Available online at www.sciencedirect.com

ScienceDirect

Information Sciences 00 (2019) 1–29

Information Sciences

Enabling Instant- and Interval-based Semantics in Multidimensional Data Models: the T+MultiDim Model

Carlo Combi[a], Barbara Oliboni[a], Giuseppe Pozzi[b], Alberto Sabaini[a], Esteban Zimányi[c]

[a] *Dipartimento di Informatica, Università degli Studi di Verona, strada le Grazie 15, I-37134 Verona, Italy, name.surname@univr.it*

[b] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, p.za L. da Vinci 32, I-20133 Milano, Italy, name.surname@polimi.it*

[c] *Department of Computer and Decision Engineering, Université libre de Bruxelles, Avenue F. D. Roosevelt 50, B-1050 Bruxelles, Belgique, ezimanyi@ulb.ac.be*

Abstract

Time is a vital facet of every human activity. Data warehouses, which are huge repositories of historical information, must provide analysts with rich mechanisms for managing the temporal aspects of information. In this paper, we (i) propose T+MultiDim, a multidimensional conceptual data model enabling both instant- and interval-based semantics over temporal dimensions, and (ii) provide suitable OLAP (On-Line Analytical Processing) operators for querying temporal information. T+MultiDim allows one to design typical concepts of a data warehouse including temporal dimensions, and provides one with the new possibility of conceptually connecting different temporal dimensions for exploiting temporally aggregated data. The proposed approach allows one to specify and to evaluate powerful OLAP queries over information from data warehouses. In particular, we define a set of OLAP operators to deal with interval-based temporal data. Such operators allow the user to derive new measure values associated to different intervals/instants, according to different temporal semantics. Moreover, we propose and discuss through examples from the healthcare domain the SQL specification of all the temporal OLAP operators we define.

Keywords: Data warehouses, temporal dimensions, OLAP (On-Line Analytical Processing), instant-based semantics, interval-based semantics, telic vs atelic facts

1. Introduction

Conceptual data modeling provides analysts and designers with a high-level representation of the real world and an efficient way to communicate among each other. Conceptual data models promote understanding of the real-world domain [15] and enhance the ability to meet users requirements [3]. One of the major advantages in using conceptual models is that the communication between users and designers is made easier. Users are not required to own skills about the implementation platform. Moreover, a conceptual model abstracts from the underlying technology, allowing designers to use various logical models. A conceptual schema is a concise description of the users data requirements, without taking into account implementation details [30]. One important issue to be faced is the lack of a universally-adopted, conceptual model for multidimensional data [38], usually stored in data warehouses. Consequently, data warehouses are often

directly designed at the logical level by following the well-known paradigm of star and/or snowflake schema. Such schemata are hard to understand to most users, and are often technology dependent [30].

An essential feature of data warehouses is the presence of multiple temporal dimensions. Traditional data warehouse models use an instant-based semantics [16, 30] by which data are interpreted as a sequence of states, where the states are independent from each other. Consider for example a data warehouse with a fact Sales that is related to two temporal dimensions OrderDate and ShippedDate. These dimensions are independent and thus, a measure such as Amount can be analyzed and aggregated along one of these temporal dimensions. On the other hand, an interval-based semantics allows one to reason on facts and their time span, since it deals with time intervals instead of time instants. For example, suppose that in a data warehouse a fact Treatment is used for analyzing the medical treatments administered to patients. Such a fact may have dimensions Patient and Drug in addition to temporal dimensions such as DrugStart, DrugEnd, EffectStart, and EffectEnd. Obviously, these temporal dimensions are not independent and define various intervals that can be used to analyze and aggregate a measure such as Dosage.

As pointed out in [21], some temporal phenomena cannot be adequately modeled through instant-based semantics. In particular, *telic* [2] facts characterized by an intrinsic *goal* or *culmination* require an interval-based semantics, while the traditional instant-based semantics is adequate for *atelic* facts. The importance of dealing with telic facts has been widely recognized in many different areas, spanning from artificial intelligence to philosophy. As an example, the sentence "the patient had a complete antibiotic therapy from September 3, 2018 to September 11, 2018" represents a telic fact. Indeed, we cannot assert that the patient had a complete therapy during any subinterval of the given interval. Instead, the sentence "the patient had fever from October 1, 2018 to October 4, 2018" refers to an atelic fact, as we can say that the patient had fever during any subinterval of the period between October 1 and October 4.

We argue that, in a data warehouse context, the use of both instant-based and interval-based semantics would benefit users. Temporal dimensions describe different aspects of the modeled reality, but they are often individually taken into account, while analysts need to combine different temporal aspects, in order to have insights on data during analysis. Unfortunately, the standard OLAP (On Line Analytical Processing) operators do not provide users with such capabilities. Even though in the database field several important issues relate to the temporal dimension of information, and thus different approaches on temporal evolution of data or schema versioning have been proposed, little attention has been so far paid to modeling specific semantics of temporal dimensions and on specifying time-oriented OLAP operators. In particular, telic and atelic facts represented in a data warehouse are related to measures, i.e. quantitative features. Thus, we need to explicitly address how telic and atelic measures have to be processed when aggregation/selection queries require the computation of new telic/atelic measures associated to new intervals. As an example, the atelic fact "the patient had a temperature of 39 from October 1, 2018 to October 4, 2018" may correspond to the (same) fact "the patient had a temperature of 39 from October 2, 2018 to October 3, 2018", if we are querying the data warehouse about that period of two days. On the other hand, the telic fact "the patient had a complete antibiotic therapy, consisting of 3600 mg of cefixime, from September 3, 2018 to September 11, 2018" may correspond to the atelic fact "the patient had a (partial) antibiotic therapy, consisting of 800 mg of cefixime, from September 4, 2018 to September 5, 2018", when we are interested to that period of two days. Moreover, if we assume to ignore the daily dosage of cefixime, we could also associate the given fact to the atelic fact "the patient had a (partial) antibiotic therapy, consisting of less than 3600 mg of cefixime, from September 4, 2018 to September 5, 2018".

In this paper, we propose new solutions to two different, but intertwined, aspects in modeling and managing temporal multidimensional data.

- We propose T+MultiDim, a temporal multidimensional data model that extends the MultiDim data model [30, 39]. It allows the specification of both instant- and interval-based semantics for temporal dimensions. T+MultiDim enables the designer to specify an explicit link between couples of instant-based temporal dimensions, considering these dimensions as interval bounds. T+MultiDim conceptual schemata may be suitably mapped to the classical logical representation based on relational star/snowflake schemata.

- We formally introduce new OLAP operators dealing with such instant and interval-based temporal

dimensions and allowing users to perform temporal aggregate queries on multidimensional data, considering both telic and atelic semantics. We explicitly focus on the derivation of new telic facts from the ones already stored in the data warehouse, according to different kinds of query. Moreover, we specify how to express these new OLAP operators by SQL queries.

It is worth noting that the idea behind our approach is not specific to the MultiDim conceptual model, and other conceptual multidimensional models may be similarly extended as well. Furthermore, without loss of generality, we will discuss all the introduced concepts and techniques through a real-world motivating example, taken from the domain of pharmacovigilance, to show the practical usefulness and applicability of our proposal.

The paper is organized as follows. Section 2 introduces some basic notions useful for understanding the context of the proposed approach. Section 3 describes a real-world scenario from the medical field, which is used throughout the paper to emphasize the need of a connection between temporal dimensions. Section 4 defines the multidimensional conceptual data model considered throughout the paper and defines the concept of link between two temporal dimensions. Section 5 defines some new OLAP operators that explicitly deal with interval-based dimensions. Section 6 sketches the specification of the new OLAP operators in SQL. Section 7 discusses related work and compares our proposal with approaches presented in literature. Finally, Section 8 draws some conclusions and sketches out some future research directions.

2. Background

In this section, we will briefly introduce the data warehouses context and the related multidimensional data models. Then, we will describe the conceptual model we extend in this paper and motivate our main contributions pertaining to the conceptual modeling of temporal multidimensional data and to the analysis of such data.

2.1. Multidimensional data model

Data warehouses are huge repositories of historical and integrated information. Data warehousing techniques provide analysts with rich mechanisms for extracting information to be used as a support in decision-making activities. In this context, temporal aspects of data are of primary importance.

The multidimensional data model is based on cubes, measures, dimensions, hierarchies, levels, and attributes. A cube represents a fact of interest. Instances of a fact correspond to occurred events that can be analyzed [30, 39]. Measures provide a quantitative description of events, since each fact is described by relevant measures. Dimensions represent the possible perspectives of analysis of a fact. Time is usually one of the considered dimensions. Dimensions are related to the concept of multidimensional space for representing multidimensional data. Each dimension is related to a hierarchy of aggregation levels. Each level represents a position in the hierarchy, and consists of a dimensional attribute. The multidimensional model defines a representation for describing real-world business events. Analysts may analyze business events by considering measures they are interested in and by examining dimensions and attributes that make the data meaningful.

2.2. TheMultiDim multidimensional conceptual model

The MultiDim model [30, 39] allows analysts and designers to represent at a conceptual level all the elements required in data warehouse. The graphical notation of the MultiDim model, shown in Figure 1, resembles the one of the ER data model [8].

A fact is represented by a cube and contain measures, which are numerical data that can be analyzed according to dimensions. Measures are usually classified according to additivity [30]: *additive* measures can be summarized through all of their dimensions; *semi-additive* can be summarized on every dimension but time; *non-additive* measures cannot be summarized through any dimension. As an example, the quantity of an administered drug is additive, as it is meaningful to consider the overall quantity of drugs administered to

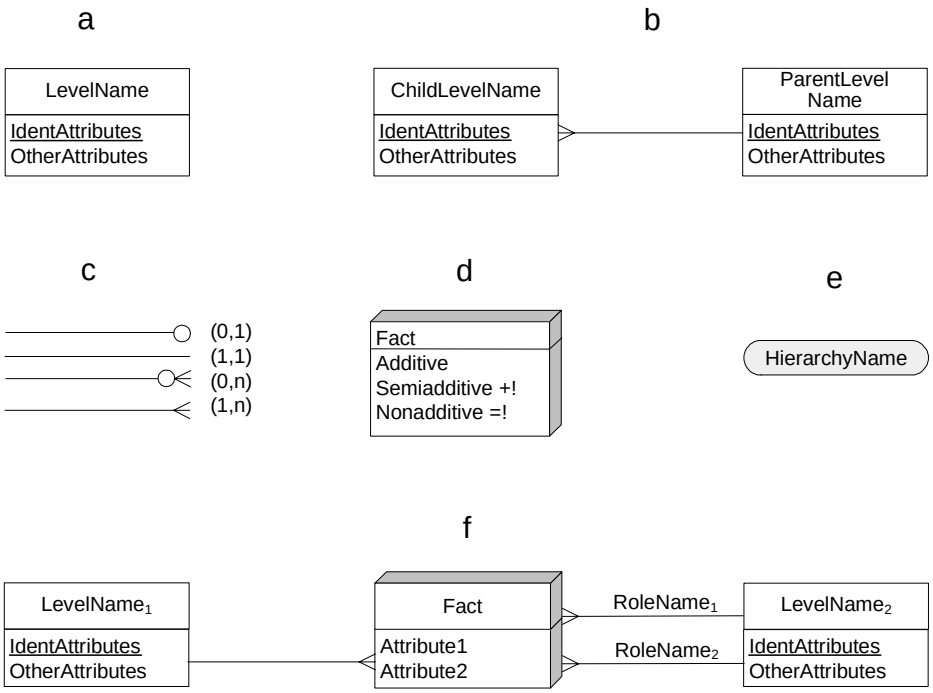

Figure 1. Notation of the MultiDim model. (a) Level. (b) Hierarchy. (c) Cardinalities of relationships between levels and between levels and fact. (d) Specification for types of measures (depicted inside a fact). (e) Hierarchy name. The oval is attached to the finest element of a dimension hierarchy, to highlight possibly multiple hierarchies for the same dimension (further examples in Figure 2). (f) Fact with measures (i.e.Attribute1, Attribute2) and associated levels (through dimension roles).

patients, for diagnoses, and for a certain period. On the other hand, the number of drugs administered daily by patients is semi-additive as it is meaningful to consider the average number of drugs administered daily to patients for diagnoses, but it is meaningless to sum such numbers for different days. Finally, the temperature measured on patients is non-additive, as summing up patient temperatures is meaningless with respect to any dimension. By default, measures are assumed to be additive. Semi-additivity or non-additivity measures are identified by an extra symbol, +! and =!, respectively.

