33

From Temporal ER Models to Relations

H. Gregersen, L. Mark, and C. S. Jensen

Many database applications manage information that varies over time, and most of the database schemas for these applications were designed using one of the several versions of the Entity-Relationship (ER) model. In the research community as well as in industry, it is common knowledge that temporal aspects of data are pervasive and important to the applications, but are also difficult to capture using the ER model. The research community has developed temporal ER models, in an attempt to provide modeling constructs that more naturally and elegantly support capturing the temporal aspects. Specifically, the temporal models provide enhanced support for capturing aspects such as lifespans, valid time, and transaction time of data.

Because commercial database management systems support neither the ER model nor any temporal ER model as a model for data manipulation—but rather support various versions of the relational model for this purpose—we provide a two-step transformation from temporal ER diagrams, with built-in support for lifespans and valid and transaction time, to relational schemas. The first step of the algorithm translates a temporal ER diagram into relations in a surrogate-based relational target model; and the second step further translates this relational schema into a schema in a lexically-based relational target model.

1 Introduction

A wide range of prominent, existing database applications manage time-varying information. These include financial applications such as portfolio management, accounting, and banking; record-keeping applications, including personnel, medical-record, and inventory; and travel applications such as airline, train, and hotel reservations and schedule management.

In the database community temporal aspects of information have been discussed over the years. In this paper, we will focus on four distinct types of temporal aspects that are candidates for being stored in a database, namely *valid time*, *lifespan*, *transaction time*, and *user-defined time* [11].

The notion of *valid time* applies to facts: the valid time of a fact is the time when the fact is true in the mini-world. All facts have a valid time, but the valid time may or may not be captured explicitly in the database. As an example, consider an Employee entity "E1" with a Department attribute. A valid time of "June 1996" associated with the value "Shipping" represents the fact that "E1 is in Shipping" is valid during June 1996. When valid time is captured in the database for an attribute such as Department, the database is capable of recording the varying Department values for the Employee entities. If it is not captured, the database will only record one (the current) Department value for each Employee entity.

The *lifespan* of an entity captures the existence time of the entity: the time during which it exists in the modeled mini-world. If lifespans of entities are supported, this means that the model has built-in support for capturing, in the database, the times when entities exist. The lifespan of a entity "E" may be seen as the valid time of the related fact, "E exists." However, we choose to consider lifespans as a separate aspect, since the recording of lifespans of entities is essential for many applications. 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.

The *transaction time* of a database fact is the time when the fact is current in the database. As is the case for lifespans the transaction time of a fact F may be seen as the valid time of a related fact, namely the fact, "*F is current in the database*," but again we have chosen to record transaction time as a separate aspect. Unlike valid time, transaction time may be associated with any structure stored in a database, not only with facts. Thus, all structures stored in a database have a transaction-time aspect. Note that the transaction time does not provide information about the existence time or the valid time of the structure it is associated with, and that it, like all the above-mentioned temporal aspects, has a duration.

User-defined time is supported when time-valued domains for attributes are available in the data model. These are then employed for giving temporal semantics—not captured in the data model, but only externally, in the application code

and by the database designer—to the ER diagrams. For Employee entities, such attributes could record birth dates, hiring dates, etc.

One of the temporally extended ER models developed in the research community is the Time Extended ER model, TIMEER [9]. This model extends the ER model with annotations for specifying the temporal aspects that need to be captured in the underlying data base, and it is the only model that explicitly supports lifespans, valid time, and transaction time. For this reason, we have chosen the TIMEER model as input to the mapping algorithms presented in this paper.

Several algorithms to construct relational schemas from conceptual schemas already exist, e.g., for the ER model [1, 18] and the Extended ER model [7]. These algorithms serve as models for the algorithm presented in this paper, but do not address temporal aspects, the focus of the present paper. More interestingly, four mappings from temporally extended ER models to the relational model have been previously proposed [8, 13, 14, 17]. We consider each algorithm in turn, discussing their specification and their limitations.

The RAKE model [8] provides shorthand notations for indicating valid-time support for attributes and relationship types and lifespan support for entity types. The description of the accompanying mapping algorithm consists of very short informal explanations, accompanied by an example for each of the shorthand notations. The algorithm does not consider transaction time and its combination with other temporal aspects, and it does not consider temporal integrity constraints, neither informally nor formally.

The MOTAR model [14] supports valid time and provides new modeling constructs for specifying temporal attributes (periodic and aperiodic) and relationship types. The mapping algorithm is well designed, and its description is somewhat comprehensive. The algorithm produces a relational schema that is at least in third normal form, given a well-designed argument ER diagram. But like the previous mapping, this mapping considers neither transaction time nor any temporal integrity constraints.

The TempEER model [13] supports both valid time and transaction time. Its mapping algorithm is divided into two steps: First, the mapping algorithm for the TempEER model [7] is reused, slightly rewritten, though, to create relations. Second, all the relations are extended with timestamp attributes to capture the lifespan of the objects stored in the relations. A constraint that limits the lifespans of relationships to be included in the lifespan of the participating entities as well as a constraint that limits the lifespan of an entity in a subclass to be included in the lifespan of the related entity in the superclass are explained, but not formally defined. In contrast, this paper formally defines a comprehensive set of 31 constraints that enforce the ER-specified time-related semantics in the relational context. TempEER's mapping does accommodate transaction time, but only a single time point, rather than a pair, is used for capturing the transaction-time duration of a tuple. This

simplifies the mapping, but also seems to be a somewhat restrictive solution.

Finally, the ERT model [17] supports valid time and has an associated stepwise, precisely formulated mapping algorithm. The algorithm is well-designed and aims to produce third normal form relational database schemas for well-designed argument ERT diagrams. The precise formulation of the steps is a very strong point of this algorithm. This algorithm, however, does not consider transaction time, and temporal integrity constraints.

The present paper's new contributions are several. It precisely and systematically articulates how temporal ER diagrams in the TIMEER model are mapped to a relational platform. This mapping accounts for all the temporal aspects addressed by the existing dozen of temporal ER models, namely valid and transaction time and lifespans, in addition to all their meaningful combinations. The mapping is accompanied by a comprehensive set of precisely defined constraints that are necessary to enforce, on a relational platform, the semantics of valid and transaction time and lifespans captured in temporal ER diagrams.

The mapping consists of two steps. In Step I, we map the TIMEER model to a surrogate-based relational target model. In Step II, we map this model to a lexically-based relational target model. The mappings are based on ideas presented in [3, 7]. We use the interval tuple-timestamp scheme to record all the supported temporal aspects of data.

There are two reasons for first mapping the TIMEER model to the surrogate-based relational target model. First, the semantics of the surrogate-based relational target model is closer to the semantics of the TIMEER model. Second, we believe that future relational models will provide support for surrogate domains as well as temporal domains. Thus, Step I of our mapping meets future needs. Step II of our combined mapping guarantees that our mapping meets today's needs.

This paper is structured as follows. In Section 2, we define the three models used in this paper. In Section 3, we present the mapping from the TIMEER model to the surrogate-based relational target model. In Section 4, the mapping from this model to the lexically-based relational target model is presented. Finally, in Section 5 a summary and future research directions are presented.

2 Argument and Target Models

In this section we present the three models used in this paper. First, we present the conceptual model TIMEER that is the argument to the first step of the mapping algorithm. Second, we define the surrogate-based relational model that is the target model of the first step and the argument model to the second step of the mapping algorithm. Finally, we define the lexically-based relational model, that is the target model of the second step.

2.1 TIMEER

Since its publication, the ER model [2] has had various notations and semantics. It has been extended in order to capture superclass/subclass relationships and complex entity types, to name a but few extensions, and is then known as "the" EER model. The EER model presented by Elmasri and Navathe [7] is chosen as the outset for TIMEER. The reader is assumed to be familiar with that model.

The modeling constructs of the TIMEER model are presented next. The TIMEER model, including its formal semantics, is described in detail in [9]. The model has implicit temporal support, but the EER constructs and their semantics are retained, i.e., notation and semantics are *added* to the EER model to arrive at the TIMEER model. It extends the EER model to better capture, where indicated, temporal aspects of entities, relationships, superclasses/subclasses, and attributes. Figure 1 presents a TIMEER diagram of a company database. We will explain and refer to this diagram throughout this section.

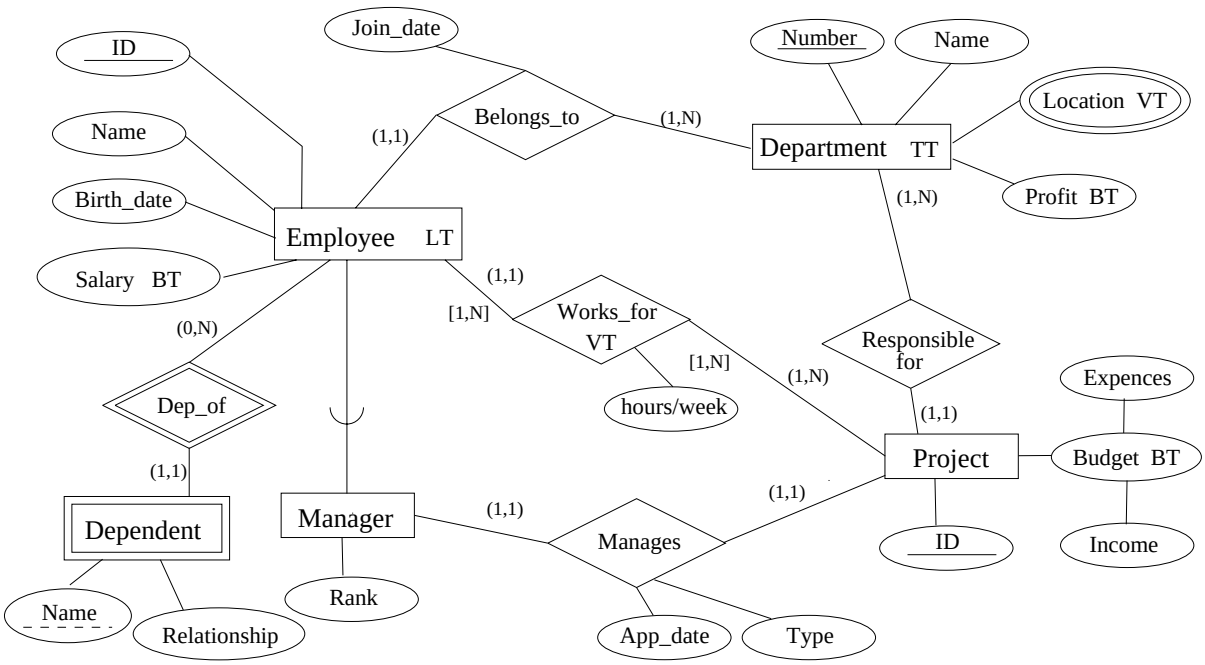


Figure 1: TIMEER Diagram of a company Database

Regular Entity Types

An entity type is represented graphically by a rectangle. Since all entities have an existence time and a transaction time aspect, it must be decided for each entity type in a TIMEER diagram whether or not to capture these temporal aspects of the entities in the database. Entity types that capture at least one temporal aspect are termed temporal.

If the lifespan or the transaction time of the entity type is captured, this is indicated by placing a LS or a TT, respectively, in the upper-right corner of the rectangle. If both the lifespan and the transaction time are captured, an LT is placed as before.

In Figure 1, we capture both the lifespan and the transaction time of the entity type Employee.

Weak Entity Types

A weak entity type is represented by a double rectangle. Weak entity types represent entities that are existence dependent on entities of other entity types and that cannot by themselves be uniquely identified. A weak entity type must therefore be related via an identifying relationship type (represented by a double diamond) to one or more regular entity types that are the owners of the weak entity type. Weak entity types may capture the same temporal aspects as regular entity types, and their temporal support are independent of the temporal support specified for the owner entity types.

In Figure 1, the entity type Dependent is weak and owned by the entity type Employee.

Attributes

Entities are characterized by their attributes. A single-valued attribute is represented by an oval, a multi-valued attribute is represented by a double oval, and a composite attribute is represented by an oval connected directly to the components of the composite attribute.

Every attribute has a valid-time and a transaction-time aspect. For each attribute the database designer must therefor decide to capture these aspects or not.

If the valid time is captured, a VT is placed to the right in the oval; if transaction time is captured, a TT is used. If both valid time and transaction time are captured, a BT (BiTemporal) is placed to the right in the oval. The components of a temporal composite attribute inherit the temporal specification of the composite attribute.

Both temporal and non-temporal entity types can have temporal and non-temporal attributes.

In Figure 1, we model that we want to capture the valid time and the transaction time of the Salary of an Employee.

It is required that the the database designer specifies a key for each entity type in the diagram. To indicate that a set of attributes represents 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 attribute.

Key attributes can be specified as temporal since the semantics of the TIMEER model ensures that all attribute types are snapshot reducible [15]. This ensures, for example, that a single-valued attribute, at any point in time, is single-valued. Thus, combining snapshot reducibility [15] of attribute types with the application of the conventional key constraint, we have that a key attribute uniquely identifies an entity at any single point in time.

Relationship Types

A relationship type is represented by a diamond. For each relationship type it has to be decided by the database designer whether or not to capture the temporal aspects of the relationships of the relationship type. If some temporal aspect is captured for a relationship type we term it temporal; otherwise, it is termed non-temporal.

Relationships can be seen from two very different points of view. We can perceive relationships among entities as attributes of the participating entities, or we can perceive relationships as things that exist in their own right. It is possible to adopt either point of view in TIMEER.

If the relationship is considered as an attribute, it must be specified whether to capture valid time, transaction time, or both. If valid time or transaction time is captured, this is indicated by placing a VT or a TT in the lower corner of the diamond. If both valid time and transaction time are captured, a BT is placed in the lower corner. The attribute view is the default view of a relationship, that is, a non-temporal relationship and a temporal relationship type supporting transaction time only is considered an attribute of the participating entities.

For each relationship type perceived as a thing that exists on its own right, it must be specified whether to capture the lifespan, or both transaction time and the lifespan. If the lifespan is captured this is indicated by an LS. If both lifespan and transaction time are captured, an LT is used.

In Figure 1, we model that we want to capture the valid time of the relationship Works_for between Employee and Project.

