Available online at www.sciencedirect.com
SCIENCE DIRECT®
Data & Knowledge Engineering 58 (2006) 410–435
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

REERM: Reenhancing the entity–relationship model

Luis González Jiménez *

Department of Business and Economics, Universidad de La Rioja, Cigüeña, 60, 26004 Logroño, Spain

Received 25 January 2005; accepted 17 May 2005
Available online 13 June 2005

Abstract

The entity–relationship model (ERM) remains one of the most widely used SW Engineering techniques. Much research effort has focused on incorporating into database design in general, and ER modeling in particular, the historical support requirements arising from the temporal nature of dynamic domains. This paper discusses REERM, an extension to the enhanced ERM, whose primary objective is to address this problem, but that may bear positively on other aspects of conceptual modeling.

REERM redefines the concept of entity, introduces additional relations classification criteria, adds the construct event and uses derived attributes and relations. Its application is illustrated with a case study.

Keywords: Conceptual modeling; Entity–relationship model; Temporal databases

1. Introduction

The entity–relationship model (ERM) [3] remains one of the most widely used modeling techniques in the field of Software (SW) Engineering. Arguably, its success derives in large measure from its application to the logical design of relational databases. However, in this paper's view, an even better reason stems from ERM's proven validity as a conceptual data model, not biased by either design or implementation. The purpose of conceptual data modeling is to describe a

* Tel.: +34 941299573; fax: +34 941299393.
E-mail address: luis.gonzalez@dee.unirioja.es

0169-023X/$ - see front matter
doi:10.1016/j.datak.2005.05.004

database information content, rather than the storage structures that managing the information will require [2]. In this sense, the ERM has no parallel given the bias towards a given implementation alternative inherent in object modeling. And, lest this be overlooked, strictly conceptual data modeling is at the basis of IS requirements engineering.

In an effort to overcome some limitations, the ERM has often been revisited. However, very few modifications of the model formulated by Chen in his seminal paper have gained widespread acceptance, notably the introduction of the category abstraction. The resulting enhanced or extended version (EERM) adds two additional constructs or representational terms to those originally established: subset hierarchy and generalization hierarchy [25,7]. The former specifies potentially overlapping subsets (called subtypes) of an entity (the supertype), the latter specifies disjoint subsets of the supertype entity [31].

A subject which has deserved much attention by researchers is the incorporation into database design in general [35,37] and ER modeling in particular [13] of the requirements as regards history support (HS) that arise from the temporal nature of the domain of interest. Interestingly enough, most of the proposals [9,19,20,24,4–6,8,21–23,29,34,33] surveyed in [13] do not involve the addition of the *event* either as a new construct or as a variety of those constituting the ontology of the ERM. This is not exclusive of attempts to incorporate time to the ERM. It would appear that it is pervasive, in Temporal Database research in general, the perception that the domain of interest is best contemplated along the axis of time as a succession of states. (As a rare exception, in [33] an event is defined as a specific type of object and its morphology is established, but the syntax is not clearly provided.) Arguably, that perception rests on the assumption that events and states are duals, states may be represented by their delimiting events, and events are implied by states; therefore, temporal (i.e., timestamped) relations encoding states should be sufficient [17].

Conversely, the ERM "reenhancement" that will be proposed in this paper rests on the basic assumption that reality is dynamic by nature, so that it is best modeled as a series of events resulting in a succession of states (given that the former affect the latter), as indicated in Fig. 1.

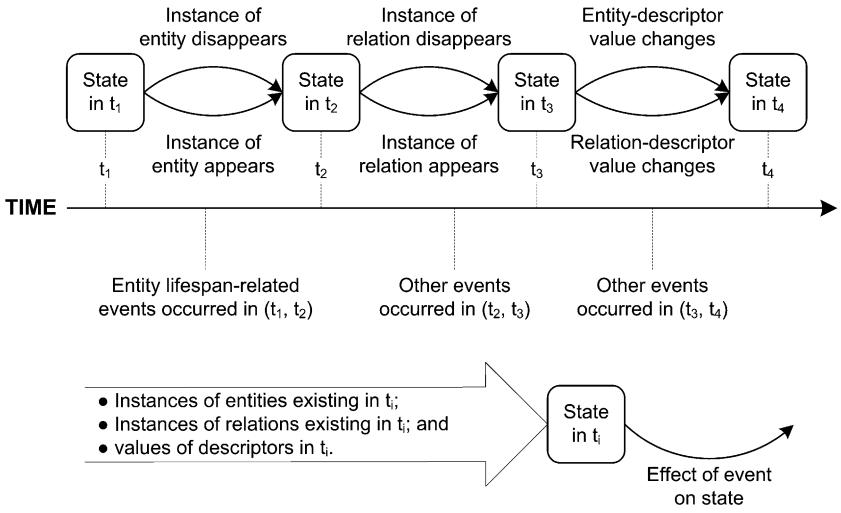

Fig. 1. Reality observed along the time axis.

ERM and above mentioned enhancements being a well established modeling tool, the formulated approach will be herein pursued with the least possible changes and additions to the EERM. Essentially, the reasoning goes as follows.

In Chen's paper [3] it is stated that "An entity is a 'thing' which can be distinctly identified. A specific person, company, or event is an example of an entity. A relationship is an association among entities." Because the two propositions *things exist* and *things happen* are true in reality, yet the things about which either proposition is true are different, Chen's claim may be reformulated in the following terms: "An entity is a thing that exists (independently) and therefore has duration. An event is a thing that happens (independently) and thus is instantaneous. Relationships are associations among entities, events or both. Relations between entities exist (and last), while those among events or between entities and events are instantaneous". Accordingly with this approach to reality (to the domain of interest, to be more precise), *entity* will be redefined so as to exclude *events*, and alternative ways (depending on the type of event, entity lifespan-related or otherwise, and the HS requirements) of modeling both, occurring events and domain states (as represented by entities and relations among them) will be put forward.

Although it is contended that the extension herein proposed may have a positive influence in other aspects of conceptual data modeling, the primary objective is to deal with the need to provide historical information, defined as "data, aggregated or otherwise, about events occurred in the domain within a given period of time, or about the state of the domain at a given point in time in the past". Consistently, it may be said that a database provides history support (HS) if it can supply the necessary flow of data for the hosting Information System (IS) to produce historical information.

Section 2 of this paper discusses the proposed extension. Section 3 provides an example of its application. In addition, standard mathematical notation is used both, to express rules for computation of derived components and to compose business rules, thus providing a possible integrated modeling method with a structure similar to the TEMPORA architecture [32–34]. Finally, Section 4 gives a summary and hints at some avenues for further research.

2. The reenhanced entity–relationship model (REERM)

REERM being an extension of the EERM, it incorporates its ontology, subset and generalization hierarchies included. However, it also redefines some of its constructs or representational terms and adds another one: the event. The first part of this section will precise the meaning of entity and relation in the context of the proposal put forward; other elements of the ERM semantics—the different types of attributes, plus weak and associative entities—will also be dealt with; in addition, the construct event will be defined; finally, changes to the standard notation to accommodate the extension will be set out. Using REERM extended ontology and corresponding notation, the second part discusses the modeling of the state of the domain, the events that take place within it, and the consequences or effects of the latter. The third part addresses concisely the incorporation of time to the model. The fourth and fifth parts define total and partial history support and consequences for modeling purposes of choices in this matter. The last part of this section summarizes the previous discussion into a set of modeling and time-stamping rules.

2.1. Representational terms: Definitions and notation

In REERM's extended ontology, entities do not include events. Thus, the term *entity* will be applicable to agents (persons, organizations of any kind, and organizational units), resources (physical or otherwise), and categories defined by the user for classification and/or aggregation purposes. Intersection or associative entities will not be considered as entities either, but as relations. Nor will weak entities be treated as proper entities, but "clustered" with the strong or parent entities (i.e., as if they were complex attributes). In short, whenever the term entity is used, it shall be construed to refer to a strong entity that cannot be considered as a relation nor as an event.

The ER model classifies *relationships* following three different criteria [31]: degree, connectivity (or maximum cardinality) and membership class (minimum cardinality, sometimes called "optionality"). In addition to this, REERM differentiates three types of relations (the three criteria just enumerated are applicable to any and everyone of them):

- External lasting relations (ELR): those connecting two or more entities (as previously defined, i.e., excluding events, intersection entities and weak entities). ELR may have a degree bigger than two and may be constant or variable. An ELR is constant if, once created, as long as the associated entities' instances exist so does the relation, otherwise it is variable.
- External instantaneous relations (EIR): those linking entities and events or different types of the latter (e.g., PURCHASE and PAYMENT). EIR will always be binary and constant.
- Internal relations (IR): between an entity's or event's weak and strong components (parent entity or event, as applicable). IR will also be binary and constant.

The term *attribute* is generally used with different meanings. It is relevant to the subsequent discussion to differentiate three different kinds of attributes: *descriptors*, *foreign keys* and *identifiers* or *primary keys* (which may be, at the same time, either descriptors or foreign keys). Whenever the term *attribute* is used, it shall be deemed to encompass the three types. In addition, the following convention as regards attribute-ascription is established: *regardless of the cardinalities*, *foreign keys belong to events or relations* (*of the ELR type*), *never to entities*. This has nothing to do with mapping to RM relations, which is not a major concern at this point, it merely serves the purpose of clarity (aside of being semantically consistent).