A fact relates several levels. Levels containing the most detailed data in a hierarchy are called leaf level. Leaf level names define the dimension name, unless one or more roles are specified for that dimension. Likewise, levels containing the most general data are called root levels. A dimension may contain several hierarchies, each one expressing a particular criterion used for analysis purposes. Hierarchy names are included in order to differentiate aggregation paths.

A level may play different roles in a fact. Each role is identified by a name and is represented by a separate link between the corresponding level and the fact, as depicted in Figure 1f. A hierarchy comprises several related levels, as depicted in Figure 1b. Given two related levels of a hierarchy, the lower level is called child and the higher one parent. Thus they form parent-child relationships whose cardinalities, as for facts, indicate the minimum and the maximum number of members in one level that can be related to a member in another level. Aggregation functions may be associated with a measure by specifying them next to measure names, as depicted in Figure 1e. The cardinality of the relationship between facts and levels indicates the minimum and the maximum number of fact members that can be related to level members, as shown in Figure 1c.

In this work, we extend the MultiDim model by defining suitable connections between different temporal dimensions. Connecting temporal dimensions allows data analysts to navigate data cubes by using interval-based temporal dimensions, thus viewing and analyzing data that otherwise would not be available.

3. Motivating Scenario

In this section, we use a real-world scenario from the pharmacovigilance domain to emphasize the need of a connection between temporal dimensions in a multidimensional schema. Pharmacovigilance is the activity related to the collection, analysis, and prevention of adverse drug reactions. Indeed, unexpected adverse reactions may go undetected and only become observable when the drug reaches the general population. Therefore, it is necessary to control drugs even after putting them on sale. This practice is invaluable, provides early warnings, and requires limited economic and organizational resources. To this end, we often need to store and analyze data about patients' treatments, observed adverse reactions to drugs, and so on [10, 12, 29]. The example focuses on treatments, only. From an analysis perspective, we can identify a main *fact* of interest called Treatment. A treatment is characterized by a patient, an administered drug, a cost, and a daily dosage. In addition, a treatment is related to four instant-based temporal *dimensions*: start of administration, end of administration, start of effect, and end of effect. We can also identify interval-based dimensions, in particular the drug administration and the effect intervals (i.e., periods), each of them composed by a start and an end timestamp.

Figure 2 shows the multidimensional conceptual schema of the scenario represented with the MultiDim conceptual model notation. The Treatment fact relates the Patient, Drug, and Time dimensions, and it has two *measures*, Cost, and DailyDosage, where the notation "+!" means that the latter measure can be partially aggregated. The considered fact has six dimensions: namely, Drug, Patient, and the four role-playing dimensions DrugStart, DrugEnd, EffectStart, and EffectEnd over the Time dimension). Dimension Drug uses the hierarchy ATC, named after the usual classification Anatomical Therapeutic and Chemical (ATC), adopted in the pharmacology domain [35].

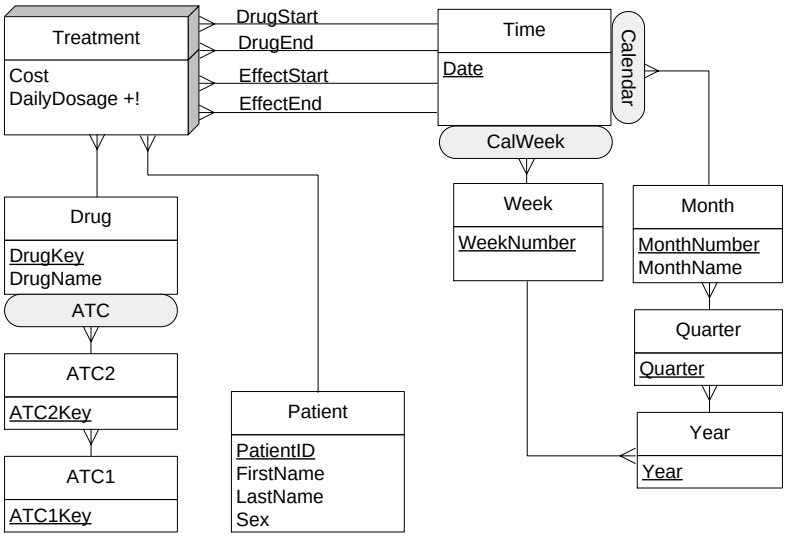

Figure 2. A multidimensional schema that represents treatments

Table 1 depicts an instance of Treatment fact represented at a logical level as a relation.

From now on, we make some assumptions on the data from Table 1. For example, let us consider first tuple. According to the given assumptions, it asserts that:

i. the patient took the first pill on Aug. 16, 2018;
ii. the patient took the last pill on Sept. 07, 2018;
iii. the effect of the administration started on Sept. 03, 2018;
iv. the effect of the administration ended on Sept. 12, 2018;
v. the overall cost of the treatment was 65.

Patient	Drug	Cost	DailyDosage	DrugStart	DrugEnd	EffectStart	EffectEnd
P1	Tylenol	65	40	16-08-2018	07-09-2018	03-09-2018	12-09-2018
P1	Tylenol	20	20	10-09-2018	14-09-2018	13-09-2018	05-10-2018
P2	Aspirin	80	30	01-09-2018	08-10-2018	06-09-2018	08-10-2018
P3	Tylenol	60	30	04-09-2018	19-09-2018	12-09-2018	18-09-2018
P4	Tylenol	30	40	20-08-2018	28-08-2018	22-08-2018	06-09-2018
P4	Aspirin	35	50	13-09-2018	20-09-2018	20-09-2018	02-10-2018

Table 1. Tuples of the Treatment fact relation from a pharmacovigilance database

From the above, we argue that:

- the duration of the administration (DrugPeriod) is 23 days, which shall be expressed as (DrugEnd - DrugStart + 1). The closed interval is [DrugStart, DrugEnd]: Aug. 16 and Sept. 07 are both counted, as we have closed intervals;
- the delay of the effect (EffectDelay) is 18 days, which shall be expressed as (EffectStart - DrugStart). The closed interval is [DrugStart, EffectStart - 1]. Aug. 16 (the first day of the administration, i.e., DrugStart) and Sept. 02 (the last day of EffectDelay, i.e., EffectStart - 1) are both counted, as we have closed intervals. Sept. 03 does not belong to the effect delay interval, as Sept. 03 is the first day when the effect is observed;
- the daily cost of the treatment is computed as Cost/DrugPeriod, i.e. Cost/(DrugEnd − DrugStart + 1).

Multidimensional structures can be queried by using OLAP tools to retrieve information such as:

1. *Compute the maximum daily dosage administered to a patient, grouping treatments according to the month when the drug administration began.*
2. *Compute the total cost of treatments for every drug.*

These are standard OLAP queries, where all the dimensions can be used for filtering or grouping purposes. Temporal dimensions can be considered as yet another dimension upon which a total order has been defined. For example, when applying query 1. to data of Table 1, we obtain two groups of tuples with respect to the month when the drug administration began, i.e., a group for August, and a group for September. Within each group, the maximum daily dosage is computed, and the result is: 50 for August and 30 for September.

The combination of two temporal dimensions can be considered as one time period, i.e., an interval. Grouping according to interval-based temporal dimensions is one of the phases to compute temporal aggregates over available data. This topic has been tackled for years in the database community, both from a modeling and implementation perspective [6, 7, 17]. We focus here on the application of these techniques in multidimensional structures. Data stored in Table 1, according to the Treatment schema depicted in Figure 2, can be queried with respect to interval-based temporal dimensions to retrieve information such as:

3. *Compute the total cost of therapies administered to patients on September 13.*
4. *Compute the total cost of therapies administered to patients during September.*
5. *Compute for every drug the total number of administrations.*
6. *Compute the maximum dosage per drug per month across all patients.*

These queries are not standard OLAP queries. We can identify several issues in computing such queries by means of standard OLAP operators. First of all, treatments can last several days and thus two dimensions must be used for temporally bounding them. This raises the issue of selecting the temporal dimensions to characterize treatment validity intervals. In Figure 2, there are four relationships between the Treatment fact and the Time dimension, namely, DrugStart, DrugEnd, EffectStart, and EffectEnd. Some of these relationships can be meaningfully combined in order to obtain a validity interval for treatments, while other combinations, e.g., the couple ⟨DrugEnd, DrugStart⟩, can lead to mostly negative validity intervals.

By switching to interval-based reasoning we also address the aggregation issues when dealing with telic and atelic measures. As an example, if we associate every treatment to an interval, we must switch from an instant-based semantics to an interval-based one. While measure DailyDosage is atelic and thus it holds at any day of the period ⟨DrugStart, DrugEnd⟩, measure Cost is telic, and it holds over the whole specified period and not over its sub-periods. Such a measure does not reflect the cost spent to provide patients with treatments for a single day as in the first query or during a month as in the second query. Therefore, the values should be adjusted accordingly to correctly represent treatment costs for a specific subinterval. In our approach, we propose some solutions allowing the derivation of new telic measure values, according to specific queries, by considering the original telic values in a data warehouse. Usually such telic measure values hold on some subintervals of the overall intervals associated to the original measure values.

For example, the third query can be computed in two different ways, depending on the adopted semantics. If we consider an instant-based semantics, data are grouped according to the administered Drug: the number of administrations is computed for each group and returned as the result. If we consider an interval-based semantics, temporal aspects of data must be taken into account. In this case, Instant Temporal Aggregation (ITA) [6] must be applied to evaluate the query. It computes the aggregate value at a given time instant t by considering the set of tuples whose timestamp contains t.

The fourth query can also be computed in two different ways, depending on the adopted semantics. If an instant-based semantics is considered, data are grouped according to the administered Drug and one of the instant-based temporal dimensions at the month level. Then, the maximum dosage is computed for each group and returned as the result. However, temporal aggregation techniques must be considered when dealing with interval-based semantics. In this case, Span Temporal Aggregation (STA) [6] must be applied to properly compute the result. STA allows one to group tuples when the related intervals overlap the considered month.