Snapshot Participation Constrains

The snapshot participation constraint of entity type E with respect to relationship type R is represented by placing min and max in parentheses next to the line connecting entity type E with relationship type R. The meaning of this is that at any point in time, each entity e of the entity type E will participate in at least min and at most max relationships r of R.

In Figure 1, the snapshot participation constraint of type Employee with respect to the relationship type Works_for is (1,1).

Lifespan Participation Constraint

The snapshot participation constraint described above constrains the participation of the entities at any given point in time. However, it is also useful to be able to constrain the participation of an entity in a relationship over the existence time of the entity.

The lifespan participation constraint of entity type E with respect to relationship type R is represented by placing min and max in square brackets next to the line connecting entity type E with relationship type R. The meaning of the lifespan participation constraint is that over all of time, each entity e of the entity type E will participate in at least min and at most max relationships r of R.

In Figure 1, the lifespan participation constraint of type Employee with respect to the relationship type Works_for is [1,N].

Superclasses and Subclasses

The TIMEER model offers support for specifying superclass/subclass relations. The syntax is as in the EER model.

A subclass inherits the attributes and the temporal support of its superclass. Is is not possible to change the temporal support of the inherited attributes, but it is possible to add attributes and to further expand the inherited temporal support of the class itself, i.e., if the inherited temporal support of a subclass is lifespan then the temporal support can be expanded to also include transaction time.

2.2 Definition of the Surrogate-Based Relational Target Model

The surrogate-based relational target model is inspired by Codd's RM/T model [3] and Snodgrass' Tuple Timestamp Scheme [15].

Domain of Attributes

In addition to the lexical domains, $\mathcal{D}_D = \{D_1, D_2, \ldots, D_n\}$, supported by the standard relational model, the surrogate-based relational target model supports a domain of surrogates [10] called the *E-domain*, and three time domains, namely the lifespan domain, $\mathcal{D}_{LS}$, the valid-time domain, $\mathcal{D}_{VT}$, and the transaction-time domain, $\mathcal{D}_{TT}$. A description of the time domains is given shortly. Surrogates are system-generated unique internal identifiers. Their values cannot be seen by and cannot be modified by the users of the model. Lifespan, valid-time, and transaction-time timestamps are represented as *chronon* [11] (defined later).

Attributes defined over the *E-domain* are called *E-attributes*, and attributes defined over time domains are called time attributes. The names of the time attributes are $\mathrm{LS_s}$, $\mathrm{LS_e}$, $\mathrm{VT_s}$, $\mathrm{VT_e}$, $\mathrm{TT_s}$, and $\mathrm{TT_e}$, where subscript *s* and *e* indicates start and end, respectively. The time attributes used to record existence time and valid time for attributes/relationships have values that are subject to user control, whereas the time attributes used to record transactions time are under system control and therefore cannot be changed by the users. The names of the attributes defined over the lexical domains come from the set $\mathcal{D}_A = \{A_1, A_2, \ldots, A_n\}$.

The database community generally agree that a discrete model of time is adequate, and we will adopt this view. The *real* time line is described by the baseline clock (BLC) [5], which provides the semantics of the timestamps. This time line is bounded in both ends, by the "Big Bang" as the lower bound and the "Big Crunch" as the upper bound. The model of time we use is thus a totally ordered, finite set of *chronons*, which is isomorphic to a finite subset of the natural numbers [6]. This may be seen as dividing the real time line into indivisible segments of equal size. A real-world time instant is expected to be much smaller than a chronon and is represented in the model by the chronon during which it occurs [6]. We will use c to denote chronons. The size of the chronons, called the granularity [4], can be specified explicitly.

For all the supported time dimensions, we adopt the above-mentioned model of time. For each dimension, there is several domains of times. Each domain for a time dimension is given by $D^{granule}_{dimension}$, where the size of the chronons defining the time domain is given by *granule*. For the valid-time and the lifespan domains, some chronons are expected to be in the future and some chronons are expected to be in the past. The chronon c_{now} denotes the chronon representing the current real-time instant, and UC ("until changed") is a special transaction-time marker. If a tuple in a relation recording transaction time has the value UC of its transaction-time endstamp this means that the tuple is current in the database.

A time interval is defined as the set of time instants between two instants. A time interval is represented by a sequence of consecutive chronons, represented in turn by a starting and an ending chronon. On the above-mentioned time domains, we thus define intervals $[c_i, c_j]^{granule}$ where c_i is the starting chronon, c_j is the terminating chronon, and the size of the chronons is *granule*. We let $[c_i, c_j]^{granule}_{vt}$, $[c_i, c_j]^{granule}_{tt}$, and $[c_i, c_j]^{granule}_{ls}$ denote intervals over the valid-time, transaction-time, and the lifespan domains, respectively.

We also define temporal elements [11] over time domains. A temporal element is a finite union of intervals and is denoted by $I^{granule} = [c_i, c_j]^{granule} \cup \cdots \cup [c_k, c_l]^{granule}$. Since any time domain $D^{granule}_{dimension}$ is discrete and finite, we can define a temporal element over this time domain as an element of the set $2^{D^{granule}_{dimension}}$. We let $I^{granule}_{vt}$, $I^{granule}_{tt}$, and $I^{granule}_{ls}$ denote temporal elements over the valid-time, transaction-time, and the lifespan domains, respectively.

Relations

The surrogate-based relational target model has two types of relations, *E-relations* and *A-relations*. An *E-relation* has a single *E-attribute* defined over the *E-domain* and, depending on the temporal support specified for the corresponding entity type in the TIMEER diagram, a number of time attributes. The set of tuples in an *E-relation* constitutes the existence list for the entity type. A surrogate value is never reused for a different object. For easier recognition, the names of E-attributes end with the character "ø", and the names of E-relations are the names of the entity types they represents. When the temporal support specified for a relationship type includes lifespan, this indicates that relationships of the type are considered to exist in their own right like entities. Consequently, E-relations represent such relationship types as well.

An *A-relation* represents attributes of entities. It has an *E-attribute* which references the E-relation representing the entity type for which it represents attributes. Single-valued, composite, as well as multi-valued attributes are represented in A-relations. Depending on the temporal support specified for an attribute, its representing A-relation also has a number of time attributes.

The relations below illustrate the E-relations and A-relations of this model. The E-relation *Employee* represents the entity type *Employee* from Figure 1, for which support for lifespan and transaction time are specified. The relation has two tuples representing one employee, and the granularity of the chronons is *day*. The tuples also indicate how we assume relations are updated. Specifically, we assume that a tuple is storing maximal transaction-time intervals so that each transaction-time chronon in the interval is associated with the same lifespan or valid-time interval. The A-relation *Employee_Salary* represents the temporal attribute *Salary* from Figure 1. The temporal support for this attribute includes valid time and transaction time.

Employee

employeeø	TT_S	TT_E	LS_S	LS_E
e1	1	8	10	15
e1	9	UC	5	20

Employee_Salary

employeeø	Salary	TT_S	TT_E	VT_S	VT_E
e1	7K	1	14	10	15
e1	7K	15	UC	5	20

Keys and Constraints

In a lexically-based data model, a primary key normally serves two roles: it is a lexical identifier, and it models existence. In a surrogate-based data model, lexical

identification and existence are separated. We ... to use the term primary key exclusively to indicate existence, and consequently only E-relations have primary keys; using the term primary key in an A-relation makes no sense because an A-relation does not model existence, other than the existence of the corresponding attributes of the entity type. The unique identifier of an A-relation is simply termed a key.

The Entity Integrity Constraint and the Referential Integrity Constraint from the RM/T model hold on any time-slice [11], that is, at any point in time or in any state. The Entity Integrity Constraint states that null-values are not allowed in *E-relations*. The Referential Integrity Constraint states that all surrogates referenced from an *A-relation* must exist in the corresponding *E-relation*.

The semantics of time attributes that we extend the relational model with are not built into the relational model, and are thus treated as regular (time-valued) attributes. They are used to record time intervals in the database. That is, we timestamp with pairs of time attributes, a start timestamp representing the starting chronon and an end timestamp representing the terminating chronon. We enforce the semantics of the temporal aspects, e.g., valid time, through the definition of a number of constraints on the database. However, as pointed out, these constraints are not enforced "automatically" by the data model, but must be enforced by explicit specifications (e.g., using assertions).

2.3 Definition of The Lexically-based Relational Target Model

The lexically-based relational target model is the relational model extended with timestamp attributes. Timestamp attributes are, as in the previously defined target model, used to capture the specified temporal support for the modeling constructs of the TIMEER model. The model of time and time domains for the time attributes used in this target model are as defined in Section 2.2.

3 Mapping From the TimeER Model to the Surrogate-based Relational Target Model

In this section we present the mapping of the TIMEER model to the surrogate-based relational target model, defined in Section 2.2. First, the mapping of entity types is presented. Second, the mapping of attribute types is presented. Third, the mapping of relationship types is presented. Finally, a set of temporal constraints that apply to the relations created in the mapping is presented. We have chosen to separate the presentation of the constraints from the presentation of the mappings in order no to disturb the reader's intuition and understanding of the mapping.

3.1 Mapping of Entity Types

First, the mapping of regular entity types is presented. Second, the mapping of weak entity types is presented. Finally, the mapping of entity types that participate in superclass/subclass relationships is presented.

Regular Entity Types

We first present the mapping of non-temporal regular entity types and then the mapping of temporal regular entity types.

A non-temporal entity type models that only a single state of each of the instances described by the entity type is to be recorded in the database. Since neither the existence time nor the transaction time of the instances of a non-temporal entity type are recorded in the database, we assume that an entity exists in the mini-world at least during the time its representing instance is recorded in the database. For each non-temporal regular entity type, an unary E-relation over the E-domain is created. The presence of a surrogate value in an E-relation indicates that the corresponding entity exists in the mini-world.

A consequence of not specifying any temporal support for an entity type is that when we logically delete an instance of the entity type from the database, *all* the information about that instance is logically deleted from the database. This implies that we have to logically delete from the database all attribute values, temporal and non-temporal, and the relationships, the instance participates in, temporal or non-temporal.

The difference between logical and physical deletion of information can be explained as follows. If information is physically deleted from the database, this information is actually erased and thus can not be accessed again. Logical deletion of information marks the information as deleted from the current state of the database, but the informations is kept in the database and can therefore be accessed by transaction-time timeslice operations. Logical and physical deletion are equivalent when no transaction time is recorded.

If the entity type has been specified as temporal then the user wants to store in the database either the existence time (lifespan) of the instances, the transaction time of the instances, or both. The lifespan of an instance records the time the corresponding real-world entity exists in the mini-world, and the timestamp recording the lifespan must be associated with the surrogate. If the user has specified transaction-time support for an entity type, the user wants to record the time during which the instance was current in the database. Again, the timestamp representing the transaction time of the instance must be associated with the surrogate. The E-relation for a temporal entity type is therefore expanded with time domains, two for the lifespan and/or two for transaction time.

When we expand the E-relation with timestamp attributes, we have to consider whether or not the E-attribute still uniquely identifies instances of the E-relation. If the temporal support for the entity type is lifespan only, this means that we explicitly store information about when an entity exists in the mini-world. We allow modifications of these timestamp attributes, but we do, of course, not allow modification of the E-attribute. However, since we allow an entity to be reborn in the database, e.g., a company rehires an employee, an E-attribute value can be associated with several pairs of timestamp attributes. This implies that the E-attribute does not uniquely identify the instances of the E-relation. The primary key of the E-relation is therefore expanded to include the timestamp attribute overlined in Figure 2 (a).

If the temporal support of the entity type is transaction time only, this means that we explicitly store information about when an instance is inserted into and logically deleted from the E-relation. We do not allow modifications of the timestamp attributes, since this is a transaction-time relation, but the same entity can be inserted and logically deleted from the database several times. This again implies that the E-attribute does not uniquely identify the instances of the E-relations, and the primary key of the E-relation must be expanded to include the timestamp attribute overlined in Figure 2 (b).

If the temporal support is both lifespan and transaction time, this means that we explicitly store information about when an entity exists in the mini-world and when this information was inserted into and logically deleted from the E-relation. Since we allow modification of the timestamp attributes for the lifespan and we never physically delete instances of the E-relation (it is a transaction-time relation), we can have multiple instances storing different lifespans for the entity. This implies that we have to expand the primary key of the E-relation with the timestamp attributes overlined in Figure 2 (c).

In subsequent diagrams, we will use the abbreviation in Figure 2 for entity types. In the diagrams, the temporal support will be indicated by an asterix and the (consequent) timestamp attributes in the relations with a T. So, for example, if the asterix denotes LS then the T in the relations denotes LS_S and LS_e.

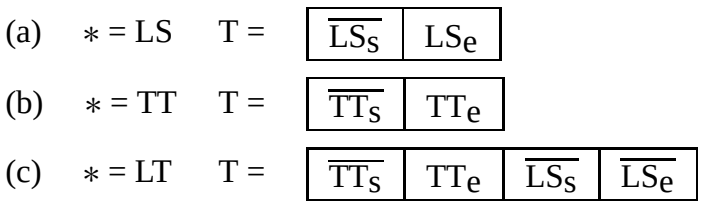


Figure 2: Abbreviation Used for Temporal Support for E-Relations

Mapping of Weak Entity Types

We consider temporal and non-temporal weak entity types in turn.

Weak entity types model entities from the mini-world that are existence dependent on entities of one or more other entity types and cannot by themselves be uniquely identified by lexical information. An identifying relationship type identifies the owner(s) of the weak entity type. Even though weak entity types model entities that are existence dependent on other entities and cannot be uniquely identified by themselves, they do exist. Therefore, an E-relation is created for each weak entity type in the diagram, i.e., the mappings of weak entity types and regular entity types are exactly the same. Constraints limiting the existence of instances in an E-relation representing a weak entity type to be dependent of the existence of instances in the E-relations representing the owners entity types are presented in Section 3.5.

The mapping of an identifying relationship type that associates the weak entity type with its owner will be given in Sections 3.3 and 3.4.

Mapping of Entity Types Participating in Superclass/Subclass Relationship Types

The syntax of the superclass/subclass relationship types in TIMEER is similar to the syntax in the EER model, and the mapping we present resembles the mapping from the EER model to the relational model presented in [7].

The semantics of the TIMEER model state that a subclass S of superclass C inherits all the properties described for C, including the temporal support. It is possible to extend, but not reduce, the temporal support of a subclass.