In [12], an *event* is defined as an instantaneous fact, i.e., something occurring at an instant. Thus defined, the concept of event seems clear enough, except where so-called *macroevents* are concerned. These may be defined as events with duration, that is, events that occur over an interval of time [12]. REERM assumes macroevents not to exist (events being instantaneous by definition); rather they are chains of events and the states that obtain in between (e.g., a plane takes off, is flying for a given period of time, and then it lands), or events whose effect on the domain is limited to a known time period (e.g., renting an apartment for one year).

Attributes of events (of any kind: identifiers, descriptors and foreign keys) are *constant*. Events take place at a given point in time and, by definition, they are not subject to changes, because the past cannot be altered. In this sense, events are *valid forever*, which does not mean that changes in the state caused by them are everlasting. Indeed, the consequences of an event may last in time, but events as such are *instantaneous*, they take place at a point in time and have no perceptible duration.

For the incorporation of events to the model, as a representational term or construct, the following should be noted:

- The categories that are applicable to entities are also valid for events, i.e., events have (constant) attributes and may have strong and weak components; subset and generalization hierarchies are also applicable, and there may be relations (of the EIR type) connecting either different types of events, or events to entities.
- Events may be linked among them or with entities (as herein defined), never with an ELR (see Fig. 2).
- Events should comply with a standard to be included in a model: they should have some effect on the state of the domain; i.e., there have to be entity descriptors, ELR attributes, or both, derived from the event (alone, or in combination with other types of events).

As shown in Fig. 3, notation to be used is standard (use of "crow's feet notation" is just a matter of personal preference) with the following exceptions:

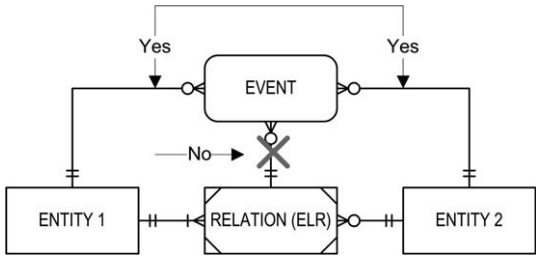

Fig. 2. Events and ELR cannot be linked.

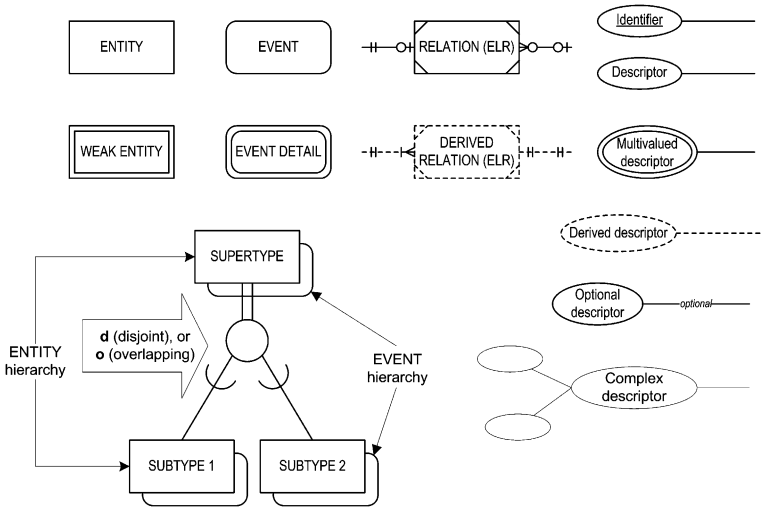

Fig. 3. REERM extended notation.

- Regardless of their having descriptors and internal identifiers (keys) or not, ELR are always represented using the standard symbol for "associative/intersection entities".
- Both, variable and constant ELR may be modeled as derived; in this case they are represented using dotted lines.
- Round-corner boxes are used for events (both for their weak and strong components and, where applicable, for supertypes and subtypes).

2.2. Modeling state, events and state-changes

The state of the domain of interest at an instant, when there is no activity going on, may be described, at a certain level of abstraction, by:

- the existing instances of each defined type of entity and the values of their attributes (identifiers and descriptors, including those of the dependent weak entities), and
- the existing instances of each type of external lasting relation and the values of their attributes (foreign keys, internal identifiers and descriptors).

As already stated, entities' identifiers are always *constant*, and so do IR linking parent (strong) entities and dependent (weak) ones. In addition to this, there may be ELR and descriptors of either entities or relations that are constant as well. Thus the state of the domain, as represented by the conceptual data model, may suffer the changes summarized in Fig. 4, that also provides a certain classification of events.

According to Fig. 4, the events that may take place in the domain of interest and that imply changes to the contents of the database, may be classified in two groups:

- *Entity lifespan-related* events, consisting either in the appearance of a new instance of an entity (and definition of values of its identifier and constant descriptors) or in the disappearance, at least from the domain of interest, of one of them.
- Events which cause changes in the values of variable attributes of entities or *external lasting relations*, or that result in the appearance or extinction of instances of the latter.

Note that any event which has both effects, like HIRE NEW EMPLOYEE, may be decomposed into two events, each belonging to either group: CREATE NEW EMPLOYEE and MAKE EMPLOYMENT CONTRACT (which happens to be applicable to old employees). For the sake of clarity, in the remainder of this paper events in the first group will always be referred to as *entity lifespan-related events*, while those in the second group will be called just *events*.

There are two alternative ways to register in the database the occurrence of events of both types and their effect on the state of the domain:

- Directly registering the changes in the state of the domain resulting from the event. This alternative may result in loss of information, depending on whether the event only adds new elements or it involves elimination or modification of those existing before the occurrence of the event.

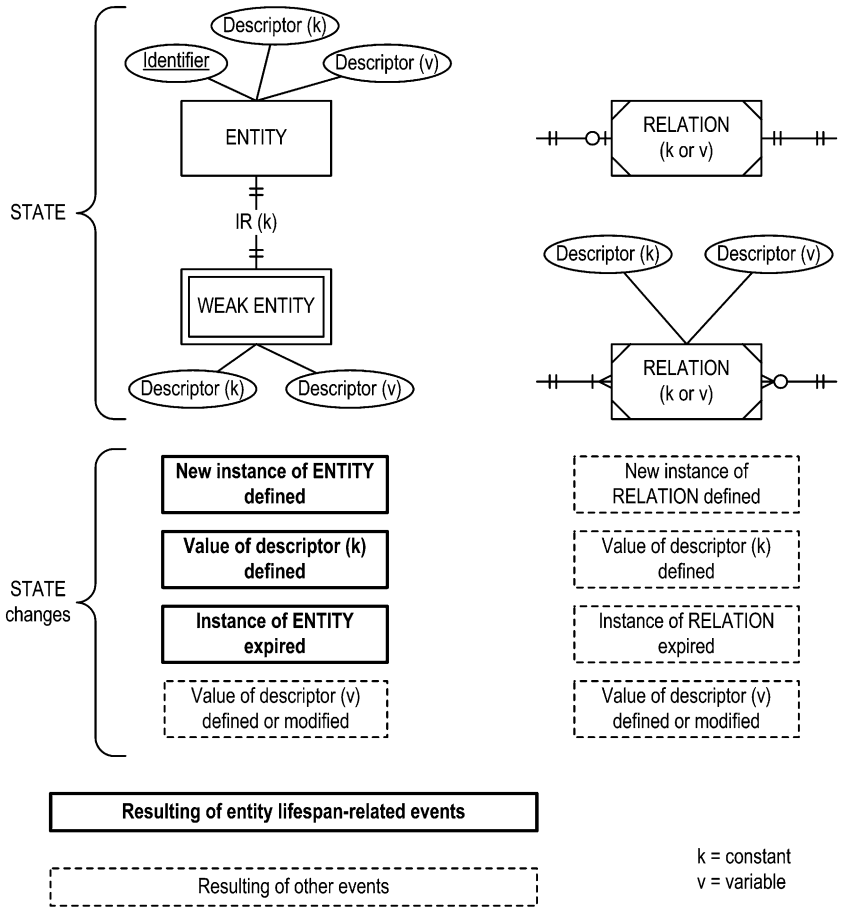

Fig. 4. State and causes of state's changes.

- Modeling both, the event and its *derived* effect on the state of the domain, thus avoiding any loss of information.

2.3. Time magnitudes and units of measurement

Temporal database literature usually gives prominence to transaction and valid times [37], while the *event* (*occurrence*) *time* tends to be relegated to a secondary place. In [12], these three time magnitudes are defined as follows.

- The *transaction time* of a database fact is the time when the fact is current in the database and may be retrieved.
- The *valid time* of a fact is the time when the fact is true in the modeled reality.
- The *event occurrence time* of an event is the instant at which the event occurs in the real-world.

Transaction and valid times are generally not time instants, but time intervals. Consistently, it is possible to define starting and ending transaction/valid times.

When any of the time magnitudes just defined is associated with an object (e.g., an attribute or a tuple), it is called a *timestamp*. The concept may be specialized to valid timestamp, transaction timestamp, interval timestamp, etc. [12].

REERM chooses to work with *event* and *valid* times, because these are the times present in the domain of interest, the ones with which users are familiar and work with. Thus, events will always carry the *event timestamp* (*ET*) and, when applicable, either or both *starting* and *ending valid timestamps* (respectively, VT_s and VT_e). Entities and *ELR* may be timestamped with either or both valid times or not be timestamped. All other elements (attributes, IR, EIR, and weak components of entities and events) will never be timestamped.