When measures are aggregated using interval-based semantics, we must distinguish between atelic and telic measures. Atelic measures satisfy the *downward hereditary* property while telic measures do not [33]. This property asserts that when a fact f holds over a time interval i, we can say that f holds over any subinterval of i. An example of an atelic measure is the balance of bank accounts, since if an account has a balance of $100 over a time interval, then the account has the same balance at any day included in that interval. On the other hand, telic measures are not downward hereditary. For example, the measure Cost of Treatment fact is telic, since the cost in a subinterval will be less than the overall cost. In this case, appropriate *measure adjustment* functions must be defined for determining the value of the measure in any subinterval, and this depends on the application domain at hand. In this paper, for conciseness reasons, we only consider *linear* measure adjustment functions, although our approach can be generalized for arbitrary functions.

Most multidimensional models allow designers to represent only instant-based temporal dimensions. As we pointed out in the previous examples, both instant-based and interval-based semantics should be considered when computing queries involving temporal dimensions. Domain expert reasoning is, indeed, based on instantaneous and interval facts. In the following, we will discuss further specific queries similar to the ones previously discussed, to show how our proposal allows the user to specify instant- and interval-based temporal queries for multidimensional data.

4. The Multidimensional Model T+MultiDim

Temporal dimensions play several roles for a fact, capturing different temporal aspects of the modeled reality. In the following, we assume that a multidimensional conceptual schema contains a special temporal dimension called Time, composed by at least Day, Month, and Year levels, and has a hierarchy called Calendar, which defines a roll-up relation between (Day, Month) and (Month, Year). Roles played by temporal dimensions represent different temporal coordinates of a fact. We name these roles as *role-playing temporal dimensions* (RPTDs). We argue that RPTDs can be combined to represent the temporal extent of a fact, focusing on the importance of interval-based temporal analysis of multidimensional structures.

In this section, we formally introduce a multidimensional data model, based on the notion of dimensions, measures, and facts. We start from the MultiDim model, and introduce links between couples of RPTDs that

share the same structure. Each dimension is organized in a hierarchy of levels, which allows us to summarize the measures in facts at different granularities. Within a dimension, values of a finer granularity can roll up to values of a coarser one. A multidimensional schema consists of facts defined with respect to a particular combination of levels. A multidimensional instance associates measures with dimension coordinates in each fact. A dimension may participate in a fact multiple times with different roles.

For simplicity, and without loss of generality, we assume that dimensions names are unique. We define next multidimensional schemas.

Definition 1. Multidimensional Schema

- A multidimensional schema $\mathcal{MS}$ is composed by a set of levels $\mathcal{L} = \{L_1, \ldots, L_n\}$, a set of dimensions $\mathcal{D} = \{D_1, \ldots, D_m, \mathsf{Time}\}$, and a set of facts $\mathcal{F} = \{F_1, \ldots, F_o\}$.
- A level $L \in \mathcal{L}$, is defined by a schema $\langle name, A_1 : dom_1, \ldots, A_n : dom_n \rangle$, where *name* is the level name, and each attribute A_j is defined over the domain dom_j. The special level $All \in \mathcal{L}$, does not have any attribute. The level name *name* is unique in $\mathcal{MS}$, and the attribute name A_j is unique in L.
- A dimension $D_i \in \mathcal{D}$ has a unique name and is composed of a set of levels $\mathcal{L}_i\{L_{i1}, \ldots, L_{in}, All\}, \mathcal{L}_i \subseteq \mathcal{L}$, and a set of hierarchies $\mathcal{H}_i = \{H_{i1}, \ldots, H_{ik}\}$.
- A hierarchy $H \in \mathcal{H}_i$ is defined by the schema $\langle name, \mathcal{S}_i, R_{name} \rangle$. *name* is the hierarchy name and it is unique in $\mathcal{MS}$. A hierarchy is composed by a set $\mathcal{S}_i$ of levels of dimension D_i, $\mathcal{S}_i \subseteq \mathcal{L}_i$. The roll-up relation R_{name} consists of a set of triples $\langle L_j, L_k, card \rangle$, where L_j and $L_k \in \mathcal{S}_i$. $card \in \{1\text{-}1, 1\text{-}m, m\text{-}m\}$ denotes the cardinality of the link between the child level L_j and the parent level L_k. The roll-up relation R_{name}, includes *All* as parent in some triple. Moreover, the *All* level is, directly or transitively, accessible from all levels of the hierarchy.

  Dimension Time has the associated hierarchy Calendar, composed by Day, Month, and Year levels. A second hierarchy for Time could be named CalWeek, and it could be composed by Day, Week, and Year. In such a dimension, weeks are numbered within a year. The first and the last week in a year might be composed by less than 7 days[1]. According to the above definitions, these two hierarchies may be defined as follows:

$$\begin{aligned} \mathcal{H}_{\mathsf{Time}} = &\ \{\langle \mathsf{Calendar}, \{\mathsf{Day}, \mathsf{Month}, \mathsf{Year}\}, R_{\mathsf{Calendar}} \rangle, \langle \mathsf{CalWeek}, \{\mathsf{Day}, \mathsf{Week}, \mathsf{Year}\}, R_{\mathsf{CalWeek}} \rangle\} \\ R_{\mathsf{Calendar}} = &\ \{\langle \mathsf{Day}, \mathsf{Month}, \text{1-m} \rangle, \langle \mathsf{Month}, \mathsf{Year}, \text{1-m} \rangle, \langle \mathsf{Year}, \mathsf{All}, \text{1-m} \rangle\} \\ R_{\mathsf{CalWeek}} = &\ \{\langle \mathsf{Day}, \mathsf{Week}, \text{1-m} \rangle, \langle \mathsf{Week}, \mathsf{Year}, \text{1-m} \rangle, \langle \mathsf{Year}, \mathsf{All}, \text{1-m} \rangle\} \end{aligned}$$

- A fact $F_i \in \mathcal{F}$ is defined by the schema $(K, \{\langle L_1, card_1, N_1 \rangle, \ldots, \langle L_m, card_m, N_m \rangle\}, \{\langle K_i, N_j, N_k, T_i, bound_i \rangle, \ldots, \langle K_h, N_p, N_q, T_h, bound_h \rangle, \texttt{with } j \neq k, p \neq q\}\{\langle M_1 : dom_1, add_1 \rangle, \ldots, \langle M_n : dom_n, add_n \rangle\})$.

  A fact has an attribute K that uniquely identifies a fact instance, i.e., it functionally determines all the dimension roles and the measures of each cell. A fact instance is characterized by a set of coordinates and a set of measures. A coordinate is a level L_j of a dimension, and it is the finest granularity on which measures are captured on that dimension. Indeed, a fact can be connected to any of the levels of a dimension. As an example, sporadic events can be captured by a temporal dimension at the month level, rather than at the day level. *card* indicates the cardinality of the relationship between the dimensions and the fact and is one of $1\text{-}1, 1\text{-}m$, or $m\text{-}m$. The same level can participate several times in a fact, playing different roles. N_i is a name of the role played by a dimension, starting from level L_j. As an example, a temporal dimension could play two different roles, one starting at the day level, and the other starting at the month level.

  Each measure M_i is defined over the domain dom_i. Measure types are additive (i.e., they can be meaningfully summarized along all the dimensions, using addition), semi-additive (i.e., they can be

[1]For the sake of simplicity, we will consider in this paper only such two time hierarchies. Further granularities may be suitably modeled, even with gaps and non convex granules [9]. In any case, we assume that all the date values stored for some fact instance must belong to all the considered time hierarchies.

meaningfully summarized using addition along some, but not all, the dimensions), and non-additive (i.e., they cannot be meaningfully summarized using addition across any dimension). The role names N_i and the measures names M_j are unique in F_i.

- An explicit link existing between two RPTDs is defined by the schema $\langle K_i, N_j, N_k, T_i, bound_i \rangle, j \neq k$. K_i is its unique name, and it represents a connection between the ordered couple of RPTDs N_j, N_k, i.e., levels of dimension Time. The two roles must refer to the same level, meaning that time must be represented at the same granularity within the same temporal dimension. Furthermore, instances b_j of N_j must be connected only to instances b_k of N_k such that $b_j \leq b_k$. Thus, the corresponding (closed) interval is $[b_j, b_k]$, having length $(b_k - b_j + 1)$. At last, the semantic of measures must be properly managed. To this purpose, we define the set $T_i \subseteq M$, which is composed by the fact measures that should be considered as telic ones over the new interval. $bound_i \in \{++, +-, -+, --\}$ specifies whether the given upper and lower bounds must contain the specified temporal values, respectively.

As pointed out by Terenziani et al. [37], by using an instant-based semantics, data are interpreted as a sequence of states indexed by points in time. Each state is independent from every other state. By considering two time points as an interval, the instant-based semantics no longer applies. Indeed, facts are now associated with sets of time instants, which are the temporal extent where a fact is now described. The telic aspects need to be considered, since the downward property could not hold over all the fact measures. The telic property expresses the completeness of a fact over an interval. As an example, an overall cost of 100.00 for a 10 days therapy, does not reflect the cost of the first 5 days.

We now extend the notation of the MultiDim model by introducing links between RPTDs. Two temporal dimensions can be connected by a link. Such a connection is characterized by a *name* and a *direction*. The combination of two connected temporal dimensions will create an interval-based dimension sharing the same name as the link one. Moreover, the link direction determines which instant-based dimension will be considered as interval starting point, and which one as interval ending point.

The extended notation is depicted in Figure 3.

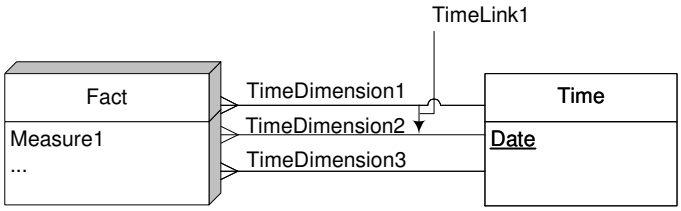

Figure 3. Extension of the MultiDim notation to represent connections between time dimensions. The connection between TimeDimension1 and TimeDimension2 by TimeLink1 means that the two temporal dimensions can be combined together to create a derived interval-based dimension

TimeDimension1 is connected with TimeDimension2 by TimeLink1. This means that TimeDimension1 and TimeDimension2 can be combined to create a derived interval-based dimension called TimeLink1. Its instances will be intervals composed by TimeDimension1 members, considered as the starting point, and TimeDimension2 members considered as ending point.

Five time links have been added to the example of Figure 2, as depicted in Figure 4.

DrugPeriod and EffectPeriod are two obvious links, as defined by their temporal dimension names. On the other hand, EffectDelay, ResidualPeriod, and TreatmentPeriod are strongly domain dependent: EffectDelay represents the amount of time between the drug administration and its absorption (i.e., the process of a substance entering the blood circulation. Once the administered drug has been absorbed, it starts to produce its effects on the treated patient); ResidualPeriod represents the amount of time between the end of drug administration and the end of its effect (i.e., once the drug administration stops, there is a period where the effects still hold). Finally, TreatmentPeriod represents the overall period of a treatment, that is from the start of the treatment to the end of residual effect of the administered drug.