As in the EER model, it must be indicated if the superclass/subclass relationship type is total or optional, and if it is overlapping or is disjoint, i.e., there is a total of four combinations of participation constraints for superclass/subclass relationship types:

$$\left\{ \begin{array}{c} \text{total} \\ \text{optional} \end{array} \right\} \times \left\{ \begin{array}{c} \text{overlap} \\ \text{disjoint} \end{array} \right\}$$

In [7], four different options for mapping superclass/subclass relationship types are presented. Depending on the application at hand, one mapping will perform better than the others. We consider only the two options that reflect the semantics of superclass/subclass relationships the best. In the case where an entity type is both a subclass and a superclass, it is assumed that each superclass/subclass relationship type is considered separately in top-down order.

The first option will work for all combinations of participation constraints for superclass/subclass relationship types, while the second only works well if the participation constraint is both total and disjoint. Temporal constraints on the constructs in the surrogate-based relational target model resulting from the mapping are presented in Section 3.5.

Option 1 If the participation in the superclass/subclass relationship type is optional, then it is possible to have an instance of the superclass that is not an instance of any of the subclasses. It must be ensured that all such instances are represented.

For each entity type in the superclass/subclass relationship type, we create an E-relation as the Cartesian product over the E-domain and, if the entity types are specified as temporal, the time domains. The name of the E-attribute in an E-relation created for a subclass is the same as the name of the E-attribute of its superclass.

Example 1 The TIMEER diagram in Figure 3 describes a mini-world of employees. The entity type *Employee* has subclasses *Secretary, Technician,* and *Engineer*. The relations created as a result of the mapping are shown below the diagram. The attributes that are overlined in the relations indicate the primary keys of the relations, while attributes that are underlined constitute keys of the relations. The attributes that the user may have specified as a key for an entity type in the diagram are indicated by the symbol "u.k." ("pt.k." for weak entity types) in a relation. Foreign keys of relations are indicated by the symbol "f.k." following the attribute names. □

Option 2 If the participation in the superclass/subclass relationship type is total and disjoint then every instance of the superclass is an instance of one and only one subclass. For each subclass only, we create an E-relation as the Cartesian product over the E-domain and, if the entity types are specified as temporal, the time domains.

Example 2 The TIMEER diagram in Figure 4 describes a mini-world of Employees. □

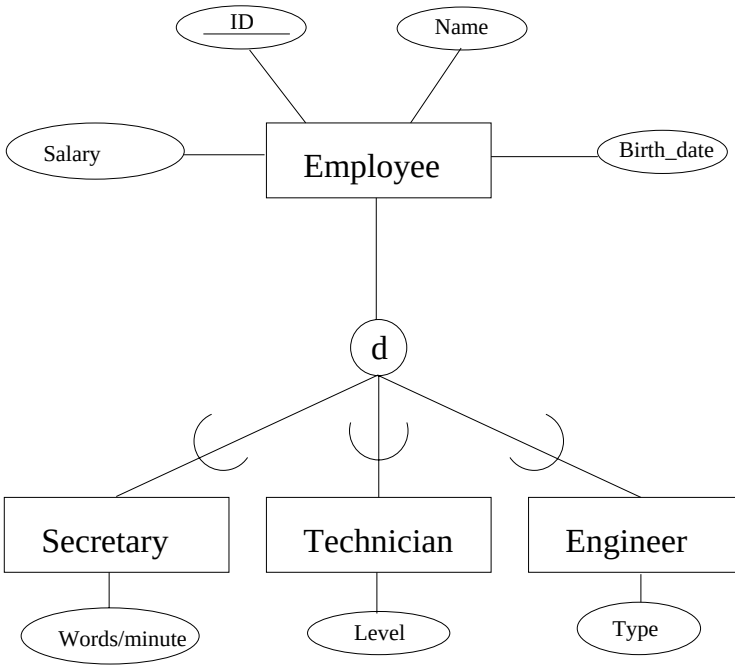
Even though it is not possible to specify a superclass/subclass relationship type as temporal, there is implicit temporal support for superclass/subclass relationships. The temporal information about the relationship itself can be deduced from the temporal information recorded by the involved entity types.

3.2 Attributes of Entity Types

In this subsection the mapping of the attributes of entity types is presented. With a few exceptions, the mapping of attributes of relationship types is the same and will be presented in Sections 3.3 and 3.4.

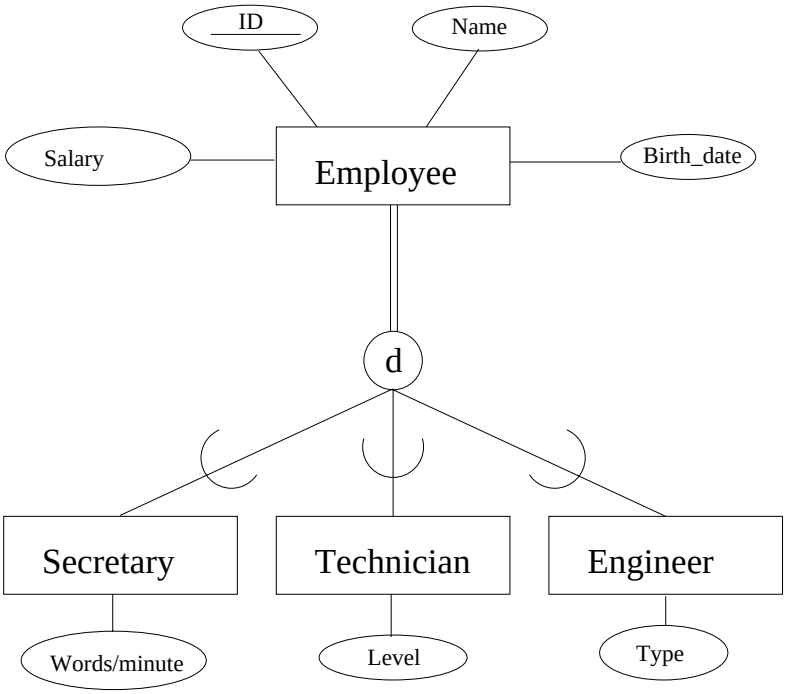
Non-Temporal Attribute Types

In this case, the variation of the attribute values over time is not recorded. The mapping in this case is similar to the mapping described in [7], with the exception



Employee	*Employee_ID_Name_Birth_date_Salary*
$\overline{employee\emptyset}$	*employeeø* f.k. | ID u.k. | Name | Birth_date | Salary
Secretary	*Secretary_Words/minute*
$\overline{employee\emptyset}$	*employeeø* f.k. | Words/minute
Technician	*Technician_Level*
$\overline{employee\emptyset}$	*employeeø* f.k. | Level
Engineer	*Engineer_Type*
$\overline{employee\emptyset}$	*employeeø* f.k. | Type

Figure 3: Mapping of Superclass/Subclass Relationships With Disjoint and Optional Participation Involving only Non-Temporal Entity Types



Secretary — *Secretary_ID_Name_Birth_date_Salary_Words/minute*

$\overline{employee\emptyset}$	*employeeø* f.k.	ID u.k.	Name	Birth_date	Salary	Words/minute

Technician — *Technician_ID_Name_Birth_date_Salary_Level*

$\overline{employee\emptyset}$	*employeeø* f.k.	ID u.k.	Name	Birth_date	Salary	Level

Engineer — *Engineer_ID_Name_Birth_date_Salary_Type*

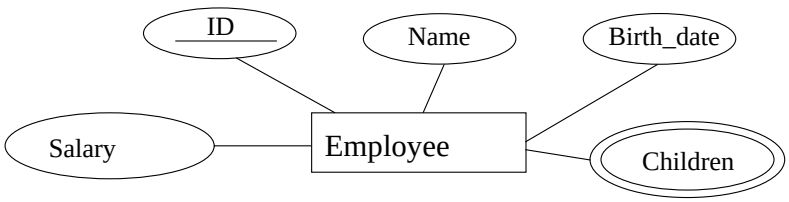
$\overline{employee\emptyset}$	*employeeø* f.k.	ID u.k.	Name	Birth_date	Salary	Type

Figure 4: Mapping of Superclass/Subclass Relationships With Disjoint and Total Participation Involving only Non-Temporal Entity Types

that our target model is surrogate-based. An A-relation is created as the Cartesian product over the E-domain and the domains of all the non-temporal single-valued and non-temporal composite attributes of the entity type. The E-attribute is the key of the A-relation. The E-attribute is also a foreign key referencing the E-relation for the entity type, the attributes are associated with. Attributes the user might have specified as a lexical key in the diagram constitute a lexical key in the A-relation. For each multivalued attribute, a new A-relation is created over the E-domain and the domain of the multivalued attribute. The key of this A-relation is the E-attribute and the attribute value in combination.

The reason why we choose to represent all the simple non-temporal attributes of an entity type in one A-relation is that such an A-relation is likely to form a maximal non-BCNF violating unit if the diagram is well designed. Additional decomposition could certainly be done, but would lead to unnecessary fragmentation (over-normalization). The reason for the creation of a separate A-relation for each multivalued attribute is that the same entity may be associated with several values in the A-relation representing the multi-valued attribute and that these values together form the set of values for the multivalued attribute of the entity.

Example 3 The TIMEER diagram in Figure 5 describes a mini-world of employees. The entity type *Employee* has attributes *ID, Name, Birth_date*, *Salary*, and *Children*. The mapping of this diagram results in the relations illustrated below the diagram. We create an E-relation named *Employee*, an A-relation *Employee_ID_Name_Birth_date_Salary* and an A-relation *Employee_Children*. □



Employee

employeeø

Employee_ID_Name_Birth_date_Salary

employeeø f.k.	ID u.k.	Name	Birth_date	Salary

Employee_Children

employeeø f.k.	Child_name

Figure 5: Mapping of Non-Temporal Entity Type With No Temporal Attributes

Temporal Attribute Types

We create an A-relation for each temporal attribute of an entity type. The design of the A-relations is dependent on the type of attribute and the type of temporal

support specified for the attribute. In general, the key of the A-relation is the E-attribute concatenated with one or more of the timestamp attributes, depending on the specified temporal support. We will be using the abbreviation schema shown in Figure 6 for describing which timestamp attributes to include in an A-relation for the temporal support of an attribute indicated by an asterix in the diagrams. In Figure 6 we have underlined the time attributes that are to be concatenated to the E-attribute to constitute the key of A-relations representing temporal attributes.

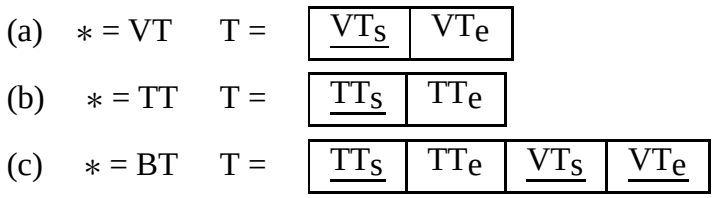


Figure 6: The Abbreviation Scheme Used for Temporal Support for A-Relations

Temporal Single-Valued Attribute

In the case where a temporal attribute is single-valued, we know from the semantics of the TIMEER model that at any point in time, this attribute will have at most one atomic value. Depending on the temporal support specified for the attribute, the following may be stored in the database: the times an attribute value is valid in the mini-world, indicated by valid-time support; the times an attribute value is recorded in the database, indicated by transaction-time support; or both times, indicated by bitemporal support. For each temporal single-valued attribute of an entity type, a separate A-relation is created as the Cartesian product over the E-domain, the domain of the attribute, and the time domains.

The choice of creating a separate A-relation for each temporal single valued attribute is based on the fact that nothing in the TIMEER diagram indicates, even if the temporal support is identical, that two different temporal attributes follow the same update pattern. This choice ensures that attribute values are not is repeated in the database, and unnecessary redundancy is avoided.

Example 4 The diagram in Figure 7 results in the relations illustrated below the diagram. We create an E-relation named *Employee*, an A-relation named *Employee_ID_Name_Birth_date* for the non-temporal attributes, and an A-relation named *Employee_Salary* for the temporal attribute. Depending on the temporal support specified for the attribute, one of three different A-relations may result from the substitution of the abbreviation "T" in the A-relation by the timestamp attributes introduced in Figure 6. □

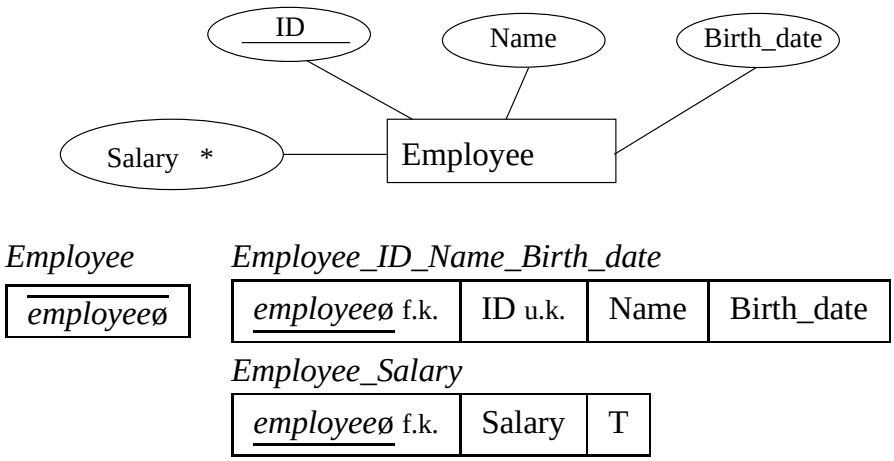


Figure 7: Mapping of Non-Temporal Entity Type With a Temporal Single-Valued Attribute

Temporal Composite Attribute

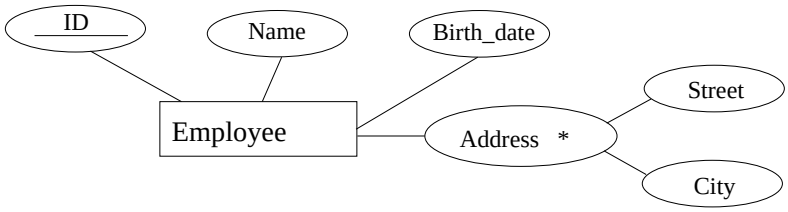
When a temporal attribute is composite, it is assumed that the components change synchronously, since the composite is considered to be one element. Also, it is assumed that the temporal support is the same for all its components since the temporal support for a composite has to be specified for the composite rather than for the components. This implies that all the components of the composite should be mapped into the same A-relation. For each temporal composite attribute of an entity type, an A-relation is created as the Cartesian product over the E-domain, the domains of the components of the composite attribute, and the temporal domains. The key of the A-relation is the E-attribute concatenated with one or more of the timestamp attributes, depending on the specified temporal support (see Figure 6).