That said, *transaction time* may also be incorporated to the model. To that end, when defining the processes or methods resulting in the population of the database, or the business rules data must comply with, the legal relation between event or valid time, as applicable, and the *insertion time* (defined as the current time when the data input takes place) has to be specified. Thus, event time may relate to insertion time, *IT*, (which, for implementation purposes, would be the *starting transaction time*) in either one of three ways: $ET = IT$ (neither anticipated nor delayed updates are allowed), $ET \leqslant IT$ (delayed update is allowed), or $ET \geqslant IT$ (anticipated update is allowed). Likewise, valid times—either starting, VT_s, or ending, VT_e, such that: $VT_s \leqslant VT_e$—will relate to the insertion time in one of the three defined ways.

In addition to the *event occurrence timestamp* (*ET*), events will always have the attribute *ordinal*, this pair being the composite identifier or primary key for events and providing, at the same time, the sequence of occurrence of events that have taken place at the same point in time.

Another matter to be addressed is the choice between what temporal database literature calls *models of time* (Fig. 5). Time values are either instants (points in time) or intervals (periods of time). In both cases, they are *numbers*. Furthermore, it seems rather convenient to assume they are *nonnegative*. What remains to be established is whether they will be considered as Reals (thus

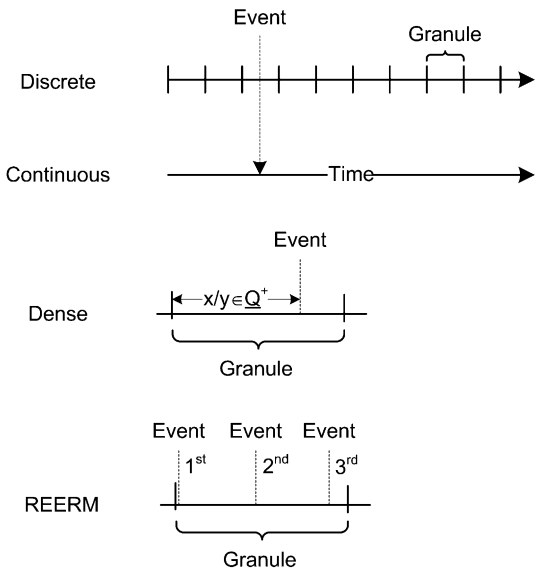

Fig. 5. Models of time and REERM ET.

supporting the "continuous model of time"), Rationals (respectively "dense model") or Integers (respectively "discrete model").

As shall be discussed in Section 2.4, only the first of these three alternatives is valid should total HS be required. The discrete model supports only approximate timing of events and if the dense model is chosen, there will be "gaps" whenever an event happens at a point in the (continuous) time axis which has no corresponding rational number. For implementation purposes, both the continuous and the dense models are not feasible, available clocks being able to provide but a discrete measure of time. (Only a seemingly dense model is feasible if time magnitudes are expressed in a time unit "coarser" than the clock's tick; e.g. if the clock is precise down to seconds and the time data type used is *YYYYMMDDhh*.) Yet, for non-critical systems the discrete model would seem to be sufficiently precise. Going a bit further, it stands to reason that, at least for business IS, SQL-supported *time data types* or truncated versions thereof are sufficient. Thus: *Date* (YYYYMMDD), *Time* (hhmm), UNIX-style *Timestamp* (YYYYMMDDhhmmss), *YYMM*, *YYYYMMDDhh*, and so on. Aside of being the most viable option implementationwise, these are the time units users will usually be willing to work with.

2.4. Total history support

Now assume a database is designed using the REERM ontology defined in Section 2.1 and in accordance with the following specifications:

(1) The database is perpetual, i.e., data cannot be deleted.
(2) Events consisting in the appearance of a new instance of an entity (and definition of values of its identifier and constant descriptors) are registered directly.
(3) All other events (all but *entity lifespan-related* events) are defined in the model and will consistently be registered. In addition, they are *timestamped* with its *event* (*occurrence*) *time* (*ET*) and, if their effect is limited in time, with their *starting* and *ending valid times* (VT_s and VT_e).
(4) ELR (constant and variable), ELR' descriptors (constant and variable) and variable entity-descriptors are *derived* from the events specified in #3 above.
(5) Entities carry *starting* and, unless they are everlasting, *ending valid timestamps* (which is consistent with #1 above and solves the problem of registering non-everlasting entities extinction or cancellation).
(6) Consistently with a "continuous model of time" both, valid and event timestamps are real numbers.

The database thus designed would provide *total history support*. It would supply information, aggregated or not, about events (of both types) occurred within a given period of time as long as such period is contained in the time interval $[inception, now]$. Let $[a, b] \in [inception, now]$ be the chosen time period. The events occurred within that period would be:

- Entity lifespan-related events:
  - Entity instance creation: select instances for which it holds that $VT_s \in [a, b]$.
  - Entity instance extinction or cancellation: select instances for which it holds that $VT_e \in [a, b]$.
- All other events: select instances for which it holds that $ET \in [a, b]$.

It would be possible as well to reconstruct the state of the domain at any point in time contained in [*inception*, *now*]. Let Ω be any past point in time, such that $\Omega \in$ [*inception*, *now*]. The state of the domain at Ω may be reconstructed as follows:

- Events: select instances for which it holds that $ET \leqslant \Omega$.
- Entities: select instances for which it holds that $VT_s \leqslant \Omega \wedge (VT_e > \Omega \veebar \nexists VT_e)$.
- ELR attributes and variable descriptors of selected entities: compute using selected events.

2.5. Modeling partial history support

Note that for a REER-modeled database to give total HS it would be necessary:

- the database to be perpetual (which implies that deletion and modification of non-derived data are both forbidden),
- ELR and time-varying entity-descriptors to be derived,
- to keep records of all the events, entity lifespan-related or otherwise, having an effect on the state of the domain, including the time of their occurrence, and
- to use the continuous model of time.

Relaxation of above conditions results in databases providing *partial history support*. The degree of HS provided will be a function of four variables or, to put it in another way, HS has four "dimensions", as depicted in Fig. 6.

If it is partial HS that is required clear-cut choices will have to be made regarding:

- entities, ELR and descriptors of either for which HS is necessary (or, in other words, what state changes is HS required for);
- events for which HS is required;

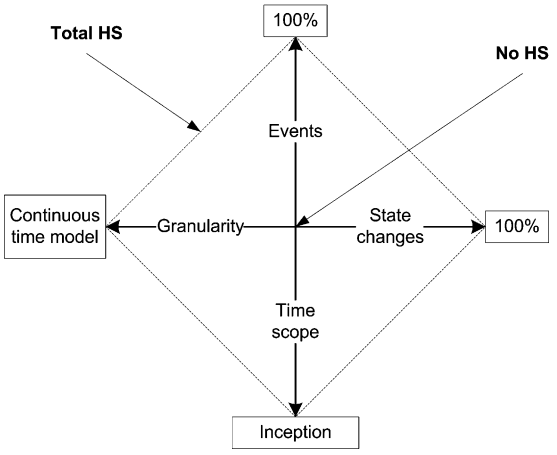

Fig. 6. HS four dimensions.

- how narrowly are events and states required to be located in time (on a case-by-case basis and/or for the database as a whole), which will be determined by the granularity (the length of the used time units: days, hours, minutes, etc.); and
- the HS time-scope, i.e., how far back in time is HS to be available (again, on a case-by-case basis and/or for the database as a whole).

Going back to Fig. 6, a conceptual data model for a database providing partial HS could be represented as a rhomboid contained in the dotted line-rhombus of the figure. Let us now discuss how does REERM addresses partiality.

Regarding granularity (which determines the exactness of the relevant time values), as already noted in Section 2.3, the *discrete model* seems a rather obvious choice. Within this frame, it is a matter of user/client requirements what granularity(ies) should be used.

Limitation of the time scope of the database, i.e. the time interval for which HS is provided, should be taken for granted. Unless it is the first information system that is being developed for a domain, events occurred within the latter from its very beginning will be irrelevant if not unavailable. In addition, once the system is in operation, what has been defined as *vacuuming* capabilities (for the periodical physical removal of historical data in a disciplined manner) [28] will be required. This limitation may be addressed at the logical design stage so that for conceptual modeling purposes, the *perpetual database hypothesis* is preserved. Otherwise, if the existence of a time-lower bound is to be incorporated to the model, duplication of descriptors and ELR requiring HS seems unavoidable. The procedure is straightforward, though it tends to obscure the picture provided by schemata. Let us see how it works for the (derived) descriptor BANK_ACC-balance of Fig. 7, taken from the case study provided in Section 3.

The value of this descriptor at any point in time subsequent to BANK_ACC-VT_s is computed adding the total of each and every related RECEIPT, and deducting thereof the amount of all related PAYMENTs. Now, let Ω be the database time-lower bound, so that it holds: PAYMENT-$ET \geqslant \Omega$, RECEIPT-$ET \geqslant \Omega$ and BANK_ACC-VT_e = **null** $\veebar$ BANK_ACC-$VT_e \geqslant \Omega$. The information contained in $\{PAYMENT \mid ET < \Omega\}$ and $\{RECEIPT \mid ET < \Omega\}$ required to compute BANK_ACC-balance has to be stored, though in an aggregated form, as the non-derived descriptor BANK_ACC-initial_balance that takes the following values:

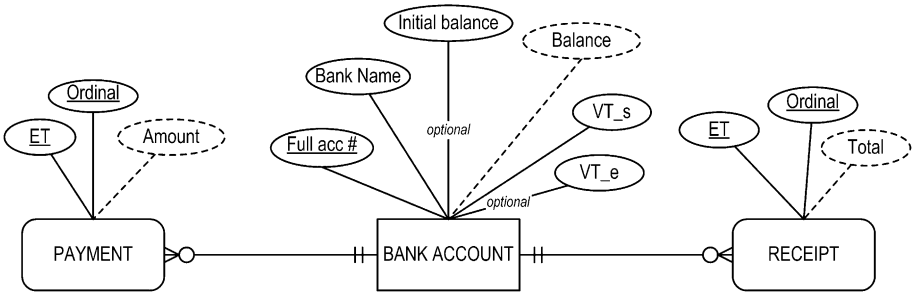

Fig. 7. Modeling descriptors with HS in a non-perpetual DB.