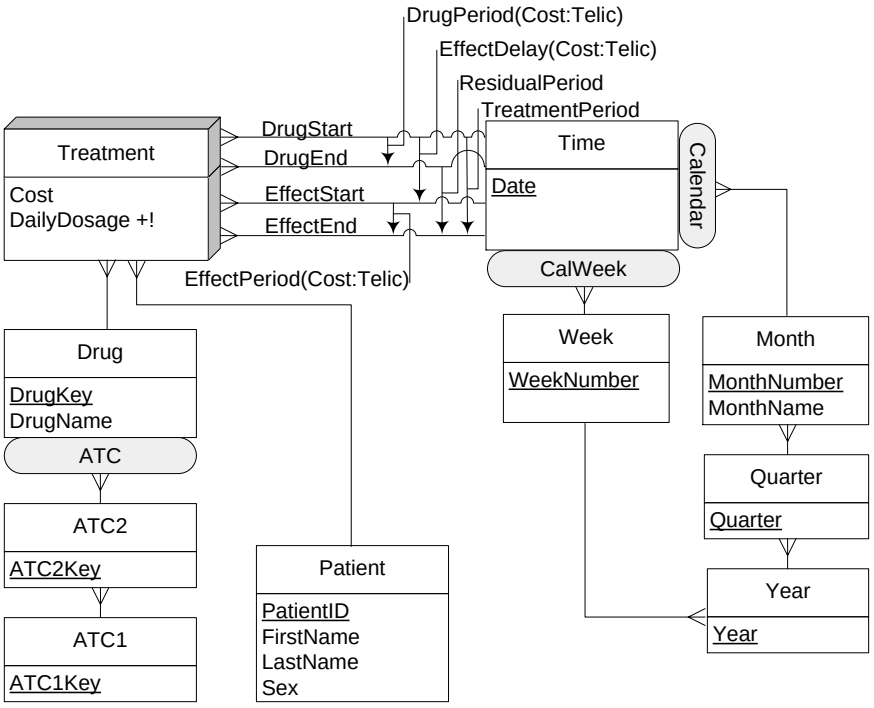

Figure 4. A multidimensional schema that represents treatments (i.e., the Treatment fact). The Time dimension plays different roles, which are linked in order to set up temporal intervals. Different arrows specify different kinds of intervals, according to Definition 1. Standard arrows correspond to ++ intervals; arrows with a filled circle correspond to +− intervals; arrows with a filled circle correspond to −+ intervals.

Now, we give an example of how to define the multidimensional schema, as defined in Definition 1, of a fact, starting from its conceptual graphical modeling. In this example, we start from Figure 4.

Example 1. The multidimensional schema of the fact Treatment depicted in Figure 4 can be defined as follows:

$\mathcal{L} =$ {(Time, Date: Date), (Week, WeekNumber: Int),
(Month, MonthNumber: Int, MonthName: String),(Quarter, Quarter: Int),
(Year, Year: Int), (Drug, DrugKey: Int, DrugName: String),
(ATC2, ATC2Key: String), (ATC1,ATC1: String), (Patient, PatientID: Int,
FirstName: String, LastName: String, Sex: Char), (All)}
$\mathcal{D} =$ {Patient, Drug, Time},
$\mathcal{F} =$ {Treatment}

As for the schemata of dimensions, Time has been already introduced in Definition 1, being a widely used dimension. Dimension Drug uses the hierarchy ATC, named after the usual classification Anatomical Therapeutic and Chemical (ATC). The schema for the Drug and Patient dimensions is:

$\mathcal{L}_{\mathsf{Drug}} =$ {(Drug, DrugKey: Int, DrugName: String), (ATC2, ATC2Key: String),
(ATC1, ATC1Key: String), (All)}
$\mathcal{H}_{\mathsf{Drug}} =$ $\{\langle \mathsf{ATC}, \{\mathsf{Drug}, \mathsf{ATC2}, \mathsf{ATC1}\}, R_{\mathsf{ATC}}\rangle\}$
$R_{\mathsf{ATC}} =$ $\{\langle \mathsf{Drug}, \mathsf{ATC2}, \text{1-m}\rangle, \langle \mathsf{ATC2}, \mathsf{ATC1}, \text{1-m}\rangle, \langle \mathsf{ATC1}, \mathsf{All}, \text{1-m}\rangle\}$
$\mathcal{L}_{\mathsf{Patient}} =$ {(Patient, PatientID: Int, FirstName: String, LastName: String, Sex: Char), (All)}
$\mathcal{H}_{\mathsf{Patient}} =$ $\{\langle \mathsf{Pat}, \{\mathsf{Patient}\}, R_{\mathsf{Pat}}\rangle\}$
$\mathcal{R}_{\mathsf{Pat}} =$ $\{\langle \mathsf{Patient}, \mathsf{All}, \text{1-m}\rangle\}$

The schema of the Treatment fact is described as:

$$\begin{aligned}\mathsf{Treatment} = \quad & \{\mathsf{TreatmentKey}, \{\langle \text{Drug, 1-m, Drug}\rangle, \langle \text{Time, 1-m, DrugStart}\rangle,\\ & \langle \text{Time, 1-m, DrugStart}\rangle, \langle \text{Time, 1-m, DrugEnd}\rangle,\\ & \langle \text{Time, 1-m, EffectStart}\rangle, \langle \text{Time, 1-m, EffectEnd}\rangle, \langle \text{Patient, 1-m, Patient}\rangle\},\\ & \{\langle \text{DrugPeriod, DrugStart, DrugEnd}, \{\mathsf{Cost}\}, ++\rangle,\\ & \langle \text{EffectDelay, DrugStart, EffectStart}, \{\mathsf{Cost}\}, +-\rangle,\\ & \langle \text{ResidualPeriod, EffectEnd, DrugEnd}, \{\}, -+\rangle,\\ & \langle \text{TreatmentPeriod, DrugStart, EffectEnd}, \{\}, ++\rangle,\\ & \langle \text{EffectPeriod, EffectStart, EffectEnd}, \{\mathsf{Cost}\}, ++\rangle\},\\ & \langle \text{Cost: Int, Additive}\rangle, \langle \text{DailyDosage: Int, Semi-Additive}\rangle\}\end{aligned}$$

Given the definition of a schema, we may define next multidimensional instances.

Definition 2. Multidimensional Instance. A multidimensional instance, also called *cube*, is composed by instances of dimensions and facts.

The instance of a dimension D_i is composed by:

- A set $\mathcal{B}_k$ of members for each level in $L_k \in \mathcal{L}_i$. The level *All* has a unique member *all*.
- A finite set of roll-up relations $R^{j,k}_{name}$, where *name* is the hierarchy name, containing pair of members (b_j, b_k), where b_j identifies a member in level $L_j \in \mathcal{L}_i$, and b_k identifies a member in level $L_k \in \mathcal{L}_i$. A pair $\langle b_j, b_k\rangle \in R^{j,k}_{name}$ if the member b_j rolls up to b_k.

A fact instance C is composed by cells $c \in C$ characterized by:

- an identifier k,
- a member b for each level participating in a role-playing dimension, and
- a value for each measure that quantifies a fact.

We also introduce an auxiliary function K that, given a cell c, $K(c)$ returns the identifier value of c.

We give next an example of fact instance.

Example 2. The instance of dimension Drug of the multidimensional schema depicted in Figure 4 can be defined as:

$\mathcal{B}_{\mathsf{Drug}} = \{(\mathsf{D1}, \mathsf{Tylenol}), (\mathsf{D2}, \mathsf{Aspirin})\}$

$\mathcal{B}_{\mathsf{ATC2}} = \{(\mathsf{N02}), (\mathsf{B01})\}$

$\mathcal{B}_{\mathsf{ATC1}} = \{(\mathsf{N}), (\mathsf{B})\}$

$R^{\mathsf{Drug,ATC2}}_{\mathsf{ATC}} = \{\langle \mathsf{D1}, \mathsf{N02}\rangle, \langle \mathsf{D2}, \mathsf{B01}\rangle\}$

$R^{\mathsf{ATC2,ATC1}}_{\mathsf{ATC}} = \{\langle \mathsf{N02}, \mathsf{N}\rangle, \langle \mathsf{B01}, \mathsf{B}\rangle\}$

The instances of fact Treatment are in the form: $\mathsf{Treatment} = \{(\mathsf{P1}, \mathsf{Tylenol}, 65, 40, \text{16-08-2018}, \text{07-09-2018}, \text{03-09-2018}, \text{12-09-2018}), \dots\}$. The tuple in the example instance corresponds to the first line of Table 1.

5. Temporal OLAP Operations

In this section, we present OLAP operators to deal with interval-based RPTDs: the *to atelic* and *to telic* operators, which change the representation of some measures; the *interval slice* operator, which filters interval slices on data; The *cumulative sum* operator, which computes cumulative telic values of the measure over intervals; and the *temporal roll-up* operator, which temporally summarizes data. As introduced in Section 4, measures can be additive, semi-additive, and non-additive. These different types of measure ensure the

summarizability property, which is needed when aggregating data [34]. Another issue to consider deals with interval-based data. Measures can quantify events that are characterized by a culmination, also called *telic* events, and events that represent states, also called *atelic* events.

In the next subsection we shall define the function for adjusting measures according to the considered intervals.

5.1. Measure Adjustment

In Definition 1, each time link is associated with a set of measures that must be considered as telic ones over that interval. When manipulating telic measures, we must consider the fact that the downward property does not hold [37]. As an example, assume that a value of 100 is associated to an interval that lasts for 10 days. If the downward property holds, the value of 100 may be associated for every subinterval within those 10 days. Otherwise, the interval represents a complete state, and the value of 100 is valid only over the 10 day period. The corresponding value for a subinterval can be approximated according to some functions: we propose here an adjustment function *Adj* that modifies a value of a measure according to a given interval.

Definition 3. Adjustment (Adj) Function

Let $\cap(i_1, i_2)$ be an auxiliary function that, given two intervals, returns their intersection. Given a measure M, its value v, an interval-based RPTD coordinate in, and an interval l, the *Adj* function is defined as follows:

$$Adj(M, v, in, l) = \begin{cases} \frac{v \cdot (end(\cap(l,in)) - start(\cap(l,in)) + 1)}{end(in) - start(in) + 1} & : M \text{ is telic over } in \\ v & : M \text{ is atelic over } in \end{cases}$$

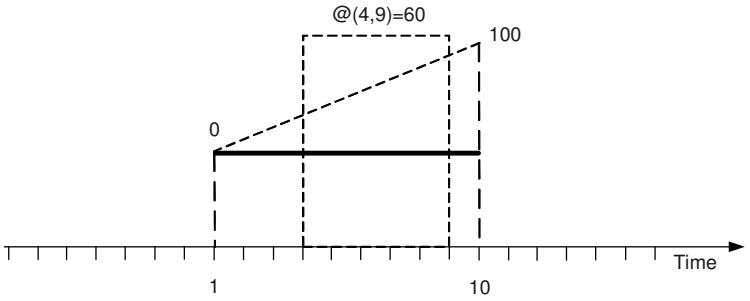

Figure 5. Example of measure adjustment for a telic measure. The value is adjusted by supposing that the measure increases linearly over time

Intuitively, the value of the measure is proportionally adjusted to reflect its value over the intersection between interval l and the RPTD instance in. As an example, let us assume that a given measure has a value of 100, and that the closed interval associated to it over the chosen temporal dimension is $[1, 10]$. If the temporal interval is $l = [4, 9]$, then the interval length is $9 - 4 + 1 = 6$ and the resulting measure value is adjusted to 60, as depicted in Figure 5. By using a simple linear monotonic function, we assume that the measure increases linearly over time: however, as already pointed out in the literature by [25], other functions can also be considered for computing a better estimate.