Example 5 The mapping of the diagram in Figure 8 results in the relations illustrated below the diagram. □

Temporal Multivalued Attribute

If a temporal attribute is multivalued then its set of values changes over time, and this variation is recorded in the database. This implies that the changing set of values is timestamped in the database as the temporal multivalued attribute changes its value.

For each temporal multivalued attribute of an entity type, a new A-relation is created as the Cartesian product over the E-domain, the domain of the attribute, and the appropriate time domains. The key of the A-relation is the concatenation of the E-attribute, the multivalued attribute, and one or more of the timestamp attributes, depending on the specified temporal support. The constraints enforcing the complicated meaning of the above are presented in Section 3.5.



Employee

employeeø

Employee_ID_Name_Birth_date

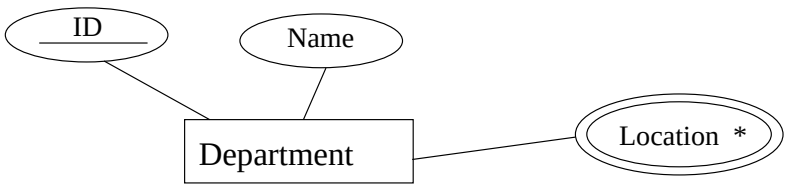
employeeø f.k.	ID u.k.	Name	Birth_date

Employee_Address

employeeø f.k.	Street	City	T

Figure 8: Mapping of Non-Temporal Entity Types With Temporal Composite Attributes

Example 6 In the diagram in Figure 9, the entity type *Department* has non-temporal attributes *ID* and *Name*, and temporal multivalued attribute *Location*. This diagram results in the relations illustrated next to the diagram. We create an E-relation *Department*, an A-relation *Department_ID_Name* for the non-temporal attributes, and an A-relation *Department_Location* for the temporal multivalued attribute. □



Department

departmentø

Department_ID_Name

departmentø f.k.	ID u.k.	Name

Department_Location

departmentø f.k.	Location	T

Figure 9: Mapping of Non-Temporal Entity type With a Temporal Multivalued Attribute

3.3 Non-Temporal Relationship Types

In this section, we describe the mappings of non-temporal binary relationship types and n-ary relationship types. In TIMEER a non-temporal relationship type is perceived as an attribute of the participating entity types, see Section 2.1. When relationship types are considered attributes of the participating entity types, they are

then represented like other attributes of the entities, i.e., in A-relations. Relationship types are represented in A-relations through the use of E-attributes as foreign keys.

Non-Temporal Binary Relationship Types

A binary non-temporal relationship type is considered to be a non-temporal attribute of the participating entity types. This view is similar to the view taken in the EER model, and we reuse the ideas from the transformation algorithm from the EER model to the relational model. Specifically, we do not always create a new A-relation to represent the relationship type, but in most cases extend one of the A-relations that already record the non-temporal attributes of the participating entity types. When to extend which A-relation or when to create a new A-relation is determined by the snapshot participation constraints specified for the entity types in the diagram.

In Table 1, let A and B be the two entity types participating in a non-temporal binary relationship type. The table displays the 16 possible, qualitatively different, combinations of snapshot participation constraints that can be specified for the involved entity types. For each combination, it is described when to extend an existing A-relation and when to create a new A-relation. Furthermore, it is described which of the participating entity types is the better candidate for having its A-relation expanded with the E-attribute of the other participating entity type as a foreign key. If the relationship type is an identifying relationship type then the E-attribute of the owner entity type is further marked with an "ow." in order to indicate the owners of a weak entity type, in the A-relation representing the relationship type.

		B			
		(1,1)	(0,1)	(0,N)	(1,N)
A	(1,1)	A or B	A	A	A
	(0,1)	B	A or B	A	A
	(0,N)	B	B	New relation	New relation
	(1,N)	B	B	New relation	New relation

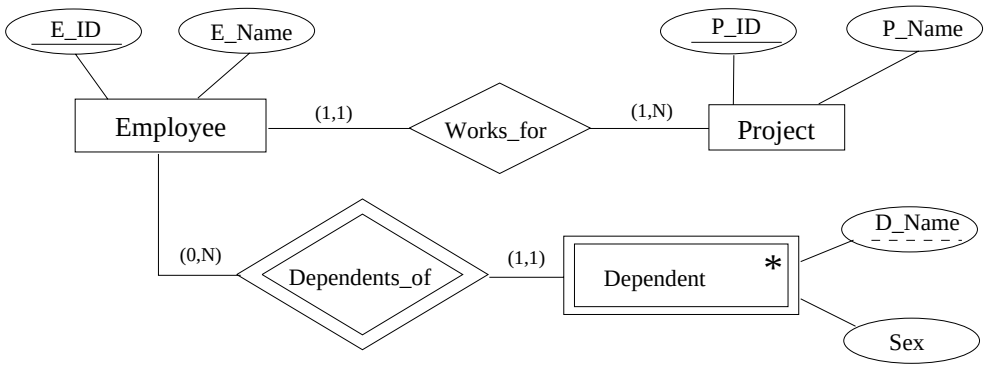
Table 1: Mapping Non-Temporal Binary Relationship Types

In the case where we extend an already existing A-relation and the non-temporal relationship type has non-temporal attributes, we further extend the chosen A-relation to include the non-temporal attributes of the relationship type.

If a new A-relation has to be created then it is created as the Cartesian product over the E-domains of the participating entity types, and the domains of the non-temporal attributes, if any, of the relationship type. The key of the new A-relation is the E-attributes concatenated.

In the case where the relationship type has temporal attributes, we proceed as already described in Section 3.2. The key of the A-relation is the concatenation of the E-attributes and one or more of the timestamp attributes, depending on the specified temporal support.

Example 7 In the diagram in Figure 10, entity type *Employee* has non-temporal attributes *E_ID* and *E_Name*. Entity type *Project* has non-temporal attributes *P_ID* and *P_Name*. The temporal, weak entity type *Dependent* has non-temporal attributes *D_Name* and *Sex*. The non-temporal relationship type *Works_for* describes that information about which employees work for which projects is stored in the database. The non-temporal identifying relationship type *Dependent_of* captures which dependent is dependent on which employee. The mapping to the relational model is as described by the relations below the diagrams. Since the snapshot participation constraint for *Employee* is (1,1) and the snapshot participations constraint for *Project* is (1,N), we expand the A-relation representing the non-temporal attributes for *Employee* with the E-attribute of *Project*. □



Employee

$\overline{employee\emptyset}$

Employee_EID_EName

employeeø f.k.	E_ID u.k.	E_Name	*projectø* f.k.

Project

$\overline{project\emptyset}$

Project_PID_PName

projectø f.k.	P_ID u.k.	P_Name

Dependent

$\overline{dependent\emptyset}$	T

Dependent_DName_Sex

dependentø f.k.	D_Name pt.k.	Sex	*employeeø* f.k., ow.

Figure 10: Mapping of Two Non-Temporal Relationship Types

Non-Temporal n-ary Relationship Types

If the number of participating entity types in a relationship type is n where $n \geq 3$ then we always create an A-relation as the Cartesian product over the surrogate domains of the participating entity types. If the snapshot participation constraint is

(0,1) or (1,1) for one or more of the participating entity types then the E-attribute of one of these is chosen as key for the A-relation; otherwise, the key of the A-relation is the E-attributes concatenated. If the relationship type is an identifying relationship type then the E-attribute of the owner entity type is further marked with an "ow." in the relations.

If the relationship type has non-temporal attributes, the above-mentioned A-relation is extended with the domains of these attributes. In the case where the relationship type has temporal attributes, we again proceed as described in Section 3.2. The key of the A-relation is then the concatenation of all or some of the E-attributes and one or more of the timestamp attributes, depending on the specified temporal support.

3.4 Temporal Relationship Types

In the TIMEER model, a temporal relationship type can be considered either to be an attribute of the participating entity types or to be something that exists in its own right. In the case where the temporal support for the relationship type is specified to capture valid time, transaction time, or both, it is considered an attribute. If, on the other hand, the temporal support for the relationship type is specified to include existence time only, or transaction time and existence time, it is considered to exist in its own right.

Temporal Binary Relationship Types, Attribute View

In this section we describe the case where the relationship type is considered an attribute of the participating entity types and is itself temporal, i.e, the relationship type is considered a temporal attribute. We create an A-relation as the Cartesian product over the E-domains of the participating entity types and the appropriate time domains. The key of the A-relation can be determined from the snapshot participation constraints and the temporal support specified for the relationship type. In Table 2, it is described which of the E-attributes should be part of the key, concatenated with timestamp attributes as specified in Figure 6. If the relationship type is an identifying relationship type then the E-attribute of the owner entity type is further marked with an "ow." in the relations.

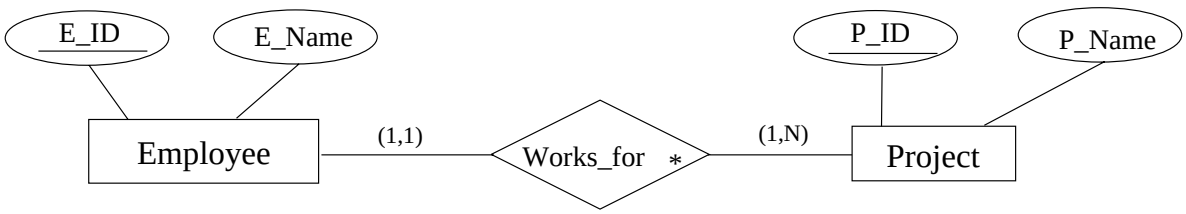
The mapping of the attributes of temporal relationship types is almost identical to the mapping of attributes of entity types described in Section 3.2 (temporal and non-temporal). There are two differences: first, the E-attributes of both the participating entity types appears in the resulting A-relation; second, the E-attributes that is to be part of the keys is again determined by the snapshot participation constraint, and the temporal support of the attribute is temporal, for the participating

		B			
		(1,1)	(0,1)	(0,N)	(1,N)
A	(1,1)	aø or bø	aø	aø	aø
	(0,1)	bø	aø or bø	aø	aø
	(0,N)	bø	bø	aø + bø	aø + bø
	(1,N)	bø	bø	aø + bø	aø + bø

Table 2: Keys For A-Relations Representing Temporal Relationship Types Perceived as Attributes

entity types, that is, the chose of E-attributes that should be part of the key is as described above.

Example 8 The diagram in Figure 11 records information about the *ID* and *Name* for *Employee* and *Project*. We also register information about which employees *Works_for* which project and how this relationship varies over time. The mapping to the relational model is as described by the relations shown below the diagram. As earlier the T in the A-relation should be replaced by the proper timestamp attributes following the scheme for temporal attributes (Figure 6). □



Employee | *Employee_EID_EName*

employeeø

employeeø f.k.	E_ID u.k.	E_Name

Project | *Project_PID_PName*

projectø

projectø f.k.	P_ID u.k.	P_Name

Works_for

employeeø f.k.	*projectø* f.k.	T

Figure 11: Mapping of Temporal Relationship Types Involving Two Non-Temporal Entity Types

Temporal Binary Relationship Types, Existence View

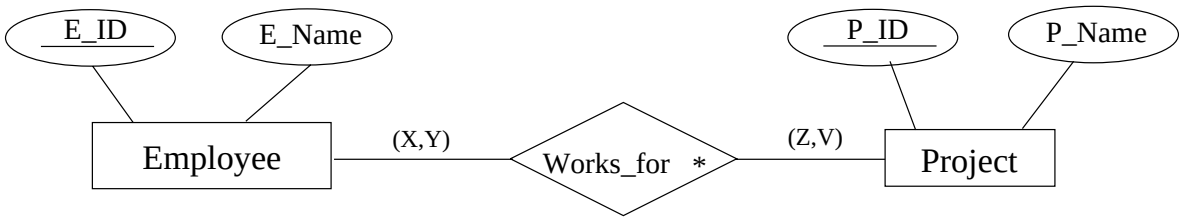
If we consider a relationship as a thing that exists in its own right, we have to assign surrogates to its instances. We therefore create an E-relation as the Cartesian prod-

uct over the surrogate domain and the time domains specified for the relationship type. The primary key of the E-relation is the E-attribute concatenated with the timestamp attributes, overlined in Figure 2.

In this case, the participating entity types of the relationship type can be seen as attributes of the relationship type. Therefore an A-relation is created as the Cartesian product over the surrogate domain of the relationship type and the surrogate domains of the participating entity types. The key of the relation is the E-attribute representing the relationship type. If the relationship type has non-temporal attributes, the A-relation is expanded with the domains of the non-temporal attributes. If the relationship type is an identifying relationship type then the E-attribute of the owner entity type is further marked with an "ow." in the relations.

If the relationship type has temporal attributes, an A-relation is created for each temporal attribute as the Cartesian product over the surrogate domain, the attributes domain, and the time domains. The key of the A-relation is the E-attribute concatenated with the proper timestamp attributes.

Example 9 Given the diagram in Figure 12, the mapping to the relational model is as described by the relations below the figure. The T in the E-relation should be replaced by the proper timestamp attributes following the scheme for E-relations (see Figure 2) with the restriction that the temporal support must include lifespan. □



Employee

employeeø

Employee_EID_EName

employeeø f.k.	E_ID u.k.	E_Name

Project

projectø

Project_PID_PName

projectø f.k.	P_ID u.k.	P_Name

Works_for

works_forø	T

Works_for_Employee_Project

works_forø f.k.	*employeeø* f.k.	*projectø* f.k.

Figure 12: Mapping of a Temporal Relationship Type Supporting Lifespan Involving Two Non-Temporal Entity types

Temporal n-ary Relationship Types

If a temporal relationship type is n-ary and the temporal support indicates that the relationship type is considered to be an attribute of the participating entity types, i.e., valid time, transaction time, or bitemporal, then an A-relation is created as the Cartesian product over the surrogate domains of the participating entity types and the time domains. The choice of key for the A-relation is depending on the temporal support specified for the relationship type and the snapshot participation constraints specified for the participating entity type. The choice of which domains of the E-attribute(s) that is to be part of the key concatenated with the proper timestamp attributes is described in Section 3.3. If the relationship type is an identifying relationship type then the E-attribute of the owner entity type is further marked with an "ow." in the relations.