- BANK_ACC-initial_balance = **null** $\Longleftrightarrow$ BANK_ACC-$VT_s \geqslant \Omega$.
- BANK_ACC-initial_balance $\in \{x \in \underline{R}^+ | x \cdot 10^2 \in \underline{Z}^+\} \Longleftrightarrow$ BANK_ACC-$VT_s < \Omega$.

As shown in Fig. 8, this procedure may be used for derived ELR in a similar fashion.

An alternative to selective modeling of initial values is to incorporate to the model the initial state as a whole. It does not seem necessary nor convenient the existence of this memory of the initial state to show in the schemata. However, the information contained in this "shadow diagram" would be used as required. To begin with, for the definition of derived elements' values.

The last two dimensions of HS refer to what events and state changes HS will be provided for (the first quadrant in Fig. 6). Two types of problems may arise at this point:

- Violation of referential integrity constraints. In principle, an entity not requiring HS need not be timestamped and (instances of it) should be deleted when expired or cancelled. However, if it is related (by an EIR) to an event (requiring HS), deletion of (an instance of) the entity may result in instances of the event containing FK values that do not correspond to the primary key of an existing instance of the entity. To avoid unwarranted complications in this matter, the proposed solution is not to allow deletion of (instances of) entities and, if HS is not required, to add a descriptor indicating cancellation (alternatively, an *ending valid timestamp*).
- A variable element (ELR, ELR descriptor or entity descriptor) does not require HS and is consistently modeled as non-derived. However, there are other HS requiring elements (entities or events) in the model that provide the necessary information for the former to be derived. To avoid data inconsistencies, in this case, the elements of reference, regardless of their not requiring HS, should be modeled as derived.

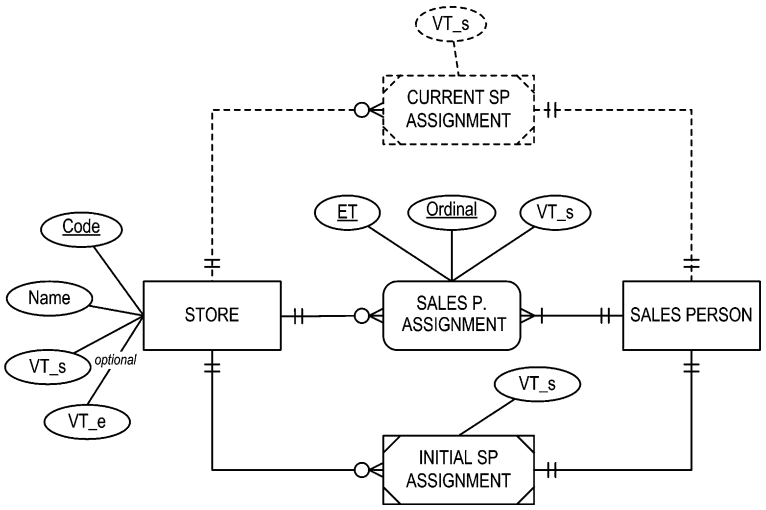

Fig. 8. Modeling HS-requiring ELR in a non-perpetual DB.

2.6. REERM modeling and timestamping rules

As a result of the previous discussion, the following rules for modeling and timestamping may be formulated.

(1) Conceptual modeling may be carried forward on either one of the following assumptions:
   (a) *the database is perpetual*, this assumption may be removed at a later stage of design and taking into account HS requirements;
   (b) there is a general time-lower bound for HS and/or specific ones.
(2) Within the HS time scope ([*inception*, *now*] if the DB is perpetual, [*time-lower bound*, *now*] otherwise), neither events nor entities can be deleted.
(3) Regarding time modeling, i.e. time magnitudes and units of measurement, the following applies:
   (a) The time magnitudes (*timestamps*) that will be visible in the resulting diagrams or schemata will be the *event occurrence time* (*ET*) and the *starting* and *ending valid times* (respectively, VT_s and VT_e). In other parts of the model (either of the database, or of the hosting system), another two time magnitudes will appear: *insertion time IT*, defined as the current time when data are created or modified, and *time intervals*.
   (b) A *time data type* must be specified for each time magnitude included in the model. (Unless a one-size-fits-all approach is deemed valid for a particular system's model, and one time data type is defined as valid for all the time magnitudes.) Specified types should be consistent with the discrete model of time. More precisely, it is contended that, in most cases, SQL-supported *time data types* or truncated versions thereof would appear to be sufficient.
   (c) When defining the processes or methods resulting in the population of the database, or the business rules data must comply with, the legal relation between event or valid time, as applicable, and the *insertion time* has to be specified. Thus, *ET* may relate to *IT* in either one of three ways: $ET = IT$ (neither anticipated nor delayed updates are allowed), $ET \leqslant IT$ (delayed update is allowed), or $ET \geqslant IT$ (anticipated update is allowed). Likewise, valid times—either starting, VT_s, or ending, VT_e, such that: $VT_s \leqslant VT_e$—will relate to the insertion time in one of the three defined ways.
(4) *Entities* for which HS is required will carry *starting valid timestamps* (VT_s) and, if applicable, *ending valid timestamps* (VT_e). The first one corresponds to the instant where the entity appears in the domain of interest, the second one to the entity's extinction or cancellation. These two time values bound an entity's *lifespan*. The value of VT_e may be unknown at *insertion time*; therefore, it may be an *optional descriptor* (and, hence, have as legal value a place-holder null). Finally, HS-requiring entities with never-ending lifespans will have no VT_e. Entity timestamps will always be constant; both VT_s and VT_e, cannot be modified (substitution of a given value for a place-holder null cannot be considered as a modification).
(5) *Entities* for which HS is not required will carry either an *ending valid timestamp* (VT_e) or a descriptor indicating expiration or cancellation (unless they are everlasting, in which case they will bear neither timestamps nor any other descriptor indicating validity or lack thereof).
(6) *Entities descriptors* may be variable or constant. In the first case and if HS is required, they will be *derived*, otherwise, they will be non-derived. Entities *identifiers* cannot include variable descriptors.

(7) HS-requiring *ELR* (both, variable and constant) will be *derived*. Both, variable and constant *ELR* not requiring HS will be non-derived and will be deleted when expired or cancelled. For the purposes of application of this rule, it has to be taken into account the convention on attribute-ascription formulated in Section 2.1: *regardless of cardinalities*, *foreign keys belong to events or relations* (*of the ELR type*), *never to entities*.
(8) *ELR descriptors* (both, variable and constant) requiring HS will be *derived*, otherwise, they will be non-derived. In any case, descriptors of derived ELR must also be derived.
(9) *Events* attributes are constant, so they must not be modeled as derived.
(10) *Events* will carry the composite identifier formed by the following two descriptors:
  (a) The *event time*.
  (b) An *ordinal*; i.e., a strictly positive integer that allows the chronological ordering of instances of an event bearing the same *event timestamp* value. There are several alternatives regarding the sequence of *ordinal* values:
    (i) it restarts (the counter goes back to 1) for each *granule* (or time unit in which *ET* is expressed); for instance, if $ET \in \underline{YYYYMMDD}$, the sequence would restart on a daily basis;
    (ii) the sequence restarts for a granule that is "coarser" than the *ET*'s; e.g., $ET \in \underline{YYYYMMDD}$, but the sequence restarts yearly;
    (iii) in some cases, it may be useful to work with an *enterprise counter*, that is, to establish as a business rule (or integrity constraint), that two events, no matter whether they are of the same *type* or not, cannot have the same *ET stamp* and the same *ordinal*.
(11) Data integrity requires that events in the model (including entity lifespan-related events) allow for derived ELR' existence and derived ELR and entity descriptors' values to be computed.
(12) Notwithstanding rules 6 through 8 above, ELR and ELR and entity descriptors, regardless of their being constant or variable and of the associated HS requirements, should always be modeled as derived if that is possible without adding to the model events, entities or descriptors thereof for that particular purpose (e.g., if an event is part of the model, because HS is required, and the information content of that event allows to compute ELR or descriptors not requiring HS).
(13) Finally, though this is not inherent nor substantial to the proposed EERM extension, it is suggested that sometimes it may be useful to include in the diagrams or schemata derived (constant) EIR and events' descriptors. Where, if at all, are these events' attributes mapped to in the DB's final design is another matter (an implementation one, as it happens). The point is that they may work as "intermediate results" for definitions making part of the model (e.g., entities' and ELR' derived descriptors values, or business rules definitions). This modeling option is illustrated in the example discussed in the following section.

3. A case study of conceptual data modeling using the REER model

This section provides an example of application of the proposed extension to the conceptual data modeling of a database. The case study is solved on the grounds that of the two options established in REERM rule #1 (Section 2.6), the first alternative has been chosen (i.e.,

the database is perpetual), and that *Date* (*YYYYMMDD*) will be the time data-type for all the timestamps. In addition, although REERM does not involve the use of any particular notation and method for such purposes, rules for computation of derived elements and business rules are defined respectively in Sections 3.2.1 and 3.2.2. In both cases, when the rule includes weak components of events, direct reference to attributes of the strong component is deemed valid; e.g., SALE_DETAIL and SALE-store (FK). This is also applicable to entities, though in the example this case does not arise. Likewise, it is valid to refer to a supertype's attribute as if it were an attribute of the subtype; e.g., SALES_PERSON-SS#.