In general, we may introduce a more general version of function *Adj*, with a fifth argument, i.e. $Adj(M, v, in, l, f)$, where f is the function representing how values of telic measure M are "distributed" over interval in. This way, a new telic measure can be derived for the intersection of the given intervals. In this case, the function is redefined for telic measures as:

$$Adj(M, v, in, l, f) = \begin{cases} f(end(\cap(l, in))) - f(start(\cap(l, in))) & : M \text{ is telic over } in \\ v & : M \text{ is atelic over } in \end{cases}$$

Hereinafter, for sake of simplicity and without loss of generality, we will consider telic measures as the result of a linear increase over the given interval.

5.2. Converting measures from telic to atelic ones and vice versa

A common need when considering temporal data is to move from atelic data to their corresponding atelic representation and vice versa, according the analysts' requirements. For instance, users may want to consider the overall quantity of drugs administered to patients, considering both the given daily quantity and the administration period. On the other side, it could be useful to move, similarly to the approach proposed for the adjustment function in the previous section, from a telic measure to its corresponding atelic one. For instance, a user could require to have the daily cost of a drug, according to some associated period.

The following operators allow one to move from atelic measures to telic ones and vice versa.

Definition 4. τ_A Operator (To Atelic)
Given a multidimensional cube C, a telic measure M, one of its interval-based RPTD I, and a new measure name M', the operator τ_A returns a cube where for cells $c \in C$ values of new atelic measure M' are computed by considering the corresponding telic values of measure M.

The syntax of the τ_A operator is:

$$C' = \tau_A(C, M, I, M')$$

and it is defined as[2]:

$$C' = \{x \mid \exists y \in C(\forall i_{[1\ldots m]}(N_i(x) = N_i(y)) \wedge \forall i_{[1\ldots n]}(M_i(x) = M_i(y)) \wedge M'(x) = \tfrac{M(y)}{end(I(y)) - start(I(y)) + 1})\}$$

Definition 5. τ_T Operator (To Telic)
Given a multidimensional cube C, an atelic measure M, one of its interval-based RPTD I, and a new measure name M', the operator τ_T returns a cube where for cells $c \in C$ values of new atelic measure M' are computed by considering the corresponding atelic values of measure M.

The syntax of the τ_T operator is:

$$C' = \tau_T(C, M, I, M')$$

and it is defined as:

$$\begin{aligned} C' = \{x \mid \exists y \in C(&\forall i_{[1\ldots m]}(N_i(x) = N_i(y)) \wedge \\ &\forall i_{[1\ldots n]}(M_i(x) = M_i(y)) \wedge M'(x) = M(y) \cdot (end(I(y)) - start(I(y)) + 1))\} \end{aligned}$$

5.3. Interval Slice

Filtering is one of the most common operations on data. The standard OLAP slice operator returns data subsets according to coordinates on instant-based dimensions. For instance, users may want to focus on some of the administered drugs, only. The interval slice operator, instead, allows one to perform filtering on interval-based temporal dimensions.

The interval slice operator allows one also to perform queries like "What is the value of the measure during a given interval?" Such operator is named *At*, and it is depicted by the symbol @. Given a time interval l and a fact instance, the operator returns the subset of cells that intersect l.

Definition 6. @ Operator (Interval Slice)
Given a multidimensional cube C, one of its interval-based RPTD I between roles N_j and N_k of dimension Time, and an interval l, the operator @ returns (with some possible adjustments) all the cells $c \in C$ for which the temporal relationship *intersect* holds between the instances of I and l. The intersection relationship

[2] In the following we will use a sort of calculus for specifying the meaning of operators. Variables stand for cube cells, while functions express dimension-related role members and measures for the given cells. Some basic arithmetic operations on measures are used.

corresponds to the disjunction of *equal*, *finishes*, *starts*, *during*, *overlaps*, and *meets* ones (and their respective inverse relationships, i.e., *started-by*, *finished-by* etc.) from the Allen's interval algebra [1].

The syntax of the @ operator is:

$$C' = @(C, I, l)$$

and it is defined as:

$$C' = \{x \mid \exists y \in C, \text{intersect}(I(y), l) \wedge \forall i_{[1\ldots m], i \neq j,k}(N_i(x) = N_i(y)) \wedge N_j(x) = start(\cap(I, l)) \wedge N_k(x) = end(\cap(I, l)) \wedge \forall i_{[1\ldots n]}(M_i(x) = \text{Adj}(M_i, M_i(y), I(y), l)))\}$$

Intuitively, only cells intersected by the given temporal interval over the chosen RPTD are kept. The first line builds up all the role playing dimensions, but the last one. The *intersect* relationship between the instance of I and l is checked, and it is then assigned to the selected RPTD. Finally, measures with adjusted values are built up.

A particular case arises when, instead of a time interval, a time point is given as input. In this case, the interval slice operator allows one to perform queries like "Compute the value of a measure at a particular point in time". Given a time point t and a fact instance, the operator returns the subset of cells whose interval contains t. Since the temporal instant t can be seen as the interval $[t, t]$, this case is equivalent to the interval-based case.

5.4. Cumulative Sum

By applying the @ operator, users are interested in the instantaneous value of a measure. As depicted by the previous example, in case of telic measures the @ operator returns the contribution of the measure for that instant. The cumulative sum operator is needed to compute measure *cumulative* values over intervals. This particular way of filtering is used to perform queries such as: "Compute the cost of Aspirin treatments up to this day". To perform such a query, we propose the Σ operator, defined next.

Definition 7. Σ Operator (Cumulative Sum)
Given a multidimensional cube C, one of its interval-based RPTD I between roles N_j and N_k of dimension Time, and an instant t, the Σ returns all the cells $c \in C$ for which the temporal relationship *contains* holds between the instances of I and t. The syntax of Σ is:

$$C' = \Sigma(C, I, t)$$

and it is defined as:

$$C' = \{x \mid \exists y \in C, \text{contains}(I(y), l) \wedge \forall i_{[1\ldots m], i \neq j,k}(N_i(x) = N_i(y)) \wedge N_j(x) = start(I(y)) \wedge N_k(x) = t \wedge \forall i_{[1\ldots n]}(M_i(x) = \text{Adj}(M_i, M_i(y), I(y), [start(I(y)), t]))\}$$

Intuitively, only those cells containing the given time interval over the chosen RPTD are kept. The first line builds up all the role playing dimensions but the last one. The *contains* relationship between the instance of I and t is checked. Finally, measures and dimensions with adjusted values are returned.

5.5. Temporal Roll-up

The roll-up operation allows one to summarize data from a finer to a coarser granularity level in the existing dimensions. The first phase in the data summarization process aims at creating the aggregation groups, that is, grouping data according to common coordinate values. Most models, like those we presented in Section 4, include the top level named *All* which allows one to get rid of dimensions in the aggregation phase. As an example, in the query "Compute the total cost of treatments per drug", only the Drug dimension is mentioned, meaning that the other dimensions are implicitly used at the *All* level. This will lead to as many aggregation groups as the number of Drug members.

While finding the aggregation groups in point-based dimensions is quite trivial, working with interval-based dimensions is more difficult. We consider here two techniques to achieve such temporal aggregations:

- Instant temporal aggregation (ITA) [6, 22, 28, 31, 40] computes the aggregate value at a given temporal instant t, considering all the tuples whose timestamp contains t. In case of telic measures, suitable adjustments are needed to derive atelic measures, as shown in the following examples. The resulting tuples at consecutive temporal instants with identical aggregate values are then coalesced (i.e., fused together) into one unique tuple, over the maximal temporal intervals during which the aggregate results are constant. ITA will be used when only nontemporal dimensions are used to create the aggregation groups.
- Span temporal aggregation (STA) [6, 22] allows an application to specify the temporal intervals for which to report resulting tuples, e.g., for each year from 2010 to 2014. For each of these intervals, a result tuple is produced by aggregating all the argument tuples that overlap such an interval.

As we shall describe later on in the current section, the proper aggregation will be selected according to the dimensions used to create the aggregation groups. In both cases (ITA and STA), measure values are adjusted to the new interval lengths. The temporal roll-up operator ρ is defined as follows.

Definition 8. ρ Operator (Temporal Roll-up)
Given a multidimensional cube C, one of its interval-based RPTD I, a set of levels L (one for each role playing dimension N_i of C), and an optional temporal granule g, the ρ operator is defined as:

$$C' = \rho(C, I, \{l_1, \ldots, l_m\}, [g])$$

The third parameter $\{l_1, \ldots, l_m\}$ is a set of levels containing one level for every dimension the cube is made of. This set is used to generate the aggregation groups. Intuitively, the *All* level states that no particular dimension is used in the aggregation groups. The optional parameter g is used to create the temporal aggregation groups, and it consists of a granularity level taken from the RPTD upon which the components of I are defined. As an example, EffectDelay is an interval-based RPTD composed by the instant-based RPTD DrugStart and EffectStart. Both instant-based RPTDs are defined over the Time dimension. The granule g is chosen among the levels of the Time dimension, namely Date, Week, Month, Quarter, and Year. ITA is performed when the optional parameter g is not defined: the algorithm looks for the new intervals in which the measure values are constant. STA, on the other hand, is performed when the optional parameter g is defined.

5.6. Temporal operators at work

Let us now consider some analysis requirements related to the motivating scenario introduced in Section 3. In the following examples we will discuss how to use and combine the previously introduced operators to manage some suitable OLAP queries, possibly involving multiple RPTDs.

Example 3. Consider the example of Figure 4. DrugStart and EffectStart have been combined to create the interval-based RPTD EffectDelay. Let us consider the query: "Compute, for each drug, the total cost of treatments during the effect delay period between Sept. 1 and Sept. 15". To evaluate the query, we need to suitably apply different temporal operators, before applying the last aggregation. More precisely, we have first to derive the DailyCost of the drug considering the telic measure Cost and the related RPTD DrugPeriod, by applying the *To Atelic* operator

$$\mathsf{Treat}' = \tau_A(\mathsf{Treatment}, \mathsf{Cost}, \mathsf{DrugPeriod}, \mathsf{DailyCost})$$

Then, we need to derive a further telic measure from the atelic DailyCost by considering RPTD EffectDelay through the *To Telic* operator

$$\mathsf{Treat}'' = \tau_T(\mathsf{Treat'}, \mathsf{DailyCost}, \mathsf{EffectDelay}, \mathsf{EffectCost})$$

Such cube Treat$''$ will be used throughout the following examples, to show the kind of temporal queries we are able to define, based on the suitable composition of different temporal operators.