If the relationship type has non-temporal attributes, a new A-relation is created as the Cartesian product over the surrogate domains of the participating entity types and the domains of all the non-temporal attributes. The key of the relation is the same set of E-attributes chosen for the A-relation representing the relationship.

If the relationship type has temporal attributes then for each temporal attribute an A-relation is created as the Cartesian product over the surrogate domains of the participating entity types, the domain of the attribute, and the time domains. Depending on the temporal support specified for the attribute, the key of the A-relation is again the same set of E-attributes chosen for the A-relation representing the relationship concatenated with the proper timestamp attributes.

If the temporal support of the relationship type includes lifespan, indicating that the relationship type is considered as something that exists in its own right then we proceed as described in Sections 3.4.

3.5 Temporal Constraints

In the previous sections the mapping from the conceptual TIMEER model to the surrogate-based relational target model was presented without specifying the many constraints that apply to the relations created by the mapping. This section presents all the temporal constraints that apply to these relations. Since the surrogate-based relational target model does not automatically enforce these constraints, they must be enforced explicitly using, e.g, assertions. Constraints that are non-temporal are inherited from the surrogate-based relational target model.

Within A-Relations

The following temporal constraints apply to A-relation recording temporal information. Constraints 3.1 – 3.3 apply to A-relations representing simple attributes and to temporal relationship types perceived as attributes. They do not apply to

A-relations representing temporal multivalued attributes. Constraints applying to those A-relations will be described later in this section.

It must be enforced that the information recorded by the A-relations is snapshot reducible [15], i.e., snapshots of the database described using the temporal ER diagram constructs are the same as those described by the corresponding snapshot ER diagram where all temporal constructs are replaced by their snapshot counterparts. As an example of snapshot reducibility, a temporal single-valued attribute is snapshot reducible to a (snapshot) single-valued attribute if the temporal single-valued attribute is single valued at each point in time. For A-relations recording valid time only, no two tuples of the A-relation containing the same E-attribute, or the same set of E-attributes, can have overlapping valid-time intervals.

Constraint 3.1 (Snapshot Reducibility of Valid-Time A-Relations) Let K denote the set of E-attributes contained in A-relation R recording valid time only; then

$$\forall r_1, r_2 \in R(r_1.K = r_2.K \Rightarrow [r_1.VT_s, r_1.VT_e] \cap [r_2.VT_s, r_2.VT_e] = \emptyset) \qquad \square$$

This constraint is necessary to ensure that no valid-time timeslice on the A-relation will return more than one value for the attribute.

If the temporal information recorded is transaction time only then a snapshot reducibility constraint for transaction time is enforced if the transaction-time start timestamp of a tuple in an A-relation is the time of the insertion of the tuple into the database and the transaction-time end timestamp is the time of the logical deletion of the tuple from the database. Since, as time passes, the transaction time increases no two tuples in the relation with the same value for the E-attribute, or the same set of E-attributes, have overlapping transactions-time intervals.

Constraint 3.2 (Snapshot Reducibility of Transaction-Time A-Relations) Let K denote the set of E-attributes contained in A-relation R recording transaction time only; then

$$\forall r_1, r_2 \in R(r_1.K = r_2.K \Rightarrow [r_1.TT_s, r_1.TT_e] \cap [r_2.TT_s, r_2.TT_e] = \emptyset) \qquad \square$$

Bitemporal A-relations must also be snapshot reducible. This means that no two tuples associated with the same E-attribute, or set of E-attributes, current at a given transactions-time point (the intersection of the transaction-time intervals is non-empty) have overlapping valid-time intervals.

Constraint 3.3 (Snapshot Reducibility of Bitemporal A-Relations) Let K denote the set of E-attributes contained in A-relation R, recording both valid time and transaction time; then

$$\begin{aligned}\forall r_1, r_2 \in R((r_1.K = r_2.K \wedge [r_1.TT_s, r_1.TT_e] \cap [r_2.TT_s, r_2.TT_e] \neq \emptyset) \\ \Rightarrow [r_1.VT_s, r_1.VT_e] \cap [r_2.VT_s, r_2.VT_e] = \emptyset)\end{aligned} \qquad \square$$

Example 10 This example presents a bitemporal A-relation that satisfies Constraint 3.3. We assume that the current time is 21. The relation records the salary

history for employees in a company. If the valid-time end timestamp of a tuple is *NOW*, this means that the attribute value recorded by the tuple remains valid until the current time, see the fourth tuple.

employeeø	Salary	TT_S	TT_e	VT_S	VT_e
e1	7K	1	14	10	15
e2	6K	10	18	10	14
e1	7K	15	*UC*	5	25
e2	8K	19	*UC*	15	*NOW*

□

Some A-relations represent temporal multivalued attributes of either entity types or relationship types. As mentioned, the constraints defined above do not apply to these A-relations. The reason is that these record set valued attributes and therefore sets of values changes over time. This implies that sets of values are timestamped in the database whenever a temporal multivalued attribute changes its value. We must therefore require, for these A-relations, that the tuples that belong to the same value set record the same temporal information. In order to ensure this, we define a *Value Set* and redefine the constraints 3.1 through 3.3, replacing them with constraints 3.4 through 3.6. These constraints apply to value sets in an A-relation instead of to individual tuples.

Definition 1 (Value Set) Let R be an A-relation representing a temporal multivalued attribute M. Let K denote the set of E-attributes and let T be the set of timestamp attributes contained in R. Let r_1 and r_2 be two tuples in R. r_1 and r_2 belongs to the same value set S in R if and only if r_1 and r_2 have matching values of the E-attributes and the timestamp attributes.

$$r_1, r_2 \in R \wedge r_1.K = r_2.K. \wedge r_1.T = r_2.T \Leftrightarrow r_1, r_2 \in S$$ □

Example 11 The following A-relation exemplifies both how we expect a multivalued attribute to be updated and value sets. The A-relation records the dependent children of employees, modeled using a bitemporal multi-valued attribute.

employeeø	Depend_children	TT_S	TT_e	VT_S	VT_e	
e1	Jens	1982	1985	1982	1985	S_1
e1	Jens	1986	1989	1986	1988	S_2
e1	Ulla	1986	1989	1986	1988	S_2
e1	Jens	1990	*UC*	1989	*NOW*	S_3
e1	Ulla	1990	*UC*	1989	*NOW*	S_3
e1	Peter	1990	*UC*	1989	*NOW*	S_3

□

To ensure the snapshot reducibility of temporal multivalued attributes supporting valid time only it must be ensured that no two different value sets have overlapping valid-time timestamps.

Constraint 3.4 (Snapshot Reducibility of Valid-time Multivalued Attributes) Let R be an A-relation representing a multivalued attribute recording valid time only. Let K denote the set of E-attributes contained in R, and let S_1 and S_2 be two value sets in R; then

$$\forall r_1, r_2 \in R((r_1 \in S_1 \wedge r_2 \in S_2 \wedge S_1 \neq S_2 \wedge r_1.K = r_2.K) \\ \Rightarrow [r_1.VT_s, r_1.VT_e] \cap [r_2.VT_s, r_2.VT_e] = \emptyset) \qquad \square$$

It must also be required that a multivalued attribute supporting transaction time only is snapshot reducible by requiring that no two value set have overlapping transaction-time intervals.

Constraint 3.5 (Snapshot Reducibility of Transaction-time Multivalued Attributes) Let R be an A-relation recording transaction time only and representing a multivalued attribute. Let K denote the set of E-attributes contained in R, and let S_1 and S_2 be two value sets in R; then

$$\forall r_1, r_2 \in R((r_1 \in S_1 \wedge r_2 \in S_2 \wedge S_1 \neq S_2 \wedge r_1.K = r_2.K) \\ \Rightarrow [r_1.TT_s, r_1.TT_e] \cap [r_2.TT_s, r_2.TT_e] = \emptyset) \qquad \square$$

The following constraint ensures the snapshot reducibility of bitemporal multivalued attributes.

Constraint 3.6 (Snapshot Reducibility of Bitemporal Multivalued Attributes) Let R be an A-relation recording valid time and transaction time representing a multivalued attribute. Let K denote the set of E-attributes contained in R, and let S_1 and S_2 be two value sets in R; then

$$\forall r_1, r_2 \in R((r_1 \in S_1 \wedge r_2 \in S_2 \wedge S_1 \neq S_2 \wedge r_1.K = r_2.K \wedge \\ [r_1.TT_s, r_1.TT_e] \cap [r_2.TT_s, r_2.TT_e] \neq \emptyset) \\ \Rightarrow [r_1.VT_s, r_1.VT_e] \cap [r_2.VT_s, r_2[VT_e] = \emptyset)) \qquad \square$$

Between A-Relations and E-Relations

In this section we present the temporal constraints that must hold between A-relations representing either temporal attributes, temporal relationship types supporting valid time, transaction time, or both, and temporal attributes of these relationship types and the E-relations that represent the involved entity types.

If the temporal information recorded by an A-relation includes transaction time, then the referential integrity constraint must be redefined to require that at all time all current tuples of the A-relation must have a value for the E-attribute(s) (f.k.) that is current in some E-relation(s). All tuples of non-temporal relations and relations recording lifespan/valid time only are current, whereas possibly only a subset of the tuples in relations recording transaction time only or both lifespan or valid time and transaction time are current (all the tuples that have *UC* as transaction-time

end timestamp). The above fact implies that we can apply the conventional referential integrity constraint, unchanged, in the case where both the A-relation and the E-relation are either non-temporal or record lifespan and valid time only, but that we have to redefine the the referential integrity constraint in all other cases. The table below indicates which constraints apply depending on the temporal support of the A-relation, (A), and E-relations, (E), respectively.

		A	
		None or VT	TT or BT
E	None or LS	Conventional Referential Integrity Constraint	Constraint 3.7
	TT or LT	Constraint 3.8	Constraint 3.9

The three temporal referential integrity constraints are formally defined as follows:

Constraint 3.7 (Temporal Referential Integrity Constraint 1) Let $R_i,\ i = 1, \ldots, n$, be the E-relations *not* recording transaction time, referenced by A-relation R_A recording transaction time; then

$$\forall R_i \forall r \in R\_A(r.TT_e = UC \Rightarrow (\exists s \in R_i(s.E\emptyset = r.E\emptyset))) \quad \square$$

Constraint 3.8 (Temporal Referential Integrity Constraint 2) Let $R_i,\ i = 1, \ldots, n$, be the E-relations, recording transaction time referenced by A-relation R_A *not* recording transaction time; then

$$\forall r \in R\_A \forall R_i\ \exists s(s.E\emptyset = r.E\emptyset \wedge s.TT_e = UC) \quad \square$$

Constraint 3.9 (Temporal Referential Integrity Constraint 3) Let $R_i,\ i = 1, \ldots, n$, be the E-relations recording transaction time referenced by A-relation R_A recording transaction time; then

$$\forall R_i \forall r \in R\_A(r.TT_e = UC \Rightarrow (\exists s \in R_i(s.E\emptyset = r.E\emptyset \wedge s.TT_e = UC))) \quad \square$$

The redefinition of the referential integrity constraints are necessary since tuples from relations with transaction-time timestamps are never physically deleted, only logically, whereas tuples of valid-time relations (if the tuples was inserted by an error) and snapshot relations (relations without timestamp attributes) are physically deleted. Also note the above defined constraints only ensures that a referenceable value of an E-attribute exists in the E-relation. Constraints on the time intervals recorded by the tuples are defined next.

A set of temporal constraints must hold between time intervals recorded of the tuples in the A-relations and the time intervals recorded of the tuples in the E-relation when the tuples have the same value of the E-attributes. The purpose of the constraints is to ensure that attributes of, and relationships among, temporal entities cannot be associated with time intervals for which the entities do not exist or are not registered in the database. There are 16 different combinations of temporal

support to consider and the table below indicates which constraint, if any, applies to a specific combination of temporal support between an A-relation (columns) and an E-relation (rows) referenced by the A-relation.

		A			
		None	VT	TT	BT
E	None	Observation 3.1	Observation 3.1	Observation 3.1	Observation 3.1
	LS	Observation 3.2	Constraint 3.10	Observation 3.3 Constraint 3.11	Constraint 3.12, Observation 3.3
	TT	Observation 3.2	Observation 3.4	Constraint 3.13	Constraint 3.13 Observation 3.4
	LT	Observation 3.2	Constraint 3.14	Constraint 3.15	Constraint 3.16

Observation 3.1 If the tuples in an E-relation does not record any temporal information, no temporal constraints can be enforced on the temporal information, if any, recorded by the tuples in an A-relation referencing the E-relation.

Observation 3.2 If the tuples in an A-relation does not record any temporal information, no temporal constraints can be enforced on the temporal information, if any, recorded by the tuples in an E-relation referenced by the A-relation.

If the temporal support recorded by the tuples in an E-relation is lifespan only, and the temporal support recorded by the tuples in an A-relation referencing the E-relation is valid time only then the valid-time interval recorded by each tuple of the A-relation must be included in the lifespan interval recorded by the tuple in the E-relation with the same value of the E-attribute. This is expressed by the following constraint.

Constraint 3.10 (Valid Time Inclusion) Let R be an A-relation recording valid time only, let S be an E-relation referenced by R recording lifespan only; then

$$\forall r \in R\ \exists s \in S(r.E\text{ø} = s.E\text{ø} \wedge [s.LS_s, s.LS_e] \supseteq [r.VT_s, r.VT_e]) \qquad \square$$

The above constraint ensures that the tuples in an A-relations representing temporal attributes only record valid-time intervals that are included in the lifespan interval recorded by the tuple the E-relation with the same value of the E-attribute. It also ensures that the valid-time intervals recorded by the tuples in a A-relation representing a relationship type recording valid time are included in the non-empty intersection of the lifespan intervals recorded by the participating temporal entities supporting lifespan only, since it must hold for each E-relation referenced by the A-relation.

Example 12 The A-relation *Employee_Salary*, recording the development in employee salaries, satisfies the valid-time inclusion constraint (3.10) with respect to the E-relation *Employee*, recording the lifespans of the employees.