3.1. Description of the domain and HS requirements

A company owns and manages a number of stores, located in different sales regions. New regions may be added, but they are never removed. Once a store is declared to be located in a particular region, the location is not subject to changes.

Stores trade on different items with business partners who may be suppliers, customers or both. Selling prices may change over time; on the other hand, they are established in the company's catalog and are applicable in every store.

The quantity of any given item available in a store is customarily referred to as the stock of that item in that store. An item may or may not be traded on by all stores; for those just included in the company's catalog, it may happen that none of the stores has stock. The cost of the stock is computed using the monthly moving weighted average unit cost.

Purchase of an item by a store from a supplier is subject to the existence of a "supply schedule" defined for the item, the supplier, and the sales region where the store is located. These schedules include a purchase price which may change over time.

Sale of an item is subject to the relevant item being in the company's catalog and to availability of stock of the item at the selling store. In addition, a sales person currently assigned to the selling store has to participate in each and every sale.

Both, payments to suppliers and receipts from customers are made through one of the bank accounts opened by the company. Payments from customers correspond to the total amount of one or more sales (partial payments of one sale are not allowed). Customers' balances (sales receipts outstanding) cannot exceed their credit limits. Payment of each purchase (one payment, and only one, for each purchase) is timed in accordance with the "payment terms" (number of days between purchase and payment) agreed with the supplier.

Whenever the company's cash management requires funds to be transferred from one bank account to another, an "internal cash transfer" (ICT) is made.

Employees may be sales persons or belong to "other" categories. All employment contracts are for a fixed term; if an employee is to stay after his/her current contract has expired, then a new contract has to be signed. Contracts establish starting and expiry dates, salary and, for sales persons, the rate applicable to compute the commission on sales to which they are entitled.

Sales persons work in the store they have been assigned to. These assignments may change over time. Stores will have one or more sales persons assigned.

Stores may be shut down, bank accounts closed, employees (sales persons and otherwise) laid off, items removed from the company's catalog, and business partners accounts cancelled. In all of

these cases, these entities will not henceforth be included in any business transaction. In addition, ''supply schedules'' may expire, hence becoming not applicable anymore.

Stores cannot be closed until all the stock has been cleared and sales personnel posted to another store. Likewise, business partners' accounts cannot be closed until they are settled (thus showing zero balance), and items cannot be taken off the catalog unless all the stock of the item has been sold.

Addresses are kept for stores, employees, business partners and banks (accounts).

Fig. 9 provides additional information about the domain and gives indication of the descriptors and ELR requiring HS. In addition, history support is required for the following events: sales, purchases, payments, receipts, ICT, selling prices update, sales person assignment and employment contracts.

3.2. REER schema and accompanying definitions

Fig. 10 shows the REER schema for the running example. The possibility stated in the rule 13 of those enumerated in Section 2.6 has been used in this case. Derived (constant) EIR and events' descriptors have been included whenever it made definition of other (variable) derived elements or the formulation of business rules easier.

In those cases where non-derived descriptors or relations are variable and thus overwriting is allowed, it has been so indicated in the schema (small arrows with ''TD'' inside). Note in this regard that, according to the rules stated in Section 2.6, the following non-derived elements are always constant: internal relations (between strong and weak components), external instantaneous relations (linking different types of events or entities to events), events descriptors, entity starting and ending valid timestamps and, finally, entity identifiers or primary keys.

3.2.1. Rules for computation of derived elements

This section provides the rules for the computation of the derived elements in Fig. 10. There are nine definitions for constant elements (EIR and events' descriptors) and 10 for variable ones (ELR and descriptors of ELR and entities). Each of these *derivation rules* includes a formal definition and the rule's expression in natural language (NLDR). The latter would be 'intentional rules' and the former 'operational rules' as defined in [18], not as different classes of rules, but as alternative expressions (informal and formal), formulated with different approaches (business context/processes) and that belong in two different stages of information systems analysis (intentional/operational). Both rules' expressions are implementation-free. In the normal course of the conceptual modeling stage of a project, NLDR would be elicited and analyzed and latter on redefined in formal form.

The values of the nine derived constant EIR and events' descriptors shown in Fig. 10 would be computed as follows.

(1) PURCHASE-sales_region (FK) = STORE_LOCATION-sales_region (FK), such that:
- PURCHASE-store (FK) = STORE_LOCATION-store (FK).

NLDR (*derivation rule expressed in natural language*): the sales region of a purchase operation is the one where the purchasing store is located.

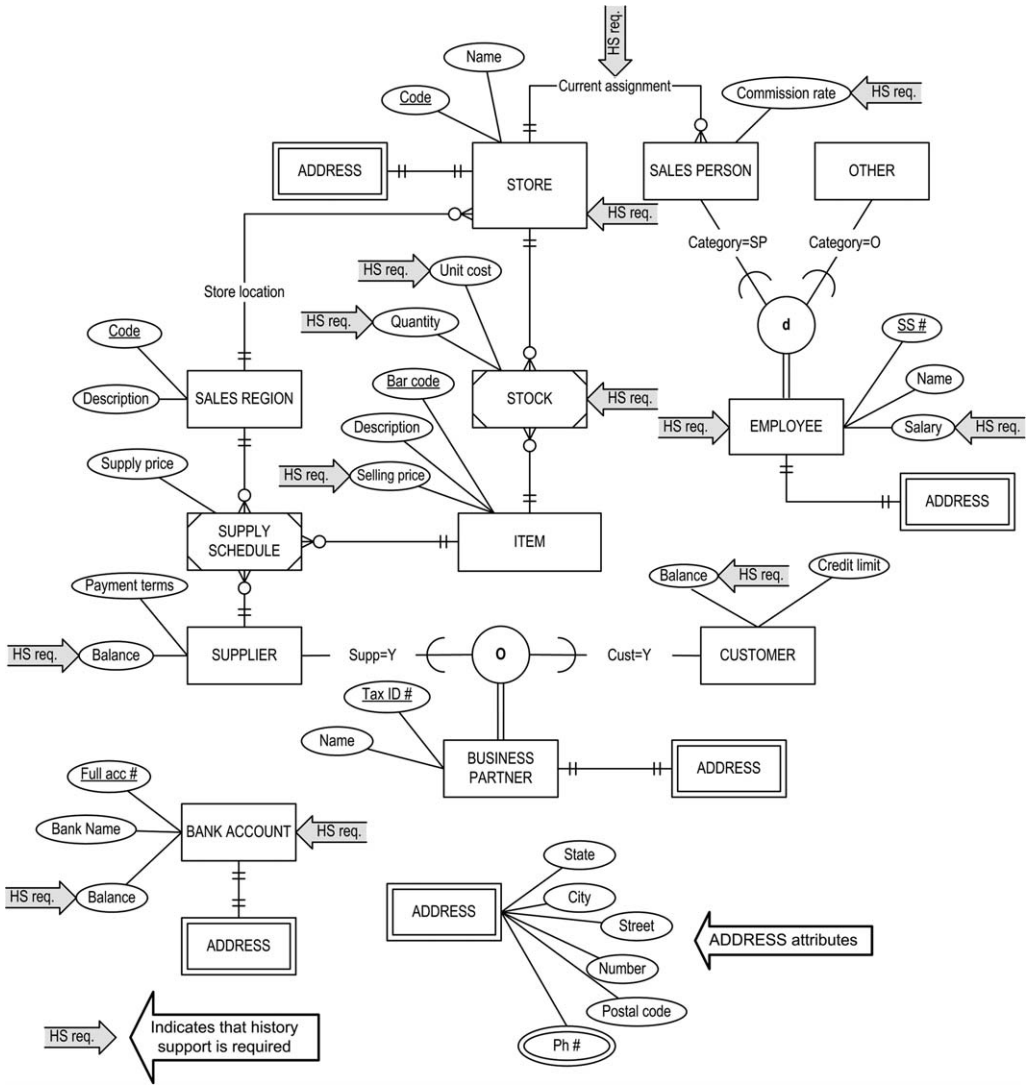

Fig. 9. Preliminary state schema.

(2) PURCHASE-total = $\sum_{\text{PURCHASE\_DETAIL}}$ PURCHASE_DETAIL-quantity × PURCHASE_DETAIL-purchase_price.
NLDR: to compute a purchase operation total, add up the subtotals by item (quantity times unit price).

(3) SALE-total = $\sum_{\text{SALE_DETAIL}}$SALE_DETAIL-quantity × SALE_DETAIL-selling_price.
NLDR: to compute a sale operation total, add up the subtotals by item (quantity times selling price).