Finally, we apply the @ operator as follows:

$$\mathsf{TreatItvl} = @(\mathsf{Treat}'', \mathsf{EffectDelay}, [\mathsf{Sept\ 1}, \mathsf{Sept\ 15}])$$

Figure 6 graphically depicts the results. Dashed lines represent cell intervals, or part of them, that have been discarded. Solid lines represent cell intervals, or part of them, intersected by [Sept. 1, Sept. 15]. Only the telic measure Cost has been adjusted: e.g., for the tuple (1, P1, Tylenol, 65, 40) whose administration started on Aug. 16, ended on Sept. 07, and whose effects started on Sept. 3, the original Cost of 65 was adjusted to a DailyCost as $\frac{65}{(Sept.07-Aug.16+1)} = 2.826$; the cost during the EffectDelay over the interval [Sept. 1, Sept. 15] is then computed as $2.826 \times (Sept.3 - Sept.1 + 1) = 5.652$.

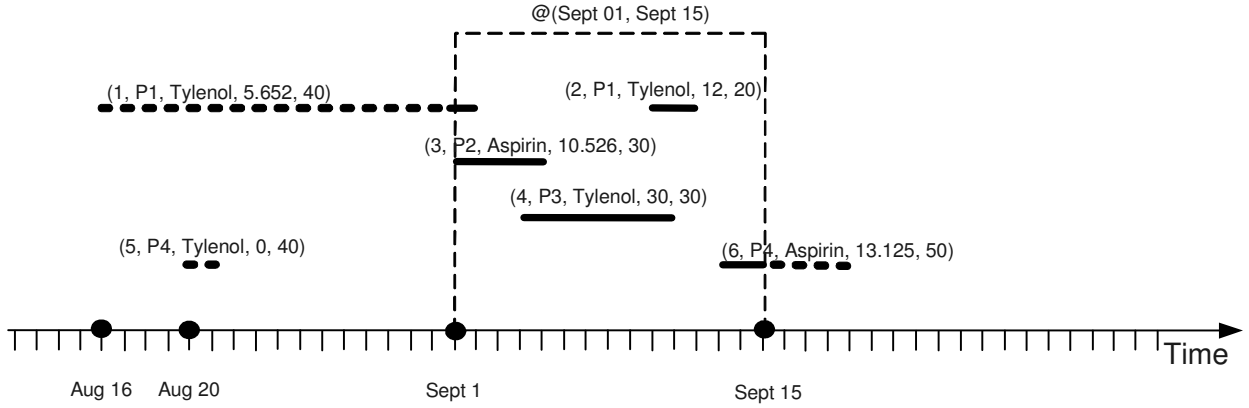

Figure 6. Result of the application of the @ operator @(Treat″, EffectDelay, [Sept 1, Sept 15]) starting from the Treatment cube, to evaluate a measure over a given temporal interval

At this point, the results evaluated over all the tuples can then be grouped by Drug dimension members (Tylenol and Aspirin) leading to the values (Tylenol, 47.652) and (Aspirin, 23.651) (not shown in the figure).

A similar approach can be taken for instant-based queries.

Example 4. Consider the example of Figure 4. DrugStart and EffectStart have been combined to create the interval-based RPTD EffectDelay. Let us consider the query: "Compute, for each drug, the total cost of treatments during the effect delay on Sept. 5". To evaluate this query, the @ operator must be applied as follows:

$$\mathsf{TreatInst} = @(\mathsf{Treat}'', \mathsf{EffectDelay}, [\mathsf{Sept\ 5}, \mathsf{Sept\ 5}])$$

Figure 7 graphically depicts the result. Dashed lines represent cell intervals that have been discarded. Filled diamonds represent time points within the interval. Only the Cost measure has been adjusted, since it is a telic one. We have two tuples whose EffectDelay includes Sept. 05: (3, P2, Aspirin) and (4, P3, Tylenol) whose DailyCost are 2.105 and 3.75, respectively.

The result over all the drugs is thus computed as $2.105 + 3.75 = 5.855$

Example 5. Consider the example of Figure 4. DrugStart and EffectStart have been combined to create the interval-based RPTD EffectDelay. Let us consider the query: "For each administered drug, compute the total costs up to Sept. 5 between the beginning of the drug administration and Sept. 5, where the effect still is unobserved as of Sept. 5". To perform the query we apply the Σ operator still still using cube TreatFinal as:

$$\mathsf{TreatSigma} = \Sigma(\mathsf{Treat}'', \mathsf{EffectDelay}, \mathsf{Sept\ 5})$$

Figure 8 graphically depicts the result. Dashed lines represent cell intervals, or part of them, that have been discarded: e.g. the tuple (1, P1) is discarded because, on September 5, the effect of the administration

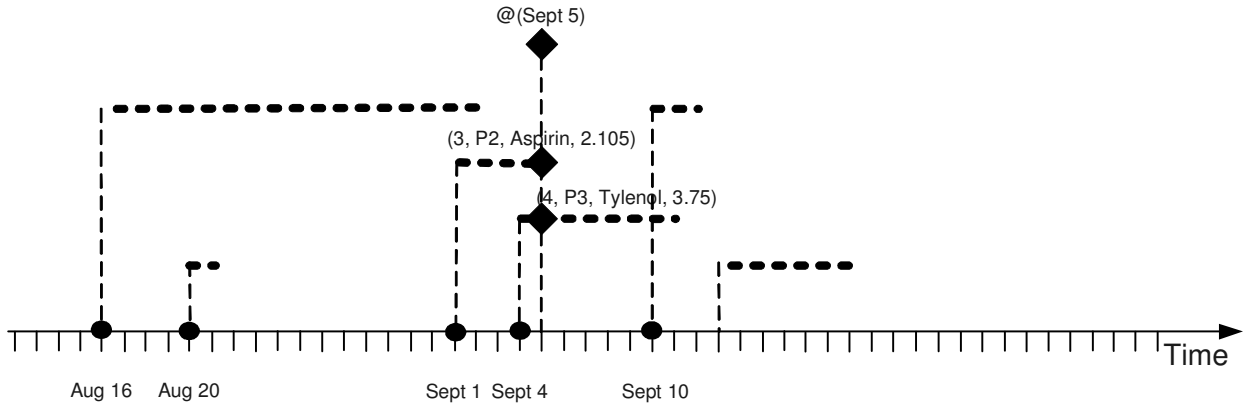

Figure 7. Result of @(Treat″, EffectDelay, [Sept 5, Sept 5]). Dashed intervals represent cells that have been discarded. The remaining intervals represent cells intersected by the selected time point. Only the telic measure EffectCost has been considered.

is visible. Solid lines represent cell intervals, or part of them, containing the temporal instant Sept. 5: e.g. only a part of the tuple (4, P3) is considered, because the effect of the administration will start to be visible on Sept. 12.

Only the telic measure Cost has been adjusted: e.g., for the tuple (4, P3) whose administration started on Sept. 04, ended on Sept. 19, and whose effects started on Sept. 12, the original Cost of 60 was converted to a DailyCost as $\frac{60}{(Sept.19-Sept.04+1)} = 3.75$; the cost during the EffectDelay up to Sept. 05 is then computed as $3.75 \times (Sept.5 - Sept.4 + 1) = 7.5$. For the tuple (3, P2) whose administration started on Sept. 01, ended on Sept. 08, and whose effects started on Sept. 06, the original Cost of 80 was converted to a DailyCost as $\frac{80}{(Sept.08-Sept.01+1)} = 10$; the cost during the EffectDelay up to Sept. 05 is then computed as $2.105 \times (Sept.5 - Sept.1 + 1) = 10.526$.

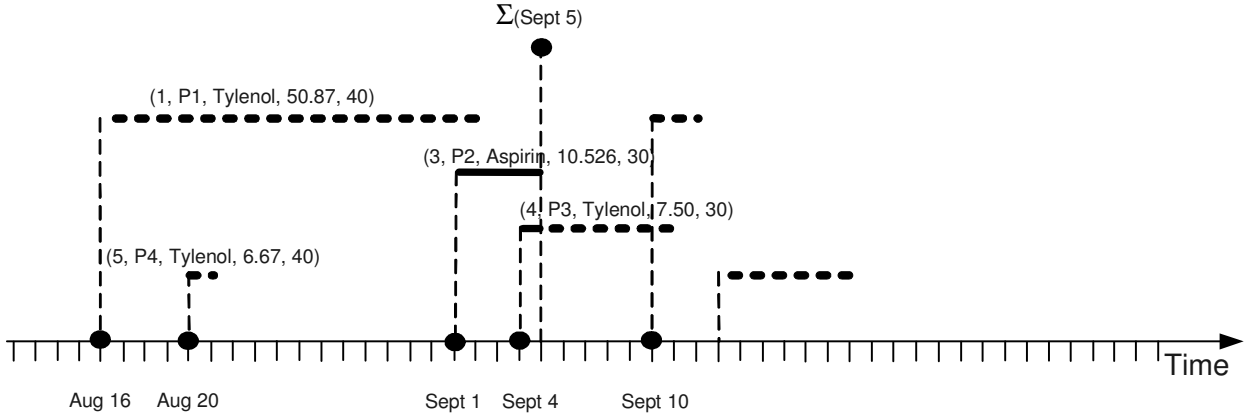

Figure 8. Result of the application of the Σ operator Σ(Treat″, EffectDelay, Sept 5), based on the Treatment cube. Dashed lines represent cell intervals, or part of them, that have been discarded. Solid lines represent cell intervals, or part of them, containing Sept. 5. The telic measure EffectCost has been considered according to given portion of intervals containing Sept 5.

Example 6. Let us consider Figure 4 and the query "Compute the total cost of treatment per drug during the effect delay period". To perform this query, we first need the temporal roll-up ρ operator as:

$$\mathsf{TreatRollUp} = \rho(\mathsf{Treat}'', \mathsf{EffectDelay}, \{\mathsf{All}, \ldots, \mathsf{Drug}, \ldots \mathsf{All}\})$$

The EffectDelay RPTD is used by the ITA algorithm to create the temporal groups. The only considered instant-based dimension is the Drug one. Figure 9 shows the result of the application of the ρ operator: two

aggregation groups have been created, one for every Drug (i.e., Tylenol and Aspirin). Within each group, the ITA algorithm identifies the intervals for which the measure value is constant. Since Cost is a telic measure, its value has been adjusted to the length of the new intervals.

For instance, the Tylenol case of patient (1, P1) has a Cost value (i.e. 65) which has been split according to the length of intervals: the DrugPeriod original interval is $(Sept.07 - Aug.16 + 1) = 23$ days long and the DailyCost is thus $\frac{65}{23} = 2.826$. Tuple (1, P1) has been divided into four parts, since it contains the treatment of patient (5,P4) and overlaps with the treatment of patient (3, P2):

i. the first subinterval is computed as $(Aug.20 - Aug.16) = 4$ days long (we consider the closed interval [Aug. 16 ÷ Aug. 19]): the measure has been adjusted according to the formula $65 \times \frac{4}{23} = 11.304$;
ii. the second subinterval, i.e. the one that contains (5, P4), is computed as $(Aug.22 - Aug.20) = 2$ days long (we consider the closed interval [Aug. 20 ÷ Aug. 21]): for patient P1 the measure has been adjusted according to the formula $65 \times \frac{2}{23} = 5.652$. During this interval, (5, P4) produces a cost for Tylenol of $30 \times \frac{2}{9} = 6.67$;
iii. the third subinterval, i.e. the one that ends as (3, P2) starts, is computed as $(Sept.1 - Aug.22) = 10$ days long (we consider the closed interval [Aug. 22 ÷ Aug. 31]): for patient P1 the measure has been adjusted according to the formula $65 \times \frac{10}{23} = 28.267$;
iv. the fourth subinterval, i.e. the one that ends as the effect starts and partially overlaps (3, P2), is computed as $(Sept.3 - Sept.1) = 2$ days long (we consider the closed interval [Sept. 01 ÷ Sept. 02]): for patient P1 the measure has been adjusted according to the formula $65 \times \frac{2}{23} = 5.652$.