Employee

employeeø	LS_S	LS_e
e1	5	20
e2	10	*NOW*

Employee_Salary

employeeø	Salary	VT_S	VT_e
e1	7K	5	10
e1	8K	10	20
e2	6K	10	14
e2	8K	15	*NOW*

□

Observation 3.3 If the temporal information recorded by the tuples in an E-relation is lifespan only and the temporal information recorded by the tuples in an A-relation referencing the E-relation is transaction time only it should not be enforced that the transaction-time intervals recorded by an tuple of the A-relation is included in the lifespan interval recorded by the tuple in the E-relation with the same value of the E-attribute.

This should not be enforced because the user may have changed the lifespan interval of the tuple in the E-relation after the time, the first tuple representing an attribute value was inserted in the A-relation, or even after the current tuple in the A-relation, was inserted into the database. However, we have to require that all current tuples in the A-relation reference tuples in the E-relation that are current, meaning that the current time instant (c_{now}) is included in the lifespan recorded by the tuple referenced. This is expressed by the following constraint.

Constraint 3.11 (Liveness Constraint) Let R be an A-relation recording transaction time only and let S be an E-relation referenced by R and recording lifespan only; then

$$\forall r \in R(r.TT_e = UC \Rightarrow (\exists s \in S(r.E\text{ø} = s.E\text{ø} \wedge c_{now} \in [s.LS_s, s.LS_e])))$$ □

Example 13 To illustrate the liveness constraint (3.11), let the A-relation *Employee_Salary* record the transaction time of the salary values for the employees recorded in the E-relation *Employee* together with their lifespan. First, assume that the current time is 17. Then it is possible for the third tuple in the A-relation to have *UC* as end timestamp. Second, assume that the current time now is 21. Then the lifespan end timestamp of employee e1 and the third tuple in the A-relation lead to a violation of the liveness constraint.

Employee

employeeø	LS_S	LS_e
e1	5	20
e2	10	*NOW*

Employee_Salary

employeeø	Salary	TT_S	TT_e
e1	7K	1	14
e2	6K	10	16
e1	7K	15	*UC*
e2	8K	17	*UC*

□

If the temporal information recorded by an E-relation is lifespan only and the temporal information recorded by an A-relation referencing the E-relation is both

valid and transaction time then the valid-time interval recorded by a current tuple in the A-relation (representing a current attribute value or a current relationship) must be included in the lifespan recorded by the tuple in the E-relation that has the same value of the E-attribute and that has a lifespan including the current time instant.

Constraint 3.12 (Valid Time Inclusion, Liveness Constraint) Let R be an A-relation recording valid time and transaction time and let S be an E-relation referenced by R and recording lifespan only; then

$$\forall r \in R(r.TT_e = UC \Rightarrow (\exists s \in S(r.E\emptyset = s.E\emptyset \wedge c_{now} \in [s.LS_s, s.LS_e] \wedge [s.LS_s, s.LS_e] \supseteq [r.VT_s, r.VT_e]))) \quad \square$$

Observation 3.3 also applies to the transaction-time intervals recorded by the tuples in a bitemporal A-relation referencing tuples in E-relations recording lifespan only.

If the temporal information recorded by an E-relation is transaction time only and the temporal information recorded by A-relations referencing the E-relation includes transaction time then the transaction-time interval recorded by a tuple in the A-relation must be included in the transaction-time interval recorded by the tuple in the E-relations with the same value of the E-attribute, since the attribute value cannot have been inserted before the insertion of the entity.

Constraint 3.13 (Transaction Time Inclusion) Let R be an A-relation recording transaction time only, or both valid and transaction time, let S be an E-relation referenced by R recording transaction time only; then

$$\forall r \in R\ \exists s \in S(r.E\emptyset = s.E\emptyset \wedge [s.TT_s, s.TT_e] \supseteq [r.TT_s, r.TT_e]) \quad \square$$

Observation 3.4 If the temporal information recorded by an E-relation is transaction time only and the temporal information recorded by A-relations referencing the E-relation is valid time only or both valid time and transaction time, it should not be enforced that the valid-time interval recorded by a tuple of the A-relation is included in the transaction-time interval recorded by the tuple of the E-relation with the same value of the E-attribute.

This should not be enforced because a tuple of the E-relation may have been inserted into the database long time after the entity it represents began to exist in the modeled mini-world and the information recorded by tuples of the A-relation were therefore valid before the insertion date.

If the temporal information recorded by an E-relation is both lifespan and transaction time and the temporal information recorded by an A-relation referencing the E-relation is valid time only, then the valid-time interval recorded by a tuple in the A-relation must be included in the lifespan interval recorded by current tuples in the E-relation (the tuples in the E-relation with transaction-time end value UC) with the same value of the E-attribute.

Constraint 3.14 (Valid Time Inclusion) Let R be an A-relation recording valid time only, let S be an E-relation referenced by R recording both lifespan and transaction time; then

$$\forall r \in R\ \exists s \in S(r.E\emptyset = s.E\emptyset \wedge s.TT_e = UC \wedge [s.LS_s, s.LS_e] \supseteq [r.VT_s, r.VT_e]) \quad \square$$

If the temporal information recorded by an E-relation is both lifespan and transaction time and the temporal information recorded by an A-relation referencing the E-relation is transaction time only then the transaction-time interval recorded by a tuple in the A-relation must be included in the union of the transaction-time intervals of all the tuples of the E-relation with the same value of the E-attribute.

Constraint 3.15 (Transaction Time Inclusion) Let R be an A-relation recording transaction time only, let S be an E-relation referenced by R and recording both lifespan and transaction time; then

$$\forall r \in R\ \exists s_i \in S,\ i = 1, \ldots, n(r.E\emptyset = s_i.E\emptyset \wedge [r.TT_s, r.TT_e] \subseteq \bigcup_{i=1}^{n} [s_i.TT_s, s_i.TT_e]) \quad \square$$

If the temporal information recorded by an E-relation is both existence time and transaction time and the temporal information recorded by an A-relation referencing the E-relation is bitemporal, then the valid-time interval recorded by a tuple in the A-relation must be included in the lifespan interval recorded by the tuple in the E-relation with the same value of the E-attribute if they are current at the same transaction time point. Furthermore, the transaction-time interval of the tuples in the A-relation must be included in the union of the transaction-time intervals of all the tuples of the E-relation with the same value of the E-attribute.

Constraint 3.16 (Bitemporal Inclusion) Let R be an A-relation recording both valid and transaction time, let S be an E-relation referenced by R recording both lifespan and transaction time; then

$$\forall r \in R\ \exists s_i \in S,\ i = 1, \ldots, n((r.E\emptyset = s_i.E\emptyset \wedge [r.TT_s, r.TT_e] \subseteq \bigcup_{i=1}^{n} [s_i.TT_s, s_i.TT_e]) \wedge ([r.TT_s, r.TT_e] \cap [s_i.TT_s, s_i.TT_e] \neq \emptyset \Rightarrow [r.VT_s, r.VT_e] \subseteq [s_i.LS_s, s_i.LS_e])) \quad \square$$

Example 14 To exemplify the above constraint, let *Employee* be an E-relation recording both lifespan and transaction time of employees, and let *Employee_Salary* be an A-relation recording the valid and transaction time of the salaries of the employees. First, to see that the relations satisfy the first inclusion in the constraint, we observe that the total transaction time interval of e1 is $[1, UC]$ and that the total transaction-time interval of e2 is $[10, UC]$. It is now easy to verify that all the tuples in the A-relation referencing e1 and e2 have transaction-time intervals that are included in the above intervals. Second, to verify that the relations also satisfy

the second inclusion in the constraint, note that the first tuple in the A-relation is current at the same time as both the first and the second tuples of the E-relation; in both cases, the valid-time interval is included in the lifespan. Similar checks may be applied to the rest of the tuples.

Employee

employeeø	TT_S	TT_e	LS_S	LS_e
e1	1	8	10	15
e1	9	*UC*	5	20
e2	10	*UC*	10	*NOW*

Employee_Salary

employeeø	Salary	TT_S	TT_e	VT_S	VT_e
e1	7K	1	14	10	15
e2	6K	10	18	10	14
e1	7K	15	*UC*	5	20
e2	8K	19	*UC*	15	*NOW*

□

Between E-Relations

In this section, we present the temporal constraints that must hold between an E-relation that represents a weak entity type and the E-relations representing the owner(s), between temporal relationships supporting lifespan and the E-relations representing the participating entity types, and between E-relations that represent subclasses and subclasses.

For every current tuple of an E-relation representing a weak entity type, there must exist a reachable current tuple in the E-relation representing the owner of the weak entity type. If a weak entity type has more than one owner, this must be true for each one. A tuple in an E-relation is reachable if and only if it is possible, by following the reference described by the foreign key in the relation representing the identifying relationship, to identify a tuple in the E-relation representing the owner of a weak entity type. The same applies for tuples of E-relations representing relationship types recording lifespan and the E-relations representing the participating entity types. There are 16 different combinations of temporal support to consider, but we do not consider the cases where one of the involved E-relations is non-temporal since the observations made in the previous section also apply here. So, we can reduce the definitions of constraints to 9 cases; the table below indicates which constraint applies to which combination of temporal support between the involved E-relations. The purpose of the constraints is twofold. First, they are temporal existence dependency constraints and, second, they are temporal inclusion constraints.

		R		
		LS	TT	LT
S_i	LS	Constraint 3.17	Observation 3.3, Constraint 3.18	Constraint 3.19
	TT	Observation 3.4	Constraint 3.20	Constraint 3.20
	LT	Constraint 3.21	Constraint 3.22	Constraint 3.23

Constraint 3.17 (Temporal Existence Dependency, Lifespan Inclusion) Let R be an E-relation recording lifespan only, referencing via an intermediate relation K the E-relations $S_i,\ i = 1, \ldots, n$, recording lifespan only; then

$$\forall r \in R\ \exists k \in K\ \forall i \exists s \in S_i (r.E\text{ø} = k.E\text{ø} \wedge k.E_i\text{ø} = s.E_i\text{ø} \wedge [r.LS_s, r.LS_e] \subseteq [s.LS_s, s.LS_e])$$ □

Example 15 To better understand the temporal existence dependency constraints, consider the relations below, which satisfy constraint 3.17 (the current time is 21. The relations are marked with S, R, and K to indicate which relation is which.

Employee (S)

*employee*ø	LS_s	LS_e
e1	3	10
e2	5	*NOW*

Dependent (R)

*dependent*ø	LS_s	LS_e
d1	5	7
d2	6	10
d3	8	*NOW*
d4	8	15

Dependent_DName_Sex (K)

*dependent*ø f.k.	D_Name pt.k.	Sex	*employee*ø f.k., ow.
d1	Jens	male	e1
d2	Ulla	female	e1
d3	Pia	female	e2
d4	Peter	male	e2

□

Note that when Constraints 3.17 through 3.23 apply to E-relations representing superclass/subclass relationships then the intermediate relation K is equal to the E-relation R.

Constraint 3.18 (Temporal Existence Dependency, Liveness Constraint) Let R be an E-relation recording transaction time only, referencing via an intermediate relation K the E-relations $S_i,\ i = 1, \ldots, n$, recording lifespan only; then

$$\forall r \in R(r.TT_e = UC \Rightarrow \exists k \in K\ \forall i \exists s \in S_i (r.E\text{ø} = k.E\text{ø} \wedge k.E_i\text{ø} = s.E_i\text{ø} \wedge c_{now} \in [s.LS_s, s.LS_e]))$$ □

Constraint 3.19 (Temporal Existence Dependency, Lifespan Inclusion) Let R be an E-relation recording both lifespan and transaction time, referencing via an intermediate relation K the E-relations $S_i,\ i = 1, \ldots, n$, recording lifespan only; then

$$\forall r \in R(r.TT_e = UC \Rightarrow \exists k \in K\ \forall i \exists s \in S_i(r.E\emptyset = k.E\emptyset \wedge k.E_i\emptyset = s.E_i\emptyset \wedge [r.LS_s, r.LS_e] \subseteq [s.LS_s, s.LS_e] \wedge c_{now} \in [s.LS_s, s.LS_e])) \quad \square$$

Constraint 3.20 (Temporal Existence Dependency, Transaction Time Inclusion) Let R be an E-relation recording transaction time only or both lifespan and transaction time, referencing via an intermediate relation K the E-relations S_i, $i = 1, \ldots, n$, recording transaction time only; then

$$\forall r \in R\ \exists k \in K\ \forall i \exists s \in S_i(r.E\emptyset = k.E\emptyset \wedge k.E_i\emptyset = s.E_i\emptyset \wedge [r.TT_s, r.TT_e] \subseteq [s.TT_s, s.TT_e]) \quad \square$$

Constraint 3.21 (Temporal Existence Dependency, Lifespan Inclusion) Let R be an E-relation recording lifespan only, referencing via an intermediate relation K the E-relations S_i, $i = 1, \ldots, n$, recording both lifespan and transaction time; then

$$\forall r \in R\ \exists k \in K\ \forall i \exists s \in S_i(s.TT_e = UC \Rightarrow r.E\emptyset = k.E\emptyset \wedge k.E_i\emptyset = s.E_i\emptyset \wedge [r.LS_s, r.LS_e] \subseteq [s.LS_s, s.LS_e]) \quad \square$$

Constraint 3.22 (Temporal Existence dependency, Transaction Time Inclusion) Let R be an E-relation recording transaction time only, referencing via an intermediate relation K the E-relations S_i, $i = 1, \ldots, n$, recording both lifespan and transaction time; then

$$\forall r \in R\ \exists k \in K\ \forall i \exists s_j \in S_i,\ j = 1, \ldots, m(r.E\emptyset = k.E\emptyset \wedge k.E_i\emptyset = s.E_i\emptyset \wedge [r.TT_s, r.TT_e] \subseteq \bigcup_{j=1}^{m} [s_j.TT_s, s_j.TT_e]) \quad \square$$

Constraint 3.23 (Temporal Existence Dependency, Lifespan and Transaction Time Inclusion) Let R be an E-relation recording both lifespan and transaction time, referencing via an intermediate relation K the E-relations S_i, $i = 1, \ldots, n$, recording both lifespan and transaction time; then

$$\forall r \in R\ \exists k \in K\ \forall i \exists s_j \in S_i,\ j = 1, \ldots, m(r.E\emptyset = k.E\emptyset \wedge k.E_i\emptyset = s.E_i\emptyset \wedge [r.TT_s, r.TT_e] \subseteq \bigcup_{j=1}^{m} [s_j.TT_s, s_j.TT_e] \wedge ([r.TT_s, r.TT_e] \cap [s_j.TT_s, s_j.TT_e] \neq \emptyset \Rightarrow [r.LS_s, r.LS_e] \subseteq [s_j.LS_s, s_j.LS_e])) \quad \square$$

Example 16 This example aids in understanding constraint 3.23; will use the first tuple of *Works_for* for illustration. Assume that the current time is 21. First we have to verify that the transaction time interval of w1 is included in the total transaction-time of e1 and p1. Second, we have to verify that lifespan intervals of w1 is included in the lifespan intervals recorded for e1 and p1 if these are current at the same time. As can be seen, tuple w1 satisfies the constraint.

