A basic characterization of relation migration

C. Maria Keet and Alessandro Artale

KRDB Research Centre, Free University of Bozen-Bolzano, Italy,
{keet,artale}@inf.unibz.it

Abstract. Representing and reasoning over evolving objects has been investigated widely. Less attention has been devoted to the similar notion of *relation migration*, i.e., how tuples of a relation (ORM facts) can evolve along time. We identify different ways how a relation can change over time and give a logic-based semantics to the notion of relation migration to capture its behaviour. We also introduce the notion of lifespan of a relation and clarify the interactions between object migration and relation migration. Its use in graphical conceptual data modelling is illustrated with a minor extension to ORM2 so as to more easily communicate such constraints with domain experts.

1 Introduction

Object migration, where, say, John migrates from being an instance of Student to one of Alumnus, has received ample attention in the temporal database and conceptual modelling communities [1–6]. But how do we migrate, say, the ORM fact $\langle$John, CS2000$\rangle \in$ EnrolledIn to $\langle$John, CS2000$\rangle \in$ GraduatedIn when John has completed the degree programme CS2000 successfully? Clearly, one can migrate John to Alumnus and then manually add $\langle$John, CS2000$\rangle \in$ GraduatedIn in the GraduatedIn table, but it is much more elegant from the conceptual point of view to declare such business knowledge by imposing a temporal constraint on the possible evolution of EnrolledIn to GraduatedIn at the ORM fact type level so that a whole tuple (fact) migrates at once.

In analogy to object migration, this paper claims the usefulness of a similar constraint for relations that we call *relation migration*. We are unable to find other accounts —either formal or informal—for the notion of relation migration other than the notion of *status relations* that were introduced for conceptual modelling in [7, 8], which was used for modelling essential and immutable part-whole relations. Tuple migration at the physical schema level has received some attention in database integration scenarios and in distributed databases, but in these scenarios a tuple "migrates" just to instantiate the same relation that happens to be stored in different servers whereas here by "relation migration" we intend the *change of membership* of a tuple from one relation to another. Observe that modelling of relation migration is thus also distinct from state transition diagrams that concern states of single objects, activity diagrams that concern processes but do not explicitly consider the participating entities, and interaction diagrams for modelling use cases. We focus explicitly on, in ORM terminology, the migration of facts (called also *tuples* or *relation instances* in this paper) and the corresponding temporal behaviour of fact types (called also *relations* in this paper).

The main purpose of this work is to fill this conceptual and formal gap by introducing the notion of *relation migration*. The natural questions we are trying to answer in this paper are about the different ways in which a relation migration can occur and how we can represent its behaviour such that it can be used in conceptual data models. We identify different ways how a relation can change over time and give it a logic-based semantics. With the formalization of relation migration, we can precisely define the notion of lifespan and its related notions for a relation, and show how interesting temporal properties of relations can be derived as logical implications from the provided formalization. In particular, we will show how interactions between object- and relation- migration can be captured in the proposed framework as logical implication. To achieve this, we build upon ideas from the framework for temporal conceptual modelling in $\mathcal{ER}_{VT}$ [1] with the ability to capture migration between relations in conceptual data models. Along these lines, we associate to the notion of relation migration a formalization in terms of a model-theoretic semantics and group migrations into *evolution* constraints, *persistence* constraints, and *quantitative* constraints. Using a formal foundation—be it the first order logic presented here or extending $\mathcal{ER}_{VT}$ that is based on the Description Logic $\mathcal{DLR}_{\mathcal{US}}$—one will be able to check consistency of the constraints during conceptual data model development using an automated reasoner.

The remainder of the paper is structured as follows. Section 2 introduces examples of relation migration and identifies requirements that should be met for an adequate representation. The proposed formal characterisation of relation migration and lifespan for relations are described in Section 3. The properties that can be derived as logical implications from the proposed formalization are provided and illustrated in Section 4. We close with our final remarks in Section 5.

2 Requirements analysis

In this section we demonstrate the relevance of the notion of relation migration in information systems and databases. We provide an informal idea of the characteristics that underpin the different ways of migration and describe the requirements that a temporal conceptual modelling language should be able to capture. The described behaviour can be extended to generic n-ary relations.