Figure 9 depicts the complete costs of Tylenol: for [Sept. 04 ÷ Sept. 05] tuple (4, P3) produces a cost of $60 \times \frac{2}{16} = 7.50$; for [Sept. 06 ÷ Sept. 09] tuple (4, P3) produces a cost of $60 \times \frac{4}{16} = 15.00$; for [Sept. 10 ÷ Sept. 11] tuple (4, P3) produces a cost of $60 \times \frac{2}{16} = 7.50$ and tuple (2, P1) produces a cost of $20 \times \frac{2}{5} = 8.00$; for [Sept. 12 ÷ Sept. 12] tuple (2, P1) produces a cost of $20 \times \frac{1}{5} = 4.00$.

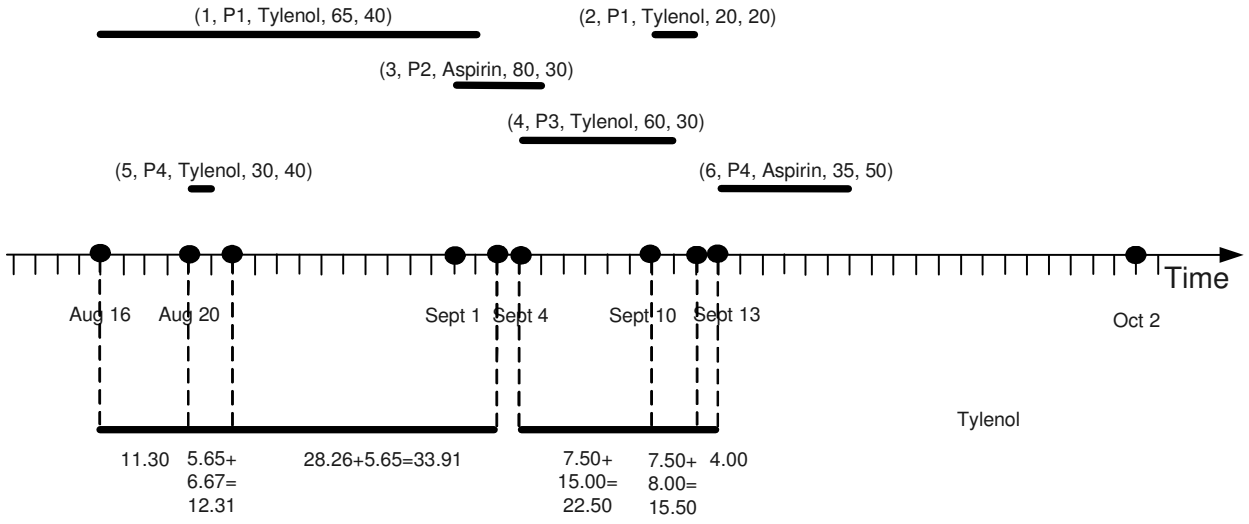

Figure 9. Graphical representation of a part of the result for the query "*Compute the total cost of treatment per drug during the effect delay period*". It has been computed by applying the temporal roll-up ρ algorithm in case of instant temporal aggregation. Two aggregation groups have been created, one for each drug (i.e., Tylenol, Aspirine). Within each group (Tylenol in the figure), the ITA algorithm found the intervals for which the measure value is constant. The measure values have been adjusted according to interval lengths. Bullets on the X-axis depict remarkable timestamps of EffectDelay period as stored in Treat" cube

.

Example 7. Let us consider Figure 4 and the query "Compute the total cost of treatments for each drug administered during the weeks which entail the effect delay period". We may apply the temporal roll-up ρ operator as:

$$\mathsf{TreatRollWeek} = \rho(\mathsf{Treat''}, \mathsf{EffectDelay}, \{\mathsf{All}, \ldots, \mathsf{Drug}, \ldots \mathsf{All}\}, \mathsf{Week})$$

In this example, Week is used by STA to create the temporal groups. Then, STA performs the aggregation as depicted in Figure 10. The time axis is equally divided according to the Week granularity, every week starting on Sunday. Each cell, represented by an interval, is assigned to the intersected granules. Consequently, measure values are adjusted according to the new interval lengths.

For instance, the treatment (2, P1) belongs to two consecutive granules (weeks), the first granule being shared with the treatment (4, P3), the second granule being shared with the treatment (6, P4). Telic measures are proportionally adjusted on how long the interval spans in every granule. The treatment (2, P1) spans in the granule [Sept. 04 ÷ Sept. 10] (we use closed intervals) for one day. The Cost measure is 20, where the DrugPeriod is 5 days long ([Sept. 10 ÷ Sept. 14]) and the EffectStart is Sept. 13. Adjusting the cost according to the formula of Definition 3, the value is computed as $20 \times \frac{1}{5} = 4.00$ over the granule [Sept. 04 ÷ Sept. 10].

Likewise, the treatment (4, P3) spans in the granule [Sept. 04 ÷ Sept. 10] (we use closed intervals) for 7 days. The Cost measure is 60, where the DrugPeriod is 16 days long ([Sept. 04 ÷ Sept. 19]) and the EffectStart is Sept. 12. Adjusting the cost according to the formula of Definition 3, the value is computed as $60 \times \frac{7}{16} = 26.25$ over the granule [Sept. 04 ÷ Sept. 10]. Thus, the grand total value for the [Sept. 04 ÷ Sept. 11] granule for Tylenol is $4.00 + 26.25 = 30.25$.

Figure 10 depicts the complete costs of Tylenol and Aspirin over the week granules as follows:

i. for [Aug. 14 ÷ Aug. 20] tuple (1, P1, Tylenol) produces a cost of $65 \times \frac{5}{23} = 14.13$ and (5, P4, Tylenol) produce a cost of $30 \times \frac{1}{9} = 3.33$;
ii. for [Aug. 21 ÷ Aug. 27] tuple (1, P1, Tylenol) produces a cost of $65 \times \frac{7}{23} = 19.78$ and (5, P4, Tylenol) produces a cost of $30 \times \frac{1}{9} = 3.33$;
iii. for [Aug. 28 ÷ Sept. 03] tuple (1, P1, Tylenol) produces a cost of $65 \times \frac{6}{23} = 16.95$ and tuple (3, P2, Aspirin) produces a cost of $80 \times \frac{3}{38} = 6.31$;
iv. for [Sept. 04 ÷ Sept. 10] tuple (3, P2, Aspirin) produces a cost of $80 \times \frac{2}{38} = 4.21$ and tuple (4, P3, Tylenol) produces a cost of $60 \times \frac{7}{16} = 26.25$;
v. for [Sept. 11 ÷ Sept. 17] tuple (2, P1, Tylenol) produces a cost of $20 \times \frac{2}{5} = 8.00$ and tuple (4, P3, Tylenol) produces a cost of $60 \times \frac{1}{16} = 3.75$ and tuple (6, P4, Aspirin) produces a cost of $35 \times \frac{5}{8} = 21.75$;
vi. for [Sept. 18 ÷ Sept. 24] tuple (6, P4, Aspirin) produces a cost of $35 \times \frac{2}{8} = 8.75$.

6. Expressing Temporal OLAP Operations in SQL

This section describes how all the OLAP operators have been implemented by SQL scripts. As language of reference, we consider here SQL-92. More particularly we introduce and discuss the implementation of operators by considering the examples provided in the previous section. With respect to the operators introduced, we now consider, for sake of simplicity, only those table attributes, relevant to the discussed queries (all the other attributes of fact and hierarchy tables will be omitted). The basic approach we follow, consists of defining suitable views corresponding to the different operators introduced in the previous sections. Thus, we will introduce examples related to views corresponding to operators τ_A and τ_T, allowing us to move from telic to atelic data and viceversa, to operator @, to operator Σ, and to operator ρ with different kinds of temporal aggregation. The different adjustment functions are embedded whithin the proposed views.

6.1. Conversion operators and Interval Slice

According to Example 3, we want to "Compute, for each drug, the total cost of treatments during the effect delay period between Sept. 1 and Sept. 15" (closed interval), as in Figure 6. To do it, we will specify operators τ_A, τ_T, and @ through suitable views, respectively. More precisely, views Treat1 and Treat2,

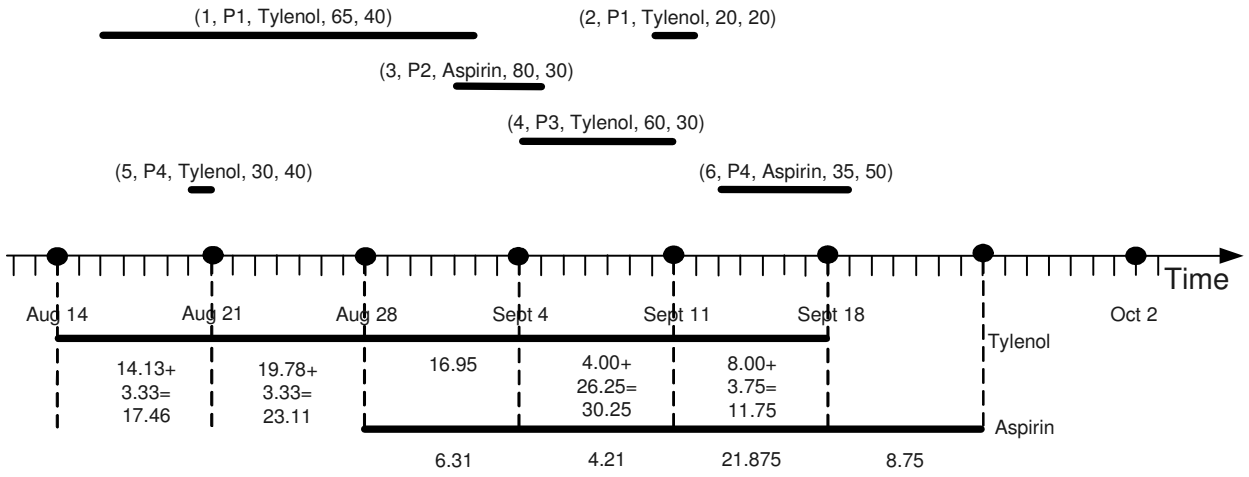

Figure 10. Graphical representation of the result of the query "*Compute the total cost during the effect delay per drug*". This result has been computed by applying the ρ algorithm in case of span temporal aggregation. As one may notice, two aggregation groups have been created, one for each drug (i.e., Tylenol and Aspirine). The time line has been divided in equally sized granules by the STA algorithm according to the Week granularity: bullets on the X-axis depict the beginning timestamp of every granule. Within each granule, the measure EffectCost has been adjusted according to how long the interval spans in every granule

correspond to the application of operators τ_A and τ_T to the data discussed in Example 4, respectively. Listing 1 provides the corresponding SQL code.