Employee (S1)

employeeø	TT_S	TT_E	LS_S	LS_E
e1	1	8	5	15
e1	9	*UC*	5	25
e2	10	*UC*	10	*NOW*

Project (S2)

projectø	TT_S	TT_E	LS_S	LS_E
p1	2	12	2	11
p2	3	18	5	15
p3	15	*UC*	15	23
p4	19	*UC*	15	*NOW*

Works_for (R)

works_forø	TT_S	TT_E	LS_S	LS_E
w1	2	10	5	11
w2	9	15	10	15
w3	10	16	10	15
w4	15	*UC*	15	20
w5	19	*UC*	15	25

Works_for_Employee_Project (K)

works_forø f.k.	*employeeø* f.k.	*projectø* f.k.
w1	e1	p1
w2	e1	p2
w3	e2	p2
w4	e2	p3
w5	e1	p4

□

In addition to the above-mentioned constraints, some additional constraints apply between E-relations representing superclasses and subclasses in superclass/ subclass relationship types. We first cover the case where Option 1 was chosen for the mapping. Constraints 3.24 through 3.27 are inclusion constraints, while Constraints 3.28 through 3.29 are disjointness constraints.

The table below describes which inclusion constraints apply between a superclass and its subclasses for the different combinations of temporal support. The “X” indicates that a combination is not valid, due to the constraints on the inheritance of temporal support.

		R_i			
		None	LS	TT	LT
Superclass R	None	Conventional inclusion constraint	Conventional inclusion constraint	Constraint 3.24, Constraint 3.25	Constraint 3.24, Constraint 3.25
	LS	X	Conventional inclusion constraint	X	Constraint 3.24, Constraint 3.25
	TT	X	X	Constraint 3.26, Constraint 3.27	Constraint 3.26, Constraint 3.27
	LT	X	X	X	Constraint 3.26, Constraint 3.27

Constraint 3.24 (Temporal Inclusion Dependency, Optional Participation) Let R be the E-relation representing entity type E, which is non-temporal or supports lifespan only. Let $R_1, R_2, \ldots, R_n$ be the E-relations representing its subclasses $E_1, E_2, \ldots, E_n$ supporting transaction time; then

$$\forall R_i \forall s \in R_i (s.TT_e = UC \Rightarrow \exists r \in R(s.E\emptyset. = r.E\emptyset)) \quad \square$$

Constraint 3.25 (Temporal Inclusion Dependency, Total Participation) Let R be the E-relation representing entity type E, which is non-temporal or supports lifespan only. Let $R_1, R_2, \ldots, R_n$ be the E-relations representing its subclasses $E_1, E_2, \ldots, E_n$ supporting transaction time. Then Constraint 3.24 and the following must hold.

$$\forall r \in R \exists s \in R_i (s.E\emptyset = r.E\emptyset \wedge s.TT_e = UC) \quad \square$$

Constraint 3.26 (Temporal Inclusion Dependency, Optional Participation) Let R be the E-relation representing entity type E, which supports transaction time. Let $R_1, R_2, \ldots, R_n$ be the E-relations representing its subclasses $E_1, E_2, \ldots, E_n$ supporting transaction time; then

$$\forall R_i \; \forall s \in R_i (s.TT_e = UC \Rightarrow \exists r \in R(s.E\emptyset = r.E\emptyset \wedge r.TT_e. = UC)) \quad \square$$

Constraint 3.27 (Temporal Inclusion Dependency, Total Participation) Let R be the E-relation representing entity type E, which supports transaction time. Let $R_1, R_2, \ldots, R_n$ be the E-relations representing its subclasses $E_1, E_2, \ldots, E_n$ supporting transaction time. Then Constraint 3.26 and the following must hold.

$$\forall r \in R(r.TT_e = UC \Rightarrow \exists R_i \; \exists s \in R_i (s.E\emptyset = r.E\emptyset \wedge s.TT_e = UC)) \quad \square$$

The disjointness constraints for superclass/subclass relationship types follow. Depending on the temporal support of the subclasses, different constraints apply. The table below describes which constraints apply to which combination of temporal support between any two subclasses belonging to the same superclass/subclass relationship type.

		Subclass R_i			
		None	LS	TT	LT
Subclass R_j	None	Conventional disjointness constraint	Conventional disjointness constraint	Constraint 3.28	Constraint 3.28
	LS	Conventional disjointness constraint	Conventional disjointness constraint	Constraint 3.28	Constraint 3.28
	TT	Constraint 3.28	Constraint 3.28	Constraint 3.29	Constraint 3.29
	LT	Constraint 3.28	Constraint 3.28	Constraint 3.29	Constraint 3.29

Constraint 3.28 (Temporal Disjoint Participation Constraint 1) Let R be the E-relation representing entity type E, the superclass of the entity types $E_1, E_2, \ldots, E_n$. Let $R_1, R_2, \ldots, R_n$ be the E-relations representing the subclasses. For any two E-relations $R_i, R_l,\ i \neq l$, where R_i records transaction time and R_l does not,

$$\forall s_j \in R_i,\ j = 1, \ldots, m(s_j.TT_e. = UC \Rightarrow \nexists r \in R_j(s_j.E\emptyset = r.E\emptyset)) \qquad \square$$

Constraint 3.29 (Temporal Disjoint Participation Constraint 2) Let R be the E-relation representing entity type E, the superclass of the entity types $E_1, E_2, \ldots, E_n$. Let $R_1, R_2, \ldots, R_n$ be the E-relations representing the subclasses. For any two E-relations $R_i, R_l,\ i \neq l$, both recording transaction time,

$$\forall s_j \in R_i,\ j = 1, \ldots, m(s_j.TT_e. = UC \Rightarrow \nexists r \in R_j(s_j.E\emptyset = r.E\emptyset \wedge r.TT_e = UC)) \qquad \square$$

We proceed to present the constraints that apply if Option 2 was chosen to map the superclass/subclass relationship type, that is, the participation constraint for the relationship type was specified as total and disjoint. In this cases, there is no E-relation representing the superclass, but only E-relations representing the subclasses. Thus, the set of constraints is reduced to include only disjointness constraints that ensure that any tuple participating in the superclass/subclass relationship is represented in exactly one subclass.

Constraint 3.30 (Temporal Disjoint Participation Constraint 3) Let $R_1, R_2, \ldots, R_n$, be the E-relations representing of the entity types $E_1, E_2, \ldots, E_n$ which are subclasses in the superclass/subclass relationship type. For any two E-relations R_i, R_l, $i \neq l$, where R_i records transaction time and R_l does not, the following holds.

$$\forall s_j \in R_i,\ j = 1, \ldots, m(s_j.TT_e = UC \Rightarrow \nexists r \in R_j(s_j.E\emptyset = r.E\emptyset)) \qquad \square$$

Constraint 3.31 (Temporal Disjoint Participation Constraint) Let $R_1, R_2, \ldots, R_n$ be the E-relations representing of the entity types $E_1, E_2, \ldots, E_n$, which are subclasses in the superclass/subclass relationship type. For any two E-relations R_i, R_l, $i \neq l$, both recording transaction time, the following holds.

$$\forall s_j \in R_i,\ j = 1, \ldots, m(_j.TT_e. = UC \Rightarrow \nexists r \in R_j(s_j.E\emptyset = r.E\emptyset \wedge r.TT_e = UC)) \quad \square$$

4 Mapping From the Surrogate-Based Model to the Lexically-Based Model

In this section the mapping from the surrogate-based relational target model to the lexically-based relational target model, defined in Section 2.3, is presented.

The input to this part of the algorithm is the E-relations and the A-relations that result from Step I. We do not always know if an A-relation represents an attribute or a relationship type. Some A-relations represent both non-temporal attributes and non-temporal relationship types. We also do not know whether an E-relation represents an entity type or a relationship type (if the temporal support of the relationship type includes lifespan).

Since this target model does not have an E-domain, and therefore no E-attributes, a set of lexical attributes, possibly concatenated with some timestamp attributes, must now serve as the existence lists and unique identifiers for the relations. In order to guarantee that we can replace all the non-lexical surrogate-based keys with lexical keys, we need to make sure that every single entity occurrence as modeled by the TIMEER diagram is uniquely lexically referenceable. The definition of the TIMEER model requires that every single entity type has a unique lexical key [9]. This requirement is stronger than requiring unique lexical referencability, and we can therefore replace all the E-attributes with the proper set of lexical attributes, i.e., the set of lexical attributes that constitutes the user-defined lexical keys of the modeling constructs in the diagram. We have chosen to retain the E-relation in the lexically-based model since they still model existence. The primary keys of these relations will be the lexical keys concatenated with the proper set of timestamp attributes (the same sets as in the previous section).

The attribute, or set of attributes, that has been defined by the user as the lexical key of a modeling construct is contained in an A-relation representing this attribute. The attribute(s) is marked "u.k" (or "pt.k." for weak entity types). A part of the key of such an A-relation is the E-attribute, referencing as foreign key the E-relation representing the modeling construct that the attributes is key of. The A-relations containing the lexical keys of the diagram must therefore be identified first.

For each of the above identified A-relations, we must now identify all the relations that contain the E-attributes that are a part of the key of the A-relation. This is done by first identifying all the E-relations referenced by the A-relation. These E-relations include the E-relation representing the entity type of which the attribute is the lexical key of and all the subclasses of this entity type, if any. Then, for each

of the just identified E-relations, we identify all the A-relations that include the E-attribute of the E-relation as foreign key. These A-relations include A-relations that represents other attributes of the entity type; A-relations representing the temporal relationships the entity type participate in; and A-relations representing attributes of the relationship, the entity type participate in.

We now have a set of relations that include the same E-attribute. This E-attribute has to be replaced by the set of lexical attributes that is specified as the key of the corresponding entity type. When we have replaced all E-attributes we have to remove duplicate columns (the only place we have duplicate columns is in the A-relation containing the attributes marked "u.k." or "pt.k.").

Example 17 This example illustrate Step II as explained until now. The first set of relations below is the relations resulting from Step I, mapping the diagram in Figure 7. The second set of relations is the set of relations resulting from Step II with the first set as input. The first relation that is identified is the A-relation *Employee_ID_Name_Birth_date*. Then the E-relation *Employee* and the A-relation *Employee_Salary* are identified. Now we replace all the occurrences of *employeeø* with ID and remove the duplicate column ID in the A-relation *Employee_ID_Name_Birth_date*.

Step I:

Employee

employeeø

Employee_ID_Name_Birth_date

employeeø f.k.	ID u.k.	Name	Birth_date

Employee_Salary

employeeø f.k.	Salary	T

Step II:

Employee

ID

Employee_ID_Name_Birth_date

ID u.k., f.k.	Name	Birth_date

Employee_Salary

ID u.k., f.k.	Salary	T

□

We have now imported the lexical key of all relations created to represent entity types and relationship types considered attributes. Note that in the algorithm presented in Appendix C, there is a slight difference in the way the lexical key import is performed for regular and weak entity types, but the overall principle as described above is the same.

We must now import the lexical key of relationship types considered to exist in their own right. We utilize the fact that the E-relations that represent these are now the only E-relations that still contain E-attributes.

For each of these E-relations, we identify all the relations referencing the E-relation. One of these relations will contain the lexical key of all the involved entity types and the non-temporal attributes of the relationship type, if any. The reason for this is that the lexical key of all the involved entity types were imported into

it during the first step of this key import procedure. This relation is characterized by not having timestamp attributes. We therefore identify this relation among all the relations that reference the E-relation. The lexical key of the relationship is the concatenation of all the foreign keys in this relation and must replace all occurrences of the E-attributes in the E-relation and all the relations that reference the E-relation. We must remove the duplicate columns from the A-relation representing the involvement.

Example 18 This example illustrates the last part of Step II. The first set of relations is the relations resulting from Step I mapping the diagram in Figure 12 and the first part of Step II. The second set of relations is the result after Step II.

Step I, Step II (first part):

Employee

$\overline{\text{E\_ID}}$

Employee_EID_EName

E_ID u.k., f.k.	E_Name

Project

$\overline{\text{P\_ID}}$

Project_PID_PName

P_ID u.k., f.k.	P_Name

Works_for

$\overline{\textit{works\_for}\text{ø}}$	T

Works_for_Employee_Project

*works_for*ø f.k.	E_ID f.k.	P_IDf.k.

Step II (second part):

Employee

$\overline{\text{E\_ID}}$

Employee_EID_EName

E_ID u.k., f.k.	E_Name

Project

$\overline{\text{P\_ID}}$

Project_PID_PName

P_ID u.k., f.k.	P_Name

Works_for

$\overline{\text{E\_ID}}$	$\overline{\text{P\_ID}}$	T

Works_for_Employee_Project

E_ID f.k.	P_ID f.k.

□

The fact that we do not delete relations means that we can utilize all the constraints developed in Section 3.5. The E-attributes mentioned in the constraints simply is the lexical key instead.

5 Summary and Future Research Direction

In this paper we have defined a two-step mapping algorithm from the TIMEER model to a lexically-based relational target model. This algorithm describes how to map a conceptual model that supports lifespans, transaction time, and valid time into relations in the relational model. Thus, the algorithm presented corrects one of the major problems, we have encountered in the existing algorithms, namely that these at best consider only valid time.

The first step maps a temporal ER diagram into relations in an intermediate, surrogate-based relational target model. The purpose of this step is to create relations that preserve the semantics of the diagram transformed. With the first step, we also meet future needs, since we believe that commercial database products in the future will support surrogate-based relational data models. The second step replaces the surrogate attributes with the lexical keys defined by the user in the temporal ER diagram. This step is necessary to meet today's needs, since no commercial DBMS currently supports a surrogate-based relational data model. We have defined a set of temporal constraints that enforce the semantics of the temporal aspects of the application modeled by the diagram. The constraints are defined for the surrogate-based target model only, but need only be slightly rewritten to apply to the lexically-based target model.