Example 1. Let us assume an airline company's passenger RDBMS and a passenger who books a flight, hence we have a relation $\langle \texttt{John}, \texttt{AZ123} \rangle \in \textsf{Booking}$ with $\texttt{John} \in \textsf{Passenger}$ and $\texttt{AZ123} \in \textsf{Flight}$, which are normally followed by the events that John also checks in and then boards the plane, $\langle \texttt{John}, \texttt{AZ123} \rangle \in \textsf{CheckIn}$ and $\langle \texttt{John}, \texttt{AZ123} \rangle \in \textsf{Boarding}$. While the booking relation holds even after the tuple *extended* to the check-in relation, i.e., $\langle \texttt{John}, \texttt{AZ123} \rangle$ is a member of both Booking and CheckIn relations, this is not the case for the step from check-in to boarding which causes the tuple $\langle \texttt{John}, \texttt{AZ123} \rangle$ to be *moved* from one to the other relation in the operational database. In addition, for any tuple that is member of the Boarding relation, we know that it must have been a member of CheckIn relation sometime earlier. On the other hand, we cannot force a business rule where tuples of the CheckIn relation migrate sometime in the future to the Boarding relation since a flight can be cancelled after the check-in or the passenger may become ill and does not board the plane anymore.

One can construct a similar story line and type of behaviour for, say, census data with $\langle \texttt{John}, \texttt{Mary} \rangle \in \mathsf{Marriage}$, where $\texttt{John}, \texttt{Mary} \in \mathsf{Person}$ and, as a consequence of a divorce event, we have that $\langle \texttt{John}, \texttt{Mary} \rangle \in \mathsf{Divorce}$. Since they can marry again, then we should allow for the same tuple $\langle \texttt{John}, \texttt{Mary} \rangle$ to become again member of the Marriage relation. On the other hand, in the event that either John or Mary dies, then $\langle \texttt{John}, \texttt{Mary} \rangle \in \mathsf{Widowhood}$, which, once it holds, it holds at all times in the future (i.e., being in Widowhood is a *persistent* relation).

One can also plan for specific time durations. For instance, it is expected that once $\langle \texttt{Professor1}, \texttt{DepartmentA} \rangle \in \mathsf{WorksFor}$, each professor also has to perform administrative duties, such as representing the department in the faculty council during some time of her employment (i.e., $\langle \texttt{Professor1}, \texttt{DepartmentA} \rangle \in \mathsf{Represents}$), and teach courses (i.e., $\langle \texttt{Professor1}, \texttt{DepartmentA} \rangle \in \mathsf{TeachesAt}$). The temporal behaviour of the Represents relation is different from that one of the TeachesAt. Indeed, assuming a scenario where professors do not change their departments, then all professors instantiate the TeachsAt relation which, in turn, can be modeled as a temporally persistent relation. On the other hand, professors do not always have to take part in faculty councils but there are scenarios where specific durations are enforced with business rules saying, for example, that *"each professor serving in a faculty council should have worked for the department since the previous year"*.

Example 2. Let us consider now part-whole relations [9]. A simple change in relation between two objects can be caused by the fact that (*i*) a is structurally a part of b but a gets loose so after that a becomes spatially contained in b, e.g., a component in a medical device breaks loose due to wear and tear. For part-whole relations that are not necessarily part-of in the mereological sense we introduce two examples. First (*ii*), the example of subquantity_of in [9] about a bottle of wine and pouring a subquantity of the wine into a wineglass so that this subquantity in the glass *used to be* a subquantity_of the wine in the bottle and one wants to maintain traceability of quantities over time, which is important especially in the food industry for food safety in the food processing chain. The second example is a case where object- and relation- migration interact. For instance, let's consider (*iii*), $\langle \texttt{John}, \texttt{ACompanyBoard} \rangle \in \mathsf{member\_of}$ with $\texttt{John} \in \mathsf{CEO}$ and subsequently we have a migration of both the object and the relation such that John's role changes to $\texttt{John} \in \mathsf{Consultant}$ and $\langle \texttt{John}, \texttt{ACompanyBoard} \rangle \in \mathsf{advisor\_of}$.