Then, view TreatItvl implements the Interval Slice operator, i.e.@, as detailed in Listing 2. In this view, both the condition of intersection between RPTD values and the given interval, in the WHERE condition, and the adjustment function, in the CASE part of SELECT clause), are expressed.

```
1
2  create view Treat1(Drug, Patient, DailyCost, EffectDelayS, EffectDelayE) as
3  select Drug, Patient, Cost / (DrugEnd - DrugStart + 1), DrugStart, EffectStart - 1
4  from Treatment;
5
6  create view Treat2(Drug, Patient, EffectCost, EffectDelayS, EffectDelayE) as
7  select Drug, Patient, DailyCost*(EffectDelayE - EffectDelayS + 1), EffectDelayS,
       EffectDelayE
8  from Treat1;
```

Listing 1. Conversion operators

```
1  with variables(var1, var2) as values('2016-09-01','2016-09-15')
2  create view TreatItvl(Drug, Patient, EffectCost, EffectDelayS, EffectDelayE) as
3  select Drug, Patient,
4         EffectCost / (EffectDelayE - EffectDelayS +1) *
5                ((case
6                      when (EffectDelayE < var2) then EffectDelayE
7                    else var2
8                  end) -
9                (case
10                     when (EffectDelayS >= var1) then EffectDelayS
11                   else var1
12                 end) + 1)), EffectDelayS, EffectDelayE
13 from Treat2, variables
14 where EffectDelayE >= var1 and EffectDelayS <= var2;
15
16 select Drug, sum(EffectCost)
17 from TreatItvl
```

```
group by Drug;
```

Listing 2. Interval Slice and final query

Finally, the last query in Listing 2 sums up the tuples of view TreatItvl, grouping them by Drug.

The interval slice operator @ can also evaluate the measure over a given granule (see Example 4 and Figure 7). In this case, lower bound (start timestamp) and upper bound (end timestamp) specify the same granule value, i.e. lower bound = upper bound. Listing 2 still applies, by only changing the values for var1 and var2, corresponding to the interval bounds.

6.2. Cumulative Sum

The cumulative sum operator Σ evaluates the cumulative value of a measure over a given interval. Even in this case, we define a corresponding suitable view, using the view already built and corresponding to the conversion operator τ_T. Such view contains both the computation of the cumulative sum and the condition, verifying the containment relation between the given timestamp and the considered RPTD value. Focusing on Example 5, let us consider the query "For each administered drug, compute the total costs up to September 5 between the beginning of the drug administration and September 5, where the effect still is unobserved" (Sept 5 included), as in Figure 8. According to the adopted approach, view (TreatSigma) stores the cost of the treatment of that drug during the part of the effect delay, which spans up to the required timestamp. Finally, the last query sums up the tuples of view TreatSigma, grouping them by Drug, as in Listing 3.

```
with variables(var1) as values('2016-09-05')
create view TreatSigma(Drug, Patient, CostCumulative, EffectDelayS, EffectDelayE) as
select Drug, Patient,
       EffectCost *((case
                  when (EffectDelayE <= var1) then EffectDelayE
                else var1
                end) - EffectDelayS + 1), EffectDelayS, EffectDelayE
from Treat2, variables
where EffectDelayS <= var1 and EffectDelayE > var1;

select Drug, sum(CostCumulative)
from DrugCostCumulative
group by Drug;
```

Listing 3. Cumulative Sum

6.3. Roll Up

To specify Roll Up operator ρ in SQL, we need some intermediate views to build up the right intervals to consider for the following temporal aggregation. Such intervals are built according to the lower and upper bounds of RPTD intervals associated to the measures we are interested in.

To analyze in detail the different intermediate views we need to define, let us consider Example 6, where we want to "Compute the total cost of treatment per drug during the effect delay period", as in Figure 9. The first view (RollUpInterval) has to contain the suitable closed intervals, over which we have to compute then the temporal aggregation. Such intervals are depicted for the given example in Figure 9. Every closed interval stored in view RollUpInterval is defined by a starting timestamp (lower bound) and an ending timestamp (upper bound). Within view RollUpInterval, every starting timestamp has a corresponding ending timestamp, which is the day before the beginning timestamp of the adjacent, right-hand side interval.

The time points used to compute such view in our example are represented as bullets over the time-axis of Figure 9. To build view RollUpInterval we need to derive before an intermediate view RollUpTimestamp. It stores any starting timestamp of an RPTD interval and any timestamp immediately following any ending timestamp of the given RPTD interval. In our example, such view stores any EffectDelayS or the successive time point of an EffectDelayE timestamp for the considered tuples (for drug Tylenol in the example). In our

case, the suitable timestamps we have to identify are Aug. 16 (EffectDelayS for P1), Aug. 20 (EffectDelayS for P5), Aug. 22 (EffectdDelayE+1 for P5), Sept 03 (EffectdDelayE+1 for P1), and so on.

The intermediate view (RollUpTimestamp) is based on view Treat2 (corresponding to conversion operator τ_T), as in the previous examples, and the timestamps for (StartPoint) are computed by considering upper and lower bounds of EffectDelay.

```
create view RollUpTimestamp(StartPoint) as
select T1.EffectdDelayS as StartPoint
from Treat2 as T1
union
select T2.EffectDelayE + 1  as StartPoint
from Treat2 as T2;
```

Listing 4. View RollUpTimestamp

Thus, the suitable intervals we have to identify are [Aug. 16 ÷ Aug 19] (from EffectdDelayS of P1 to EffectdDelayS of P5 - 1), [Aug. 20 ÷ Aug. 21] (from EffectdDelayS of P5 to EffectdDelayS of P5 - 1), [Aug. 22 ÷ Sept. 02] (from EffectdDelayE of P5 to EffectdDelayE of P1), and so on. Over these intervals we shall then compute the cost per drug during the effect delay.

We build up the intermediate view (RollUpInterval), storing the beginning timestamp and the ending timestamp of the intervals. In such case a nested query is needed, to find the corresponding ending timestamp of any given starting timestamp.

```
create view RollUpInterval(BeginTimestamp, EndTimestamp) as
select RUT1.StartPoint as BeginTimestamp,
       RUT2.StartPoint - 1 as EndTimestamp
from RollUpTimestamp as RUT1, RollUpTimestamp as RUT2
where RUT1.StartPoint < RUT2.StartPoint
      and not exists (select RUT3.Drug
          from RollUpTimestamp as RUT3
          where RUT1.StartPoint < RUT3.StartPoint and
                RUT3.StartPoint < RUT2.StartPoint);
```

Listing 5. View RollUpInterval

Moving from view RollUpTimestamp, view RollUpInterval picks the StartPoints from two subsequent tuples in RollUpTimestamp (namely, RUT1 and RUT2 in the `from` clause of Listing 5), on condition that no tuple exists in the RollUpTimestamp view in between RUT1 and RUT2 (`not exists` command of Listing 5).

We build up one more intermediate view (RollUpIntervalCost), which computes, for every drug and for every interval in RollUpInterval, the cost of the administration during the part of the EffectDelay contained in that interval. It mainly consists in the application of the adjustment function previously introduced.

```
create view RollUpIntervalCost(BeginTimestamp, EndTimestamp, Drug, Patient,
       CostDuringInterval) as
select RUI.BeginTimestamp, RUI.EndTimestamp, T.Drug, T.Patient,
       (T.EffectCost/(T.EffectDelayE - T.EffectDelayS + 1))*
            (RUI.EndTimestamp - RUI.BeginTimestamp + 1)
from RollUpInterval as RUI, Treat2 as T
where (T.EffectDelayS <= RUT1.StartPoint and RUT1.StartPoint < RUT2.StartPoint
       and RUT2.StartPoint <= T.EffectDelayE);
```

Listing 6. RollUpIntervalCost

The view RollUpIntervalCost computes a Cartesian product between RollUpInterval and Treat2, and requires the EffectDelay interval [T.EffectDelayS ÷ T.EffectDelayE] to contain the interval [RUT1 ÷ RUT2] (`where` clause of Listing 6). The CostDuringInterval is then computed for every drug and every selected interval.

Finally, the query of Listing 7 selects the tuples from the view RollUpIntervalCost of Listing 6, groups them by interval (RUIC.BeginTimestamp, EndTimestamp), and sums up the CostDuringInterval. To increase readability, the query also sorts tuples by beginning timestamp.

```
create view TreatRollUp as
select RUIC.Drug, RUIC.BeginTimestamp, RUIC.EndTimestamp,
       sum(CostDuringInterval) as CostDuringSubPeriod
from RollUpIntervalCost as RUIC
group by RUIC.Drug, RUIC.BeginTimestamp, RUIC.EndTImestamp
order by RUIA.BeginTimeStamp, RUIA.Drug;
```

Listing 7. Query RollUp

After this example of ITA, let us consider, as a final case, an example of STA. According to Example 7, we want to "Compute the total cost of treatments for each drug administered during the weeks which entail the effect delay period", as in Figure 10. Analogously to the previous case, we need to derive some intermediate views, collecting in this case the bounds of all the required weeks. In order to do it, view (WeekRollUpInterval) stores the suitable closed intervals, as those depicted in Figure 10 for the considered example. Every closed interval stored in the view WeekRollUpInterval defines one week, starting on Sunday. For the given example, every Sunday is depicted by a bullet over the X-axis. The corresponding ending timestamp is the day before the adjacent, right-hand side bullet.

View WeekRollUpInterval is specified by leveraging the intermediate view WeekRollUpTimestamp. In our case, it stores the bullets over the X-axis of Figure 9 and contains any DrugStart or EffectStart timestamp. Thus, view WeekRollUpInterval, defined as in Listing 8, includes the first day of the weeks (Sunday) during which we have either a DrugStart or an EffectStart, or both of them. The SQL `weekday` function returns the day of the week specified as a number (Sunday is 1, Saturday is 7).

```
create view WeekRollUpTimestamp(StartPoint) as
select T1.DrugStart - weekday(T1.DrugStart) + 1 as StartPoint
from Treat2 as T1
union
select T2.EffectStart - weekday(T2.EffectStart) + 1 as StartPoint
from Treat2 as T2;
```

Listing 8. View WeekRollUpTimestampl

As in the previous case, we then define view WeekRollUpInterval of Listing 9, storing the granules (weeks) over which we shall then aggregate data. Every week starts with the Sunday defined in the view WeekRollUp-Timestamp of Listing 8.

```
create view WeekRollUpInterval(BeginTimestamp, EndTimestamp) as
select WRUT.StartPoint as BeginTimestamp, WRUT.StartPoint + 7 as EndTimestamp
from WeekRollUpTimestamp as WRUT
```

Listing 9. View WeekRollUpInterval

We build up one more intermediate view (WeekRollUpIntervalCost) of Listing 10, which computes, for every drug and for every week in the view WeekRollUpInterval of Listing 9, the cost of the administration of that drug during the part of the EffectDelay contained in that week.

```
create view WeekRollUpIntervalCost(BeginTimestamp, Drug, Patient, CostDuringInterval) as
select WRUI.BeginTimestamp as BeginTimestamp, T.Drug as Drug,
       T.Patient as Patient,
       (T.Cost / (T.DrugEnd - T.DrugStart + 1)
       *((case
              when (T.EffectStart <= WