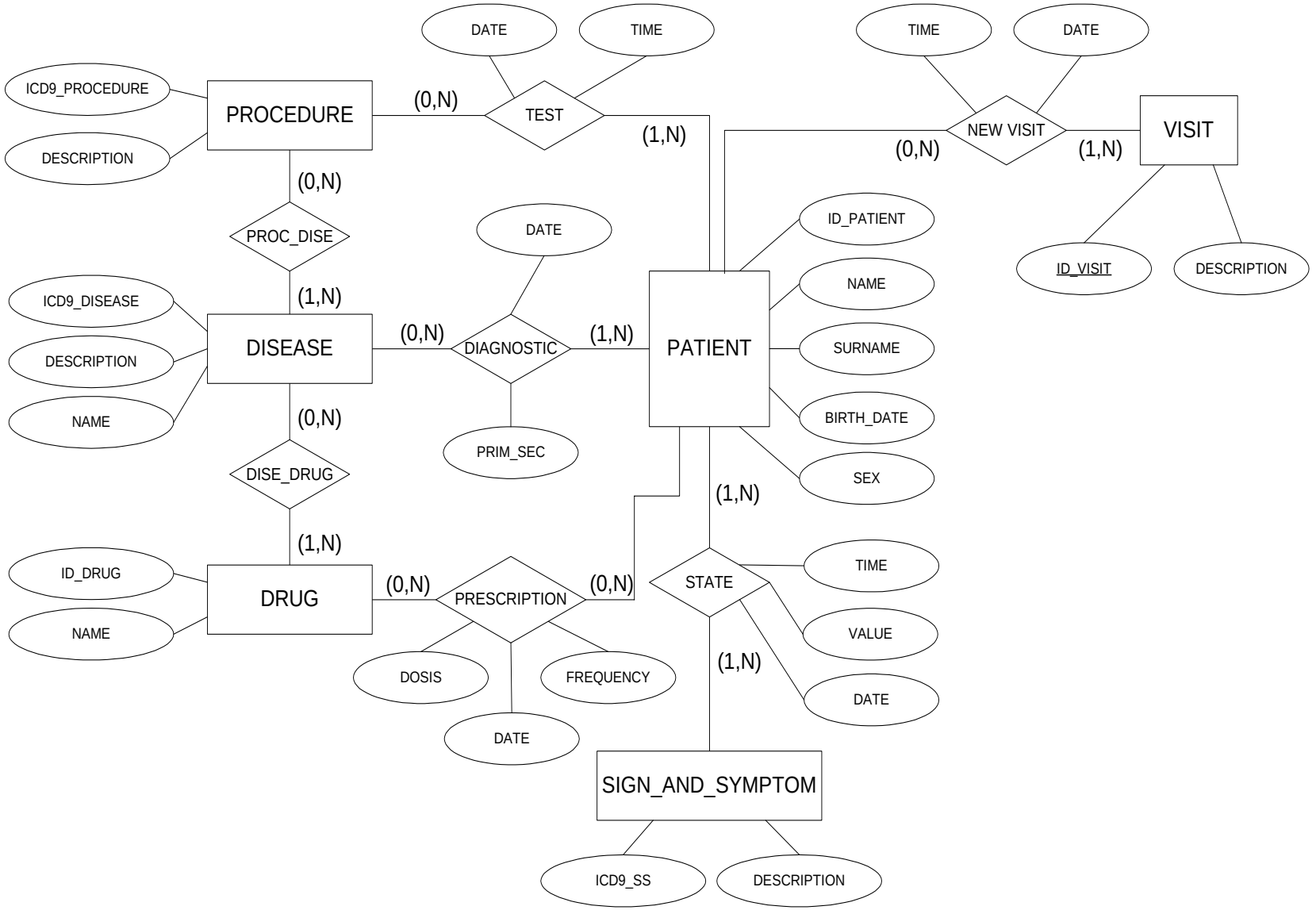

Figure 20: EER model for medical knowledge production (including user defined times)