We can add a further dimension regarding the behaviour of relation migration when we also consider rigidity of the objects and the relation that holds between the objects. In particular, to capture the temporal behaviour of part-whole relations, essential and immutable part-whole relations have been introduced [7, 10]; e.g., the part-whole relation $\langle \texttt{wolfram-thread}, \texttt{candescent-light-bulb} \rangle \in \mathsf{structural\_part}$, where the wolfram-thread is an immutable part of the candescent-light-bulb. When the light is broken, the tuple migrates *irreversibly* to $\langle \texttt{wolfram-thread}, \texttt{candescent-light-bulb} \rangle \in \mathsf{containment}$, unlike the medical device in example (*i*).

Types of behaviour for relation migration. Summarising and generalising from the examples, we have identified the following types of behaviour, which are in analogy to their object migration counterpart [1].

- *Evolution constraints* specify how elements of a relation can possibly migrate to another relation—e.g., the instance of Booking migrates (extends) to CheckIn.

– *Persistence constraints* specify persistent states for a relation—e.g., being in Widowhood is persistent for a couple.
– *Quantitative evolution constraints* specify the exact amount of time for the relation migration to happen—e.g., the case of professors who, to be members of the faculty council, should have been members of the faculty since one year.

3 Formalization of relation migration

In this section, we formalise the constraints for relation migration that were described informally in Section 2, and investigate the impact of relation migration on *lifespan* for relations. To keep the formal apparatus to a minimum, we avail of the notion of status relations—formalised in [7, 8] and graphically depicted in Fig.2—that constrain the evolution of a tuple's membership in a relation along its lifespan and which apply only to temporal relations: R is the normal (active) relation, a relation is scheduled (Scheduled-R) if its instantiation is known but its membership will only become active some time later, a suspended relation (Suspended-R) is temporarily inactive, and disabled relations (Disabled-R) are expired relations that never can be used again. Based on the characterisation of relation migration (Section 3.1) and lifespan (Section 3.2), we can model the identified ways of relation migration precisely and explore the interactions between relation and object migration afterwards.

3.1 Basic constraints for relation migration

We start by formalising the behavioural constraints without considering object migration. We distinguish between *evolution* constraints—specifying how elements of a relation can possibly migrate to another relation—*persistence* constraints—specifying persistent states for a relation—and *quantitative evolution* constraints—specifying the exact amount of time for the relation migration to happen. We present the textual syntax and the model-theoretic semantics considering, without loss of generality, binary relations. We use a *temporal interpretation* of the signature of a conceptual data model $\mathcal{L}$, which is a structure of the form: $\mathcal{I} = \big((\mathbb{Z}, <), \Delta^{\mathcal{I}}, \{\cdot^{\mathcal{I}(t)} \mid t \in \mathbb{Z}\}\big)$, where $(\mathbb{Z}, <)$ is the set of integers denoting the intended *flow of time*, $\Delta^{\mathcal{I}} \neq \emptyset$ is the *interpretation domain*, and $\cdot^{\mathcal{I}(t)}$, for $t \in \mathbb{Z}$, is the *interpretation function* which assigns a set $C^{\mathcal{I}(t)} \subseteq \Delta^{\mathcal{I}}$ to each entity type $C \in \mathcal{C}$, and a set $R^{\mathcal{I}(t)}$ of tuples over $\Delta^{\mathcal{I}}$ to each relation $R \in \mathcal{R}$.
Evolution constraints. We distinguish between five different kinds of evolution constraints.
R RDEX R′. D*ynamic* EX*tension of a* R*elation.*
$\langle o_1, o_2\rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \exists t' > t.\langle o_1, o_2\rangle \in \mathtt{R}'^{\mathcal{I}(t')}$
R RDEV R′. D*ynamic* EV*olution of a* R*elation.*
$\langle o_1, o_2\rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \exists t' > t.\langle o_1, o_2\rangle \in \mathtt{R}'^{\mathcal{I}(t')} \wedge \langle o_1, o_2\rangle \notin \mathtt{R}^{\mathcal{I}(t')}$
R SRDEX R′. S*trong* D*ynamic* EX*tension of a* R*elation.*
$\langle o_1, o_2\rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \langle o_1, o_2\rangle \in \mathtt{Scheduled\text{-}R}'^{\mathcal{I}(t)} \wedge \exists t' > t.\langle o_1, o_2\rangle \in \mathtt{R}'^{\mathcal{I}(t')}$
R RDEX^{-} R′. D*ynamic* EX*tension of a* R*elation in the past.*
$\langle o_1, o_2\rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \exists t' < t.\langle o_1, o_2\rangle \in \mathtt{R}'^{\mathcal{I}(t')}$