This paper presents a mapping from the temporal ER model to two relational models that do not support the semantics of the time domains. Another approach could be to develop an algorithm that has as its target one of the many temporally extended relational models that have been defined over the last two decades [16]. Yet another approach could be to map to the Bitemporal Conceptual Data Model [12], which is a design model, and then utilize the mapping algorithms provided by this model to five different bitemporal relational data models.

As an alternative to mapping temporal ER diagrams into a schema of a different implementation platform, another approach is to assume a system that implements the temporal ER model directly and develop a query language for querying the temporal ER databases.

Acknowledgements

This work is supported in part by grants 9502695 and 9700780 from the Danish Technical Research Council and grant 9701406 from the Danish Natural Science Research Council. The research was performed in part while the first author visited Georgia Institute of Technology.

References

[1] M. A. Casanova and J. E. Amaral de Sa. Mapping Uninterpreted Schemes into Entity-Relationship Diagrams: Two Applications to Conceptual Schema Design. *IBM Journal of Research and Development*, 28(1):82–98, 1984.

[2] P. P-S. Chen. The Entity-Relationship Model – Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[3] E. F. Codd. Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems*, 4(4):397, December 1979.

Reprinted in M. Stonebraker, Readings in Database Systems, Morgan Kaufmann, 1988.

[4] C. E. Dyreson and R. T. Snodgrass. Temporal granularity. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, Chapter 19, pages 347–383. Kluwer Academic Publishers, 1995.

[5] C. E. Dyreson and R. T. Snodgrass. The Baseline Clock. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, Chapter 5, pages 77–96. Kluwer Academic Publishers, 1995.

[6] C. E. Dyreson, M. Soo, and R. T. Snodgrass. The Data Model for Time. In R. T. Snodgrass, editor, *The TSQL2 Temporal Query Language*, Chapter 6, pages 97–101. Kluwer Academic Publishers, 1995.

[7] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Benjamin/Cummings, 2. edition, 1994.

[8] S. Ferg. Modeling the Time Dimension in an Entity-Relationship Diagram. In Proceedings of the *4th International Conference on the Entity-Relationship Approach*, pages 280–286, Silver Spring, MD, 1985.

[9] H. Gregersen and C. S. Jensen. Conceptual Modeling of Time-varying Information. TIMECENTER Technical Report TR-35, September 1998.

[10] P. Hall, J. Owlett, and S. J. P. Todd. Relations and Entities. In G. M. Nijssen, editor, *Modelling in Data Base Management Systems*, pages 201–220. North-Holland, 1976.

[11] C. S. Jensen and C. E. Dyreson [editors]. The Consensus Glossary of Temporal Database Concepts - February 1998 Version. In O. Etzion, S. Jajodia, and S. Sripada, editors, *Temporal Databases: Research and Practice*, Volume 1399 of *Lecture Notes in Computer Science*, pages 367–405. Springer-Verlag, 1998.

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

[13] V. S. Lai, J-P. Kuilboer, and J. L. Guynes. Temporal Databases: Model Design and Commercialization Prospects. *DATA BASE*, 25(3):6–18, 1994.

[14] A. Narasimhalu. A Data Model for Object-Oriented Databases with Temporal Attributes and Relationships. Technical report, National University of Singapore, 1988.

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

[16] R. T. Snodgrass. Temporal Databases. In A. U. Frank, I. Campari, and U. Formanti, editors, *Theories and Methods of Spatio-Temporal Reasoning in Ge-*

ographic Space, Volume 639 of *Lecture Notes in Computer Science*, pages 22–64. Springer-Verlag, 1992.

[17] C. I. Theodoulidis, P. Loucopoulos, and B. Wangler. A Conceptual Modelling Formalism for Temporal Database Applications. *Information Systems*, 16(4):401–416, 1991.

[18] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, Volume I. Computer Science Press, 1988.

A Procedure for Adding Timestamp Attributes to Relations

This section presents the procedure that will be used to extend relations representing temporal entity types, temporal attributes, and temporal relationship types with timestamp attributes.

1. If the temporal support of the modeling construct, for subclasses including the support inherited from any superclass, is
   (a) LS, then extend the relation with two timestamp attributes L_s and L_e and extend the primary key of the relation with L_s.
   (b) LT, then extend the relation with four timestamp attributes L_s, L_e, TT_s, and TT_e and extend the primary key of the relation with TT_s, L_s, and L_e.
   (c) TT, then extend the relation with two timestamp attributes TT_s and TT_e, and extend the key of the relation with TT_s.
   (d) VT, then extend the relation with two timestamp attributes VT_s and VT_e, and extend the key of the relation with VT_s.
   (e) BT, then extend the relation with four timestamp attributes VT_s, VT_e, TT_s, and TT_e, and extend the key of the relation with, VT_s, VT_e, and TT_s.

B Step I

This section presents the algorithm that transforms TIMEER diagrams to relations in the surrogate-based relational target model.

1. For each entity type E, *not* participating in a superclass/subclass relationship, create an E-relation R that includes an E-attribute, $e\emptyset$. If E has non-temporal attributes $A_1, A_2, \ldots, A_n$, create an A-relation $R\_A_1\_A_2\_\ldots\_A_n$ that includes $e\emptyset$ and $A_1, A_2, \ldots, A_n$. Include all single-valued components of non-temporal composite attributes. Mark the lexical attributes that participate in a user-defined key with "u.k." if the entity is regular, or "pt.k." if it is weak.

The key of $R\_A_1\_A_2\_ \ldots \_A_n$ is the E-attribute. The primary key of R is $e\text{ø}$. If E is temporal, proceed as described in Appendix A.

2. For each entity type participating in a superclass/subclass relationship with superclass E and subclasses $S_1, S_2, \ldots, S_n$, either

   (a) create an E-relation R to represent the superclass E and E-relations $R_1, R_2, \ldots, R_n$ to represent the the subclasses $S_1, S_2, \ldots, S_n$ that each include the E-attribute, $e\text{ø}$. If one of E or S_i has non-temporal attributes $A_1, A_2, \ldots, A_n$, create A-relations $R\_A_1\_A_2\_ \ldots \_A_n$ and $R_i\_A_1\_A_2\_ \ldots \_A_n$ that include $e\text{ø}$ and $A_1, A_2, \ldots, A_n$. Include all single-valued components of non-temporal composite attributes. The key of $R\_A_1\_A_2\_ \ldots \_A_n$ and $R_i\_A_1\_A_2\_ \ldots \_A_n$ is the E-attribute. The primary key of R and R_i is the E-attribute $e\text{ø}$. If E or S_i are temporal, extend the relations as described in Appendix A.

   (b) Or create E-relations $R_1, R_2, \ldots, R_n$ to represent the subclasses $S_1, S_2, \ldots, S_n$ that include an E-attribute, $e\text{ø}$. If E has non-temporal attributes $A_1, A_2, \ldots, A_k$ and R_i has non-temporal attributes $A_l, A_m, \ldots, A_n$, create an A-relation $R_i\_A_1\_A_2\_ \ldots \_A_n$ that includes $e\text{ø}$ and $A_1, A_2, \ldots, A_n$. Include all single-valued components of non-temporal composite attributes. The key of $R_i\_A_1\_A_2\_ \ldots \_A_n$ is the E-attribute. The primary key of R_i is the E-attribute. If S_i is temporal, apply the procedure from Appendix A

3. For each simple temporal attribute A of entity type E, create an A-relation R_A that includes the E-attribute of E, $e\text{ø}$, the attribute A; include all single-valued components of temporal composite attributes. The key of R_A is the E-attribute in combination with A if A is multivalued. Extend R_A using the procedure in Appendix A.

4. For each binary non-temporal relationship type S, involving entity types E_1 and E_2, both with snapshot participation constraint (0,1) or (1,1), identify the A-relations R_1_X and R_2_X that represent the non-temporal attributes of the participating entity types. Extend, say, R_1_X with the E-attribute $e_2\text{ø}$ of R_2_X as a foreign key. If the relationship type is identifying, always extend the the relation representing non-temporal attributes of the weak entity type and mark the imported $e\text{ø}$ with "ow.". If S has non-temporal attributes $A_1, A_2, \ldots, A_n$, extend R_1_X to include these.

5. For each binary non-temporal relationship type S, involving entity types E_1 and E_2, where one has snapshot participation constraint (0,1) or (1,1) and the other has snapshot participation constraint (0,n) or (1,n), identify the A-relation R_X that represents the non-temporal attributes of the participating entity type that has snapshot participation constraint (0,1) or (1,1). Extend

R_X with the E-attribute e_iø of the E-relation R, representing the other participating entity type as a foreign key. If the relationship type is identifying, mark the imported eø with "ow.". If S has non-temporal attributes $A_1, A_2, \ldots, A_n$, extend R_X to include these.

6. For each binary non-temporal relationship type S, involving entity types E_1 and E_2, both with snapshot participation constraint (0,n) or (1,n), create an A-relation R that include the E-attributes, e_1ø and e_2ø from the E-relations representing E_1 and E_2. If S has non-temporal attributes $A_1, A_2, \ldots, A_n$, extend R to include these. The key of the A-relation is the E-attributes in combination.
7. For each n-ary relationship type S, involving entity types $E_1, E_2, \ldots, E_n$, create an A-relation R that includes the E-attributes, e_iø, of all the participating entity types as foreign keys. If the relationship type is identifying, mark the e_iø of the owner entity types with "ow.". If S has non-temporal attributes $A_1, A_2, \ldots, A_n$, extend R to include these. The key of the A-relation is
   (a) *one* of the E-attributes, e_iø, for which E_i has snapshot participation constraint (0,1) or (1,1),or
   (b) the concatenation of the E-attributes, e_iø, otherwise.
8. For each binary temporal relationship type S, involving entity types E_1 and E_2, supporting valid time, transaction time, or both, create an A-relation R that includes the E-attributes, e_iø, from the E-relations representing the participating entity types as foreign keys. If the relationship type is identifying, mark the e_iø's of the owner entity types with "ow.". The key of R is chosen as in step 7. Use the procedure in Appendix A to extend R. If S has non-temporal attributes $A_1, A_2, \ldots, A_n$, create an A-relation $R\_A_1\_A_2\_\ldots\_A_n$ that includes the E-attributes e_iø and $A_1, A_2, \ldots, A_n$. The E-attributes e_iø that are to part of the key of $R\_A_1\_A_2\_\ldots\_A_n$ are chosen as just described for R.
9. For each temporal n-ary relationship type S involving entity types $E_1, E_2, \ldots, E_n$, supporting valid time, transaction time, or both create an A-relation R that includes the E-attributes, e_iø, of all the participating entity types as foreign keys. If the relationship type is identifying, mark the e_iø's of the owner entity types with "ow." The key of R is chosen as in step 7. Extend R according to Appendix A. If S has non-temporal attributes $A_1, A_2, \ldots, A_n$, create an A-relation $R\_A_1\_A_2\_\ldots\_A_n$ that includes the E-attributes e_iø and $A_1, A_2, \ldots, A_n$. The E-attributes e_iø that is to part of the key of $R\_A_1\_A_2\_\ldots\_A_n$ is chosen as described above.
10. For each temporal attribute A of relationship type S, which is either non-temporal or temporal supporting valid time, transaction time, or both, involv-

ing entity types $E_1, E_2, \ldots, E_n$, create an A-relation R_A that includes the E-attributes, e_iø, of the participating entity types, the attribute A. The E-attributes that is to be part of the key of R_A is chosen as in step 7, and if A is multivalued, include A in the key. Proceed as described in Appendix A.

11. For each temporal relationship type S supporting lifespan, or both lifespan and transaction time, create an E-relation R including an E-attribute sø and the set of timestamp attributes described in Appendix A. Create an A-relation R_X that includes sø and the E-attributes e_iø of the participating entity types as foreign keys. If the relationship type is identifying, mark the e_iø of the owner entity types with "ow.". If S has non-temporal attributes $A_1, A_2, \ldots, A_n$, include these in R_X. The key of R_X is the E-attribute sø.
12. For each temporal attribute A of temporal relationship type S supporting lifespan or both lifespan and transaction time, create an A-relation R_A that includes the E-attribute, sø, of the E-relation representing S, the attribute A, and the set of timestamp attributes prescribed by the procedure in Appendix A. The key of R_A is the E-attribute in combination with the appropriate timestamp attributes. If A is multivalued, include A in the key.

C Step II

This section presents the algorithms that transforms relations in the surrogate-based relational model to relations in the lexically-based relational target model.

1. For each A-relation containing a user-specified key (marked "u.k."), do
   (a) Let the set of attributes that constitute the user-specified key of the A-relation be identified by *UK*.
   (b) Identify the E-relations referenced by the above A-relation by E-attributes that are part of *UK*.
   (c) For each E-relation identified in 1b, do
      i. Let the E-attribute be identified by *EK*.
      ii. identify all relations referencing the E-relation (containing *EK* as foreign key).
      iii. for each of the relations identified in 1(c)ii, do
         A. replace *EK* with *UK*.
      iv. replace *EK* with *UK*.
   (d) Remove duplicate columns in the A-relation.
2. For each A-relation containing a user-specified partial key (marked "pt.k."), do

(a) Let the set of attributes that constitute the user-specified partial key of the A-relation concatenated with all attributes marked "o.w" be identified by *UK*.

(b) Identify the E-relations referenced by the above A-relation by E-attributes that are part of *UK*.

(c) For each E-relation identified in 2b, do

i. Let the E-attribute be identified by *EK*.

ii. identify all relations referencing the E-relation (containing EK as foreign key).

iii. for each of the relations identified in 2(c)ii, do

A. replace *EK* with *UK*.

iv. replace *EK* with *UK*.

(d) Remove duplicate columns in the A-relation.

3. For each E-relation still containing E-attributes, do

(a) Let the E-attribute be denoted *EK*.

(b) Identify all relations containing *EK*.

(c) Identify the relation R among the relations identified in 3b not containing timestamp attributes.

(d) Let *UK* be the set of foreign keys in R that is not an E-attribute.

(e) For each relation identified in 3b, do

i. replace *EK* with *UK*.

(f) Remove duplicate columns from R, the relation identified in 3c.