8 Conclusions

Temporal aspects are prevalent in most database applications, but they are also difficult to capture elegantly using an ER model. There have been presented many extensions of ER model, which allows capturing different aspects of time. The proposed extensions are based on different approaches. One approach is to devise new notational shorthands that replace some of the patterns that occur frequently in ER models when temporal aspects are being modelled. Other approach is to change semantics of the existing ER model constructs, making them temporal. There were also presented some concepts in TSQL2 query language.
In more detail, we present the TIMEER model, which was chosen for database modelling for medical knowledge production. The TIMEER model extends the EER model with new enhanced modelling constructs with implicit temporal support. The new constructs provide support for capturing lifespans of entities and relationships, transaction times for all modelling constructs, and valid times for attributes and relationships. Temporal aspects of the modelling constructs are captured using either instants or temporal elements. Support for multiple granularities is included. The database designer may or may not use the new temporal constructs, and the resulting model is upward compatible with respect to the EER model.
Database model has been introduced for storing the information generated in the processes of medical diagnosis and treatment. The model was designed to ease the process of generating knowledge structures able to help physicians in their daily on-line duties and also to make off-line studies.

9 Acknowledgements

This work was realized as a result of an integrated action between the Department of Computer Science and Mathematics at the Rovira i Virgili University, Tarragona (Spain) and the Institute of Informatics at the Faculty of Electrical Engineering and Computer Science, University of Maribor (Slovenia).

10 References

[1] A. Salatian, J. Hunter: Deriving trends in historical and real-time continuously sampled medical data, Journal of Intelligent Information Systems 13, Kluwer 1999, pp. 47-71.
[2] B. Tauzovich: Toward Temporal Extensions to the Entity-Relationship Model, The 10th International Conference on the Entity Relationship Approach, pp. 163-179, October 1991.
[3] C. Bettini, C. E. Dyreson, W. S. Evans, R. T. Snodgrass, X. S. Wang: A Glossary of Time Granularity Concepts, Temporal Databases – Research and Practice, LNCS 1399, 99. 406 – 413.
[4] C. Combi, Y. Shahar: Temporal reasoning and temporal data maintenance in medicine: issues and challenges, Comput. Biol. Med. 1997, pp. 353-368.
[5] C. Shahabi: Spatial and Temporal Databases, 2001, http://infolab.usc.edu/csci599/Fall2001/index.html
[6] C. Dyerson, F. Grandi, W. Käfer, et. al.: A Consensus Glossary of Temporal Database Concepts; SIGMOD, Vol. 23, No.1, March 1994.
[7] D. Riaño: Guideline Composition from Minimum Basic Data Set, CBMS 2003, NY, USA, 2003.
[8] E. Zimanyi, C. Parent, S. Spaccapietra: TERC+: A Temporal Conceptual Model, International Symposium on Digital Media Information Base, Nara, Japan, November 1997.
[9] H. Gergersen, C. S. Jensen: Conceptual Modelling of Time-Varying Information, A TIMECENTER Technical Report, September 1998.
[10] ICD-9-CM, Official Guidelines for Coding and Reporting, April 1, 2005.

[11] J. C. Augusto: Temporal reasoning for decision support in medicine, Artificial Intelligence in Medicine 2005.
[12] P. P-S. Chen: The Entity-Relationship Model – Toward a Unified View of Data, ACM Transactions on Database Systems, Vol.1, No.1, pp. 9-36, March 1976.
[13] R. Elmasri, S. B. Navathe: Fundamentals of Database Systems, The Benjamin/Cummings Publishing Company, 2. edition, 1994, ISBN 0-8053-1753-8.
[14] R. Elmasri, G. T. J. Wuu: A Temporal Model and Query Language for ER databases, Proceedings of the 6th International Conference on Data Engineering, pp. 76-83, 1990.
[15] R. T. Snodgrass: The Temporal Query Language TQuel, ACM SIGMOD International Conference on Management of data 1984, pp. 204-213, ISBN 0-89791-128-8.
[16] R. T. Snodgrass, I. Ahn: A taxonomy of time in databases, ACM SIGMOD International Conference on Management of data 1985, pp. 236-246, ISBN 0-89791- 160-l/8.
[17] R. T. Snodgrass, I. Ahn, G. Ariav, et. al.: TSQL2 Language Specification, September 1994.
[18] V. S. Lai, J-P. Kuilboer, J. L. Guynes: Temporal Databases: Model Design and Commercialization Prospects, DATA BASE, Vol. 25, No. 3, pp. 6-18, 1994.