R RDEV$^-$ R′. D*ynamic* EV*olution of a* R*elation in the past.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \exists t' < t . \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t')} \wedge \langle o_1, o_2 \rangle \notin \mathtt{R}^{\mathcal{I}(t')}$

Persistence constraints. We distinguish between five different kinds of persistence constraints.

R RPEX R′. P*ersistent* EX*tension of a* R*elation.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \forall t' > t . \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t')}$

R RPEV R′. P*ersistent* EV*olution of a* R*elation.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \forall t' > t . \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t')} \wedge \langle o_1, o_2 \rangle \notin \mathtt{R}^{\mathcal{I}(t')}$

R SRPEX R′. S*trong* P*ersistent* EX*tension of a* R*elation.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \langle o_1, o_2 \rangle \in \mathtt{Scheduled\text{-}R'}^{\mathcal{I}(t)} \wedge \forall t' > t . \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t')}$

R RPEX$^-$ R′. P*ersistent* EX*tension of a* R*elation in the past.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \forall t' < t . \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t')}$

R RPEV$^-$ R′. P*ersistent* EV*olution of a* R*elation in the past.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \forall t' < t . \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t')} \wedge \langle o_1, o_2 \rangle \notin \mathtt{R}^{\mathcal{I}(t')}$

Quantitative constraints. We distinguish between five different kinds of quantitative constraints.

R RQEX R′. Q*uantitative* EX*tension of a* R*elation.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t+1)}$

R RQEV R′. Q*uantitative* EV*olution of a* R*elation.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t+1)} \wedge \langle o_1, o_2 \rangle \notin \mathtt{R}^{\mathcal{I}(t+1)}$

R SRQEX R′. S*trong* Q*uantitative* EX*tension of a* R*elation.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \langle o_1, o_2 \rangle \in \mathtt{Scheduled\text{-}R'}^{\mathcal{I}(t)} \wedge \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t+1)}$

R RQEX$^-$ R′. Q*uantitative* EX*tension of a* R*elation in the past.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t-1)}$

R RQEV$^-$ R′. Q*uantitative* EV*olution of a* R*elation in the past.*

$\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow \langle o_1, o_2 \rangle \in \mathtt{R'}^{\mathcal{I}(t-1)} \wedge \langle o_1, o_2 \rangle \notin \mathtt{R}^{\mathcal{I}(t-1)}$

Concerning the specific examples metioned in Section 2, they can be modelled with the following migration constraints:

Booking	RDEX	CheckIn,	Widowhood	RPEX	Widowhood,
Boarding	RDEV$^-$	CheckIn,	Represents	RQEX$^-$	WorksFor,
Divorce	RDEV$^-$	Marriage,	EssentialStructuralPart	RPEV	contained_in,
Marriage	SRDEX	Widowhood,	Disabled-SubQuantity	RDEV$^-$	SubQuantity.

Thus, e.g., the penultimate constraint assumes that in the domain of interest and normal course of operation, it holds that for each working candescent light bulb, the wolfram thread always will break at some point in the future, and remain broken indefinitely, and the latter constraint can be used for, e.g., the amount of wine poured into the wineglass which used to be a subquantity of the wine in the bottle.

We leave it to Human-Computer Interaction experts to devise the optimal way to add relation migration to the ORM graphical and textual languages: the constraints can be added alike rules are added in pseudo-natural language analogous to [2] or, e.g., with named dashed lines that serve as syntactic sugar for the axioms. We use the latter option in the remainder of the paper thanks to its compactness; an example is shown in Fig. 1.

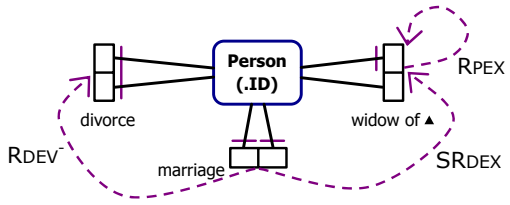

Fig. 1. Example of relation migration between various civil statuses (in ORM2 notation), where the dashed arrow denotes the direction of migration and its label the type of migration.

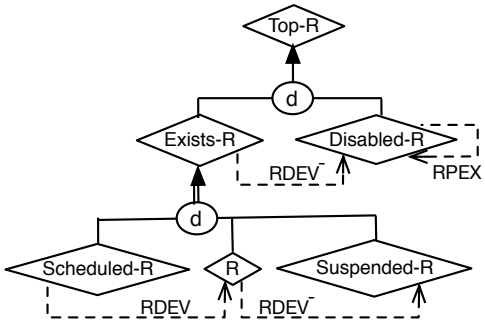

Fig. 2. Graphical depiction (in EER) of status relations (from [7]) extended with relation migration constraints between them, which are denoted with dashed lines and an arrow in the direction in which the relation migrates; e.g., a relation in `Disabled-R` was evolved dynamically (RDEV^{-}) from `Exists-R` (i.e., used to be an instance of `Exists-R`). In ORM notation, the diamonds are replaced by rectangles, encircled d with double shafted arrow with disjoint exclusive notation.

3.2 Lifespan and related notions

The lifespan of an object with respect to a class describes the temporal instants where the object can be considered a member of that class [1]. In an analogous way, we introduce the lifespan and related notions for temporal relations. The lifespan of a particular relation instance (tuple, fact) $r = \langle o_i, o_j \rangle$ (for simplicity we consider binary relations) with respect to a relation R describes the temporal instants where the particular relation instance can be considered a member of the relation. With the notion of status relations, one can distinguish between the following aspects of lifespan: EXISTENCESPAN_R, LIFESPAN_R, ACTIVESPAN_R, BEGIN_R, BIRTH_R and DEATH_R, which are functions that depend on the relation's membership to the status relation associated to a temporal relation R.

The *existencespan* of a relation instance r describes the temporal instants where the relation is either a scheduled, active or suspended member of a relation (i.e., of `Scheduled-R`, `R`, or of `Suspended-R`). Recollecting the relational hierarchy in Fig. 2, where `Exists-R` subsumes the former three, then we have $\text{EXISTENCESPAN}_R : \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}} \rightarrow 2^{\mathcal{T}}$, such that:

$$\text{EXISTENCESPAN}_R(r) = \{t \in \mathcal{T} \mid \langle o_i, o_j \rangle \in \texttt{Exists-R}^{\mathcal{I}(t)}\}$$

The *lifespan* of a relation instance describes the temporal instants where the relation is an active or suspended member of a given relation (thus, $\text{LIFESPAN}_R(r) \subseteq \text{EXISTENCESPAN}_R(r)$). More formally, $\text{LIFESPAN}_R : \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}} \rightarrow 2^{\mathcal{T}}$, such that:

$$\text{LIFESPAN}_R(r) = \{t \in \mathcal{T} \mid \langle o_i, o_j \rangle \in \texttt{R}^{\mathcal{I}(t)} \cup \texttt{Suspended-R}^{\mathcal{I}(t)}\}$$

The *activespan* of a relation instance describes the temporal instants where the relation is an active member of a given relation (thus, $\text{ACTIVESPAN}_R(r) \subseteq \text{LIFESPAN}_R(r)$). More formally, $\text{ACTIVESPAN}_R : \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}} \rightarrow 2^{\mathcal{T}}$, such that:

$$\text{ACTIVESPAN}_R(r) = \{t \in \mathcal{T} \mid \langle o_i, o_j \rangle \in \texttt{R}^{\mathcal{I}(t)}\}$$

The functions BEGIN_R and DEATH_R associate to a relation instance the first and the last appearance, respectively, of the relation as a member of a given relation, while BIRTH_R denotes the first appearance of the relation as an active member of that relation. More formally, BEGIN_R, BIRTH_R, $\text{DEATH}_R : \Delta^{\mathcal{B}} \times \Delta^{\mathcal{B}} \rightarrow \mathcal{T}$, such that:

$$\text{BEGIN}_R(r) = \texttt{min}(\text{EXISTENCESPAN}_R(r))$$
$$\text{BIRTH}_R(r) = \texttt{min}(\text{ACTIVESPAN}_R(r)) \equiv \texttt{min}(\text{LIFESPAN}_R(r))$$
$$\text{DEATH}_R(r) = \texttt{max}(\text{LIFESPAN}_R(r)) \equiv \text{EXISTENCESPAN}_R(r)$$

We could still speak of existencespan, lifespan or activespan for snapshot relations, but in this case $\text{EXISTENCESPAN}_R(r) \equiv \text{LIFESPAN}_R(r) \equiv \text{ACTIVESPAN}_R(r) \equiv \mathcal{T}$.

4 Logical consequences

In this section we show how further constraints can be derived as logical implications from the axiomatization proposed so far on relation migration (denoted with Σ_{st}) and the lifespan definitions. We will also study the interactions between object and relation migration.

The first result shows that mixing subtyping of entity types (ISA) with migration may result into inconsistent relations (see 'Case A' in Section 4.1 for an example).

Proposition 1 (ISA Vs. Relation Migration). *Let R, R' be two relations such that R' ISA R and R RMC R', with* RMC $\in \{$SRDEX, SRPEX, SRQEX, RDEV, RPEV, RQEV, RDEV^-, RPEV^-, $\text{RQEV}^-\}$ *then both R and R' are unsatisfiable, i.e., $\Sigma_{st} \cup \{R'$ ISA $R,\ R\,\text{RMC}\,R'\} \models \{R$ ISA $\bot, R'$ ISA $\bot\}$, with $\bot$ the empty set.*

Specifying constraints on relation migration can force new constraints on the lifespan of the related relations:

Proposition 2 (Migration Vs. Relation Lifespan). *Let R, R' be two relations, then:*

1. *If R RPEV R', then $\text{DEATH}_R(r) \leq \text{DEATH}_{R'}(r)$.*
2. *If $R\,\text{RMC}\,R'$, with* RMC $\in \{$SRDEX, SRPEX, SRQEX$\}$, *then* $\text{DEATH}_R(r) \leq \text{BIRTH}_{R'}(r)$.

To see why (2) holds, note that from the semantics of the relation migration involved in R RMC R' it immediately follows that (i) R and R' are two disjoint relations, and (ii) that R implies Scheduled-R$'$.

We now proceed to examine how the notion of lifespan for both classes and relations can interact with each other. Such interactions will be useful to understand how relation migration and object migration influence each other, which we address in the next subsection.

Proposition 3 (Objects Vs. Relations Lifespan). *Given the set of axioms Σ_{st} and the lifespan notions for both status classes and status relations, let R be an n-ary relation (where $n \geq 2$) to which object types $C_1, \ldots C_m$ participate (where $m \leq n$) and $r = \langle o_1, \ldots, o_n \rangle \in R$, then:*

1. ACTIVESPAN *of relations is shorter or equal to the* ACTIVESPAN *of objects participating in the relation.*
   $\text{ACTIVESPAN}_R(r) \subseteq \text{ACTIVESPAN}_{C_i}(o_i)$, *for* $i = 1, \ldots, n$
2. BEGIN *of objects occurs at the same time or before the* BEGIN *of the relation they participate in.*
   $\text{BEGIN}_{C_i}(o_i) \leq \text{BEGIN}_R(r)$, *for* $i = 1, \ldots, n$
3. DEATH *of relations occurs before or at the same time when one or more participating objects die.*
   $\text{DEATH}_R(r) \leq \text{DEATH}_{C_i}(o_i)$, *for* $i = 1, \ldots, n$