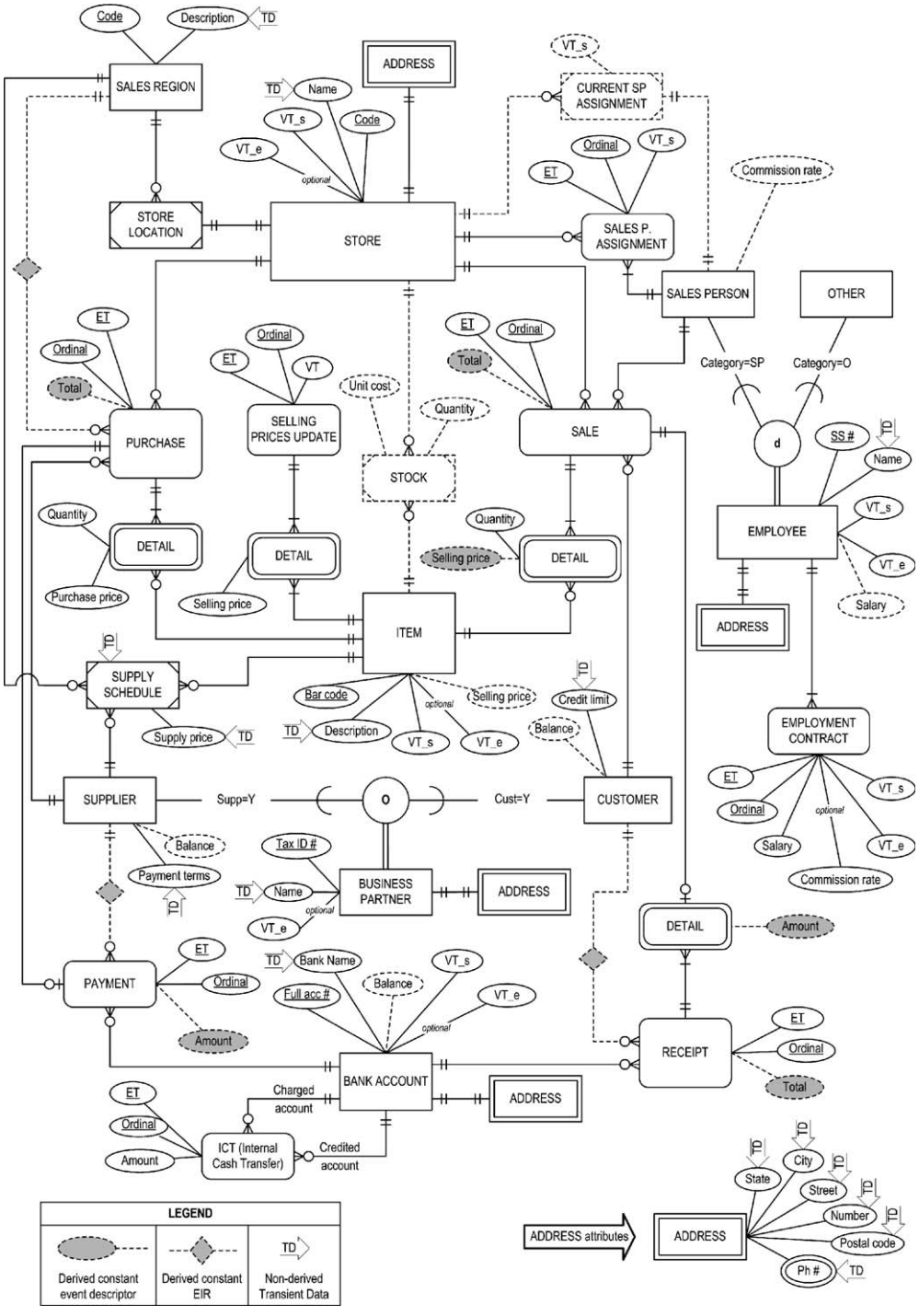

Fig. 10. REER schema.

(4) SALE_DETAIL-selling_price = SPU_DETAIL-selling_price, such that:
- SALE_DETAIL-item (FK) = SPU_DETAIL-item (FK),
- SPU-$VT \leqslant$ SALE-ET, and
- maximum SPU-VT.

Where SPU stands for SELLING_PRICES_UPDATE.
NLDR: selling prices for a sale operation are the current prices at the time (of the sale); for each item, the one in the prices update with the latest valid time prior or equal to the time of the sale.

(5) PAYMENT-supplier (FK) = PURCHASE-supplier (FK), such that:
- PAYMENT-(purchase_ET, purchase_ordinal) (FK) = PURCHASE-(ET, ordinal).

NLDR: the supplier associated with a payment is the one in the purchase operation being paid for.

(6) PAYMENT-amount = PURCHASE-total, such that:
- PAYMENT-(purchase_ET, purchase_ordinal) (FK) = PURCHASE-(ET, ordinal).

NLDR: the amount of a payment equals the total of the purchase operation being paid for.

(7) RECEIPT-customer (FK) = SALE-customer (FK), such that for every RECEIPT_DETAIL:
- RECEIPT_DETAIL-(sale_ET, sale_ordinal) (FK) = SALE-(ET, ordinal).

NLDR: the customer associated with a receipt is the one in the sales operations being collected.

(8) RECEIPT_DETAIL-amount = SALE-total, such that:
- RECEIPT_DETAIL-(sale_ET, sale_ordinal) (FK) = SALE-(ET, ordinal).

NLDR: the amount in each line (or detail) of a receipt corresponds to the total of an uncollected sale.

(9) RECEIPT-total $= \sum_{\text{RECEIPT_DETAIL}}$RECEIPT_DETAIL-amount.
NLDR: to compute a receipt total, add up the amounts in each line of the receipt.

Number 4 above may alternatively be computed as:
4b. SALE_DETAIL-selling_price = ITEM-selling_price, such that:
- SALE_DETAIL-item (FK) = ITEM-bar_code, and
- ITEM-selling_price computed at SALE-ET.

The values of derived variable attributes (ELR and descriptors of ELR and entities) in the schema of Fig. 10 would be computed at a given computing time, CT, such that $CT \in [inception, now]$, as follows.

(1) SUPPLIER-balance $= \sum$PURCHASE-total $- \sum$PAYMENT-amount, such that:
- PURCHASE-supplier (FK) = SUPPLIER-Tax_ID#,
- PAYMENT-supplier (FK) = SUPPLIER-Tax_ID#,
- PURCHASE-$ET \leqslant CT$, and
- PAYMENT-$ET \leqslant CT$.

NLDR: to compute a supplier's balance, add up the totals of all purchases from the supplier and subtract the amounts of all payments made to him/her, in both cases occurred on or before the computing time.

(2) (STOCK-store (FK), STOCK-item (FK)) = (STORE-code, ITEM-bar_code), such that:
- {PU_DETAIL | PU-store (FK) = STORE-code, PU_DETAIL-item (FK) = ITEM-bar_code, PU-$ET \leqslant CT$} $\neq \emptyset$.

Where PU stands for PURCHASE.

NLDR: there is a stock for each and every pair of store and item for which a purchase operation (or part thereof) has happened at any time on or before computation (there may be stocks with zero quantity).

(3) STOCK-quantity $= \sum_{\text{DETAIL}\in A}$DETAIL-quantity $- \sum_{\text{DETAIL}\in B}$DETAIL-quantity, where:
- A = {PURCHASE_DETAIL | PURCHASE-store (FK) = STOCK-store (FK), PURCHASE_DETAIL-item (FK) = STOCK-item (FK), PURCHASE-$ET \leqslant CT$}, and
- B = {SALE_DETAIL | SALE-store (FK) = STOCK-store (FK), SALE_DETAIL-item (FK) = STOCK-item (FK), SALE-$ET \leqslant CT$}.

NLDR: the quantity of a stock results of adding all the associated purchased quantities and deducting all the associated sold quantities (up to the computing time).

(4) STOCK-unit_cost $= \dfrac{\sum_{\text{DETAIL}\in A}\text{DETAIL-quantity}\times\text{DETAIL-purchase\_price}}{\sum_{\text{DETAIL}\in A}\text{DETAIL-quantity}}$, where:
- A = {PURCHASE_DETAIL | $CT \geqslant$ PURCHASE-$ET \geqslant CT - 29$, PURCHASE-store (FK) = STOCK-store (FK), PURCHASE_DETAIL-item (FK) = STOCK-item (FK)}.

NLDR: the unit cost of a stock is the weighted average (purchase) price of the associated quantities purchased within the last 30 days.

(5) ITEM-selling_price = SPU_DETAIL-selling_price, such that:
- ITEM-bar_code = SPU_DETAIL-item (FK),
- SPU-$ET \leqslant CT$, and
- maximum SPU-VT.

Where SPU stands for SELLING_PRICES_UPDATE.

NLDR: an item selling price is the one in the prices update with the latest valid time prior or equal to computing time.

(6) BANK_ACCOUNT-balance $= \sum$RECEIPT-total $- \sum$PAYMENT-amount $+ \sum_{ICT\in A}$ICT-amount $- \sum_{ICT\in B}$ICT-amount, such that:
- RECEIPT-bank_account (FK) = BANK_ACC.-full_acc#,
- PAYMENT-bank_account (FK) = BANK_ACC.-full_acc#,
- RECEIPT-$ET \leqslant CT$,
- PAYMENT-$ET \leqslant CT$,
- A = {ICT | ICT-credited_acc (FK) = BANK_ACC.-full_acc#, ICT-$ET \leqslant CT$}, and
- B = {ICT | ICT-charged_acc (FK) = BANK_ACC.-full_acc#, ICT-$ET \leqslant CT$}.

NLDR: to compute the balance of bank account, add up the totals of all associated receipts, subtract the amounts of all associated payments, add the amounts of all associated ICT where the bank account was credited and deduct those of the ICT where it was charged, in all cases consider operations occurred on or before the computing time.

(7) CURRENT_SPA-(store (FK), sales_person (FK), VT_s) = SPA-(store (FK), sales_person (FK), VT_s), such that:
- SPA-$ET \leqslant CT$,
- maximum SPA-VT_s, and

- EC-$VT_e \geqslant CT$ | EC-employee (FK) = SALES_PERSON-SS#, maximum EC-*ET*.

Where SPA stands for SALES_PERSON_ASSIGNMENT and EC for EMPLOYMENT_CONTRACT.

NLDR: current sales persons assignments are the latest assignments to come into force (as indicated by their starting valid times), occurred on or before the computing time, subject in each case to the sales person still belonging to the company at that time.

(8) CUSTOMER-balance = $\sum$SALE-total − $\sum$RECEIPT-total, such that:
- SALE-customer (FK) = CUSTOMER-Tax_ID#,
- RECEIPT-customer (FK) = CUSTOMER-Tax_ID#,
- SALE-$ET \leqslant CT$, and
- RECEIPT-$ET \leqslant CT$.

NLDR: to compute a customer's balance, add up the totals of all sales by the customer and deduct the totals of all receipts collected from him/her, in both cases occurred on or before the computing time.

(9) SALES_PERSON-commission_rate = EC-commission_rate, such that:
- SALES_PERSON-SS# = EC-employee (FK),
- EC-$ET \leqslant CT$, and
- maximum EC-VT_s.

Where EC stands for EMPLOYMENT_CONTRACT.

NLDR: a sales person commission rate is the one in the latest associated employment contract to come into force, occurred on or before the computing time.

(10) EMPLOYEE-salary = EC-salary, such that:
- EC-employee (FK) = EMPLOYEE-SS#,
- EC-$ET \leqslant CT$, and
- maximum EC-VT_s.

Where EC stands for EMPLOYMENT_CONTRACT.

NLDR: salary for each employee is the one in the latest associated employment contract to come into force, occurred on or before the computing time.

3.2.2. Business rules definitions

As in Section 3.2.1, business rules defined below are expressed both in natural language (*NLBR: business rule expressed in natural language*) and in formal form. They exclude cardinality, domain, entity integrity and referential integrity constraints, which are contained either in the schema itself or would be included in the definition of data types.

(1) PURCHASE_DETAIL-purchase price = SUPPLY_SCHEDULE-supply_price, such that:
- SUPPLY_SCHEDULE-sales_region (FK) = PURCHASE-sales_region (FK),
- SUPPLY_SCHEDULE-supplier (FK) = PURCHASE-supplier (FK), and
- SUPPLY_SCHEDULE-item (FK) = PURCHASE_DETAIL-item (FK).

NLBR (business rule expressed in natural language): purchase operations are subject to the existence of a supply schedule for the supplier, the item and the sales region where the purchasing store is located; provided this, the applicable purchase price is the one in the supply schedule.

(2) SALE_DETAIL-item ∈ {ITEM-bar_code | ITEM-VT_e = **null**}.
NLBR: items delisted (from the company's catalog) cannot be sold.
(3) PURCHASE-store (FK) ∈ {STORE-code | STORE-VT_e = **null**}.
NLBR: purchases can only be made by non closed stores.
(4) PURCHASE-supplier (FK) ∈ {SUPPLIER-Tax_ID# | SUPPLIER-VT_e = **null**}.
NLBR: purchases can only be made from suppliers whose account is not closed.
(5) SALE-store (FK) ∈ {STORE-code | STORE-VT_e = **null**}.
NLBR: sales can only be made by non closed stores.
(6) SALE-customer (FK) ∈ {CUSTOMER-Tax_ID# | CUSTOMER-VT_e = **null**}.
NLBR: sales can only be made to customers whose account is not closed.
(7) SALE-sales_person (FK) ∈ {CURRENT_S.P._ASSIGN.-sales_person (FK) | CURRENT_S.P._ASSIGN.-store (FK) = SALE-store (FK)}.
NLBR: sales of a store can only be made by sales persons currently assigned to the store.
(8) SALE-total + CUSTOMER-balance ⩽ CUSTOMER-credit_limit, such that:
- SALE-customer (FK) = CUSTOMER-Tax_ID#.

NLBR: sale to a customer is subject to not exceeding his/her credit limit as a result of the sale.
(9) SALE_DETAIL-quantity ⩽ STOCK-quantity, such that:
- SALE-store (FK) = STOCK-store (FK), and
- SALE_DETAIL-item (FK) = STOCK-item (FK).

NLBR: sale of an item by a store is subject to the associated stock not being negative as a result of the sale.
(10) RECEIPT-bank_acc. (FK) ∈ {BANK_ACC.-full_acc# | BANK_ACC.-VT_e = **null**}.
NLBR: receipts can only be credited to non closed bank accounts.
(11) PAYMENT-bank_account (FK) ∈ {BANK_ACC.-full_acc# | BANK_ACC.-VT_e = **null**}.
NLBR: payments can only be charged to non closed bank accounts.
(12) PURCHASE-ET ⩽ **now**.
NLBR: delayed recording of purchase operations is allowed, but not their anticipated recording.
(13) SALE-ET ⩽ **now**.
NLBR: delayed recording of sales operations is allowed, but not their anticipated recording.
(14) SELLING_PRICES_UPDATE-ET = **now**.
NLBR: neither delayed nor anticipated recording of selling prices update operations is allowed.
(15) SELLING_PRICES_UPDATE-VT > **now**.
NLBR: selling prices updates cannot be applicable (as indicated by their valid time) on or before their occurrence (date).
(16) EMPLOYMENT_CONTRACT-ET ∈ [EMPLOYEE-VT_s, **now**], such that:
- EMPLOYMENT_CONTRACT-employee (FK) = EMPLOYEE-SS#.

NLBR: employment contracts cannot be postdated nor be dated earlier than the date of registration of the contracting employee (indicated by his/her VT_s).
(17) ICT-credited_acc (FK) ≠ ICT-charged_acc (FK).
NLBR: in ICT the credited and charged accounts cannot be the same.

(18) EMPLOYMENT_CONTRACT-$ET \leqslant$ EMPLOYMENT_CONTRACT-VT_s.
NLBR: an employment contract starting date cannot be earlier than the contract's date.

(19) EMPLOYMENT_CONTRACT-$VT_s <$ EMPLOYMENT_CONTRACT-VT_e.
NLBR: employment contracts expiry date must be later than their starting date.

(20) **If** EC-employee (FK) $\in$ {EMPLOYEE-SS# | EMPLOYEE-category = SP},
then EC-commission_rate $\in$ [0%, *maximum rate*], **otherwise** EC-commission_rate = **null**.
Where EC stands for EMPLOYMENT_CONTRACT.
NLBR: the commission rate in employment contracts with sales persons may vary from zero to an established *maximum rate*; contracts with other employees will not indicate this rate (hard copies of the contract should read "not applicable" in the corresponding space).

(21) EC′-$ET >$ EC-$ET \Longleftrightarrow$ EC′-$VT_s >$ EC-VT_e, where:
- EC′, EC $\in$ EMPLOYMENT_CONTRACT and EC′-employee (FK) = EC-employee (FK).

NLBR: for any two employment contracts with the same employee, the starting date of the latest one must always be later than the expiry date of the other one.

(22) SALES_PERSON_ASSIGNMENT-$VT_s <$ EC-VT_e, such that
- EC-employee (FK) = SALES_PERSON-SS#, and
- maximum EC-*ET*.

Where EC stands for EMPLOYMENT_CONTRACT.
NLBR: the starting date of a sales person assignment has to be earlier than his/her ending valid time (expiry date of his/her latest employment contract).

(23) SALES_PERSON_ASSIGNMENT-$ET \leqslant$ **now**.
NLBR: delayed recording of sales persons assignments is allowed, but not their anticipated recording.

(24) SALES_PERSON_ASSIGNMENT-$VT_s \geqslant$ **now**.
NLBR: sales persons assignments validity cannot start on or before their recording (date).

(25) STORE-$VT_e \in$ (STORE-VT_s, **now**], subject to:
- $\nexists$ CURRENT_S.P._ASSIGNMENT | CURRENT_S.P._ASSIGNMENT-store (FK) = STORE-code, and
- $\nexists$ STOCK | STOCK-store (FK) = STORE-code, STOCK-quantity $\neq 0$.

NLBR: closing of a store is subject to cancellation of any existing sales person assignment (to the store) and clearance of the existing stocks; delayed (but not anticipated) recording of the closing is allowed.

(26) ITEM-$VT_e \in$ (ITEM-VT_s, **now**], subject to:
- $\nexists$ STOCK | STOCK-item (FK) = ITEM-bar_code, STOCK-quantity $\neq 0$.

NLBR: delisting of an item is subject to clearance of any existing stock; delayed (but not anticipated) recording of the delisting is allowed.

(27) BANK_ACC.-$VT_e \in$ (BANK_ACC.-VT_s, **now**], subject to: BANK_ACC.-balance = 0.
NLBR: closing of a bank account is subject to cancellation of any outstanding balance; delayed (but not anticipated) recording of the closing is allowed.

All of above rules refer to events (the last three to entity lifespan-related events) and, with the exception of rules 17 through 21, they all establish constraints on some event's attribute in relation

with the current state of the domain. Hence, they can only be enforced at the relevant event's insertion time. Most of them can be formulated so that they hold at any time, but there is no obvious gain and their expressions become more complex to write and to read. Moreover, because they only apply at insertion time, it is easy to account for the possibility of time-varying business rules; e.g., assigning them starting and ending valid timestamps. This is important given the volatile nature of business rules and their relation with software systems evolution [18,26,27,36].

Finally, note that the rules defined in this section are declarative (i.e., non-procedural), atomic, technology independent and business oriented. Another set of active business rules (not part of the CDM), formulated accordingly with the ECA (Event–condition–action) paradigm [1], would have to be specified.

4. Summary and directions for future research

This paper has discussed a possible reenhancement of the entity–relationship model. The main purpose of this extension is to provide the constructs and the syntax to smoothly incorporate into conceptual data modeling (CDM) the history-support requirements that may arise from the temporal nature of the domain of interest. As an added advantage, the proposed extension allows to make choices regarding the required extent of history support, and to build these choices into the data model.

Arguably, REERM may have a positive influence in other aspects of CDM and of IS requirements engineering (RE) in general, the former being the cornerstone of the latter. For instance, working simultaneously and interactively on and with a list of events and another one of entities should improve elicitation and analysis of information about the domain for CDM purposes. Likewise, it would likely improve communication with the client/user (at the three levels of RE: elicitation, analysis and specification) to work with *event clusters* instead of *entity clusters* [30]. Different units within a business or businesslike organization are concerned with a certain set of events, while entities tend to be ''transversal'', to be present in events concerning several units of the organization.

Another task that may benefit from the modeling approach that has been discussed is the definition of business rules (BR). Their importance [14,15,18,26,27,36], and the need to integrate their analysis and specification with CDM capable of catering to the problems arising of the temporal dimension of the domain of interest have been highlighted long ago [1]. In this matter, two aspects of what has been discussed are worth noting. As regards the REERM itself (discussed in Section 2), it would seem that, more often than not, expressing business rules in terms of events and their relation to pre-/post-states will be the natural thing to do. Not only for passive (declarative) rules, but also for the specification of active rules within the Event–Condition–Action (ECA [1]) paradigm. Secondly, the method for rules modeling drafted in Section 3 is formal (i.e., mathematical) and thus unambiguous and expressed in a universal standard notation, yet simple enough so as not to involve but very basic mathematics. On account of this, it would seem worth exploring the possibility of defining a notation allowing to map REERM schemata into algebraic definitions, better suited for the specification of derivation and business rules, queries and, to some extent, use cases. A possibility consistent with later years trends in formal methods, which point to

their lightweight application [16] and, what is specially relevant to the case, to their integration with other methodologies [10,11].

Acknowledgements

The author wishes to thank the editor, Professor R.P. van de Riet, and two of the anonymous referees for their valuable comments and suggestions.

References

[1] F. Van Assche, P.J. Layzell, P. Loucopoulos, G. Speltincx, Information systems development: a rule-based approach, Journal of Knowledge Based Systems (September) (1988) 227–234.
[2] C. Batini, S. Ceri, S.B. Navathe, Conceptual Database Design: An Entity–Relationship Approach, Benjamin/Cummings, 1992.
[3] P.P. Chen, The entity–relationship model toward a unified view of data, ACM Transactions on Database Systems 1 (1) (1976) 9–36.
[4] R. Elmasri, G. Wuu, V. Kouramajian, A temporal model and query language for EER databases, in: A. Tansel et al. (Eds.), Temporal Databases: Theory, Design, and Implementation. Benjamin Cummings, Database Systems and Applications series, 1993, pp. 212–229.
[5] R. Elmasri, G. Wuu, A temporal model and query language for ER databases, in: Proceedings of the Sixth International Conference on Data Engineering, 1990, pp. 76–83.
[6] R. Elmasri, I. El-Assal, V. Kouramajian, Semantics of temporal data in an extended ER model, in: 9th International Conference on the Entity–Relationship Approach, 1990, pp. 239–254
[7] R. Elmasri, A. Hevner, J. Weeldreyer, The category concept: an extension to the entity–relationship model, Data Knowledge Engineering 1 (1) (1985) 75–116.
[8] R. Elmasri, V. Kouramajian, A temporal query language for a conceptual model, in: N.R. Adam, B.K. Bhargava (Eds.), Advanced Database Systems, LNCS 759, Springer-Verlag, 1993, pp. 175–190.
[9] S. Ferg, Modeling the time dimension in an entity–relationship diagram, in: 4th International Conference on the Entity–Relationship Approach, Computer Society Press, 1985, pp. 280–286.
[10] R.B. France, M.M. Larrondo-Petrie, A framework for integrating formal and structured techniques, Technical Report TR-CSE-93-24, Department of Computer Science and Engineering, Florida Atlantic University, 1993.
[11] R.B. France, M.M. Larrondo-Petrie, From structured analysis to formal specifications: State of the theory, in: Proceedings of the ACM 1994 Computer Science Conference, ACM Press, 1994, pp. 249–256.
[12] H. Gregersen, C.S. Jensen, et al. (Eds.), The Consensus Glossary of Temporal Database Concepts—February 1998 Version, in: O. Etzion, S. Jajodia, S. Sripada (Eds.), Temporal Databases: Research and Practice, LNCS 1399, Springer-Verlag, 1998, pp. 367–405.
[13] H. Gregersen, C.S. Jensen, Temporal entity–relationship models—a survey, IEEE Transaction on Knowledge and Data Engineering 11 (3) (1999) 464–497.
[14] H. Herbst, Business rules in systems analysis: a meta-model and repository system, Information Systems 21 (2) (1996) 147–166.
[15] H. Herbst, G. Knolmayer, T. Myrach, M. Schlesinger, The specification of business rules: a comparison of selected methodologies, in: A.A. Verrijn-Stuart, T.W. Olle (Eds.), Methods and Associated Tools for the Information System Life Cycle, Elsevier, 1994, pp. 29–46.
[16] D. Jackson, J. Wing, Formal methods light: lightweight formal methods, IEEE Computer 29 (4) (1996) 21–22.
[17] C.S. Jensen, R.T. Snodgrass, Semantics of time-varying information, Information Systems 19 (4) (1994) 33–54.
[18] P. Kardasis, P. Loucopoulos, Expressing and organising business rules, Information and Software Technology 46 (11) (2004) 701–718.
[19] M.R. Klopprogge, TERM: an approach to include the time dimension in the entity–relationship model, in: Proceedings of the 2nd International Conference on the Entity Relationship Approach, 1981, pp. 477–512.

[20] M.R. Klopprogge, P.C. Lockeman, Modeling information preserving databases: consequences of the concept of time, in: Proceedings from the 9th International Conference on Very Large Data Bases 399–416, 1983.
[21] P. Kraft, Temporale Kvaliteter i ER Modeller. Hvordan? Working paper, Aarhus School of Business, Department of Information Science, 1996.
[22] V.S. Lai, J.P. Kuilboer, J.L. Guynes, Temporal databases: model design and commercialization prospects, DATA BASE 25 (3) (1994) 6–18.
[23] P. McBrien, A.H. Seltveit, B. Wangler, An entity–relationship model extended to describe historical information. in: International Conference on Information Systems and Management of Data, 1992, pp. 244–260.
[24] A. Narasimhalu, A data model for object-oriented databases with temporal attributes and relationships, Technical report, National University of Singapore, 1988.
[25] S. Navathe, A. Cheng, A methodology for database schema mapping from extended entity–relationship models into the hierarchical model, in: G.C. Davis et al. (Eds.), The Entity–Relationship Approach to Software Engineering, Elsevier North-Holland, 1983.
[26] S. Ram, V. Khatri, A comprehensive framework for modeling set-based business rules during conceptual database design, Information Systems 30 (2) (2005) 89–118.
[27] D. Rosca, C. Wild, Towards a flexible deployment of business rules, Expert Systems with Applications 23 (4) (2002) 385–394.
[28] J. Skyt, C.S. Jensen, L. Mark, A foundation for vacuuming temporal databases, Data and Knowledge Engineering 4 (1) (2003) 1–29.
[29] B. Tauzovich, Toward temporal extensions to the entity–relationship model, in: 10th International Conference on the Entity Relationship Approach, 1991, pp. 163–179.
[30] T.J. Teorey, Database Modeling and Design, Morgan Kauffman, 1999.
[31] T.J. Teorey, D. Yang, J.P. Fry, A logical design for relational databases using the extended entity–relationship model, ACM Computing Surveys 18 (2) (1986) 197–222.
[32] C.I. Theodoulidis, P. Loucopoulos, The time dimension in conceptual modelling, Information Systems 16 (4) (1991) 273–300.
[33] C.I. Theodoulidis, P. Loucopoulos, B. Wangler, A conceptual modelling formalism for temporal database applications, Information Systems 16 (4) (1991) 401–416.
[34] C.I. Theodoulidis, B. Wangler, P. Loucopoulos, The entity relationship time model, in: Conceptual Modelling, Databases, and CASE: An Integrated View of Information Systems Development, John Wiley, 1992, pp. 87–115.
[35] V.J. Tsotras, A. Kumar, Temporal database bibliography update, SIGMOD Record 25 (1) (1996) 41–51.
[36] W.M.N. Wan-Kadir, P. Loucopoulos, Relating evolving business rules to software design, Journal of Systems Architecture 50 (7) (2004) 367–382.
[37] Y. Wu, S. Jajodia, X.S. Wang, Temporal database bibliography update, in: O. Etzion, S. Jajodia, S. Sripada (Eds.), Temporal Databases: Research and Practice, LNCS 1399, Springer-Verlag, 1998, pp. 338–366.

Luis González Jiménez (Cádiz, Spain, 1960) is a Graduate (Licenciado) in Arts (Universidad Autónoma de Madrid, 1982) and Economics and Business (Universidad Complutense de Madrid, UCM, 1991) and holds a Doctorate in Economics and Business (UCM, 1996). Since 1998 he is an Associate Professor (Professor Titular; tenured since 2000) of the Department of Business and Economics of the Universidad de La Rioja (Spain). Before that (1992–1998) he was Assistant Professor at the School of Computer Science of the UCM. He also has some years of experience, both as manager and consultant, mainly in the field of financial planning and control.
His research areas are: information systems requirements engineering (in particular, conceptual modeling) and mathematical models for management decisions support (cost–volume–profit analysis, financial forecasting and investment valuation).