The first property is a consequence of the "ACT" axiom ($\langle o_1, o_2 \rangle \in \mathtt{R}^{\mathcal{I}(t)} \rightarrow o_i \in C_i^{\mathcal{I}(t)},\ i = 1, 2$) of the formalization of status relations [7] saying that active relations involve only active classes. From this property follows that $\text{BIRTH}_{C_i}(o_i) \leq \text{BIRTH}_R(r)$, for $i = 1, \ldots, n$. The second property is a consequence of the "RSUSP2" axiom ($\langle o_1, o_2 \rangle \in \mathtt{Suspended\text{-}R}^{\mathcal{I}(t)} \rightarrow o_i \in C_i^{\mathcal{I}(t)} \vee o_i \in \mathtt{Suspended\text{-}C_i}^{\mathcal{I}(t)},\ i = 1, 2$) [7] saying that objects participating in a suspended relation have to be scheduled or active objects. The last property is a consequence of the "RDISAB4" axiom [7] saying that disabled classes can participate only in disabled relations. As an obvious consequence we have that $\text{LIFESPAN}_R(r) \subseteq \text{LIFESPAN}_{C_i}(o_i)$ for $i = 1, \ldots, n$, similarly for EXISTENCESPAN.

4.1 Relation migration Vs. object migration

We consider now how the migration of a relation and the migration of a participating object can influence the temporal behaviour of each other. We show that, in addition to explicitly asserting an object- or relation- migration constraint, the constraints already expressed in a temporal conceptual model can force such kind of migrations. We have found two cases, depicted in Fig. 3, where (case A) object migration implies a relation migration, and (case B) relation migration implies an object migration. Note that the object migration constraints have a semantics similar to the one for relation migration. We assume that the relations R, R' are generic n-ary relations with the same arity and that the entity types E_1, E_2 play the same role/position in R and R', respectively.

Case A: Object migration → relation migration. Given an object migration between two entity types, E_1, E_2, participating in two relations, R, R', respectively, then to derive an analogous migration between the relations, the constraint expressed in Fig. 3-A must hold, where the timestamp (not drawn in Fig. 3) forces R to be time-invariant, called also snapshot in the literature (i.e., $\langle o_1, o_2 \rangle \in R^{\mathcal{I}(t)}$, then $\langle o_1, o_2 \rangle \in R^{\mathcal{I}(t')}$ for all $t' \in \mathcal{T}$). Note that if the specified object migration constraints are one of DEV/PEV/QEV then the diagram in Fig. 3-A would be inconsistent (see Proposition 1).

Case B: Relation migration → object migration. Given a migration between two relations, R, R', bounded to entity types, E_1, E_2, respectively, then to derive an analogous migration between the corresponding participating entity types, the constraints

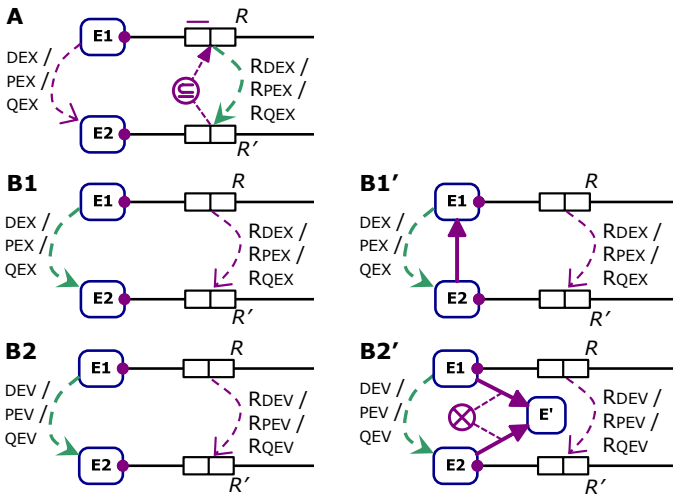

Fig. 3. Interaction between object- and relation migration; thin (pink) dashed line with open arrowhead = declared migration, thick (green) dashed line = implication for the relation or object. Observe that B1 includes cases like B1′. B2 vs. B2′ depends on the underlying formalism: E_1 and E_2 must be disjoint, which is either assumed to be so when the entity types are not in a hierarchy (B2, normally the case in conceptual models) or must added explicitly (when one principally uses an arbitrary suitable logic and uses an ORM diagram as 'syntactic sugar').

expressed in Fig. 3-B1/B1′ must hold; that is, E_1 and E_2 may, but do not need to be, disjoint. When the migration between the relations are one of RDEV/RPEV/RQEV, then to derive a similar object migration constraint, the entity types E_1, E_2 need to be disjoint in the conceptual diagram as shown in Fig. 3-B2 (if the underlying logic already takes care of disjointness of E_1 and E_2 by convention) or B2′ (declaring and communicating explicitly the disjointness of E_1 and E_2 in an ORM diagram). On the other hand, if the entity types E_1, E_2 are declared to be disjoint then, in its turn, the relation migration is forced to be one of RDEV/RPEV/RQEV.

We illustrate these two cases in the following example.

Example 3. For Case A, let us assume a company where, sooner or later, each employee—who works for exactly one department—will be promoted within the same department he or she works for and such that demotion does not occur. This means an object migration of type PEX between Employee and Manager; see Fig. 4-A. To maintain consistency of the model, this forces a relation migration of type RPEX between works for and manages.

For Case B, let us recollect the example of Section 2 about `John` as CEO and his membership in the CompanyBoard, which has a business rule that at some point he will cease to be a CEO and full member of the board but then must continue consulting the company board to foster continuation and smooth transition of the management, i.e., we have RDEV between member of and consultant of. In the ORM model depicted in Fig. 4-B, this forces `John` to dynamically evolve (DEV) from CEO to Consultant.

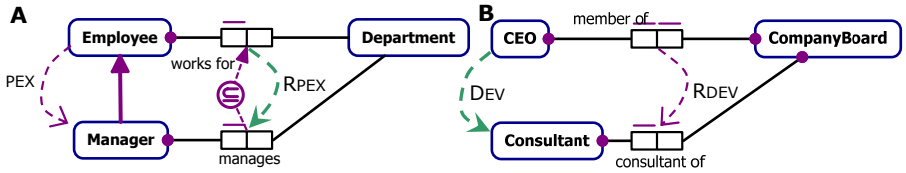

Fig. 4. Examples of interaction between object- and relation migration.

5 Conclusions

We have identified different ways how a relation can change over time and given a logic-based semantics to the notion of *relation migration* to capture its behaviour precisely. Relation migration imposes constraints regarding the notions of lifespan of a relation, in a similar way as object migration does for lifespans of objects. In addition, we explored the interaction that exists between object- and relation migration. We presented two different cases showing how object migration influences relation migration and, vice-versa, how from a relation migration we can derive migration constraints for the classes participating in the relation.

Current and future work involves integrating aspects of rigidity, its effects on migrations between part-whole relations, and tractable reasoning with relation migration.

References

1. Artale, A., Parent, C., Spaccapietra, S.: Evolving objects in temporal information systems. Annals of Mathematics and Artificial Intelligence **50**(1-2) (2007) 5–38
2. Balsters, H., Carver, A., Halpin, T., Morgan, T.: Modeling dynamic rules in ORM. In Meersman, R., Tari, Z., Herrero., P.e.a., eds.: OTM Workshops 2006. Volume 4278 of LNCS., Springer-Verlag (2006) 1201–1210
3. Hall, G., Gupta, R.: Modeling transition. In: Proc. of ICDE'91. (1991) 540–549
4. Etzion, O., Gal, A., Segev, A.: Extended update functionality in temporal databases. LNCS. In: Temporal Databases - Research and Practice. Springer-Verlag (1998) 5695
5. Halpin, T.: Temporal modeling and ORM. In Meersman, R., Tari, Z., Herrero., P., eds.: OTM 2008 Workshops. Volume 5333 of LNCS., Springer (2008) 688–698
6. Parent, C., Spaccapietra, S., Zimányi, E.: Conceptual modeling for traditional and spatio-temporal applications—the MADS approach. Berlin Heidelberg: Springer Verlag (2006)
7. Artale, A., Guarino, N., Keet, C.M.: Formalising temporal constraints on part-whole relations. In Brewka, G., Lang, J., eds.: 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08), AAAI Press (2008) 673–683
8. Artale, A., Keet, C.M.: Essential, mandatory, and shared parts in conceptual data models. In Halpin, T., Proper, H., Krogstie, J., eds.: Innovations in Information Systems modeling: Methods and Best Practices. IGI Global (2008) 17–52
9. Keet, C.M., Artale, A.: Representing and reasoning over a taxonomy of part-whole relations. Applied Ontology **3**(1-2) (2008) 91–110
10. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Phd thesis, University of Twente, The Netherlands. Telematica Instituut Fundamental Research Series No. 15 (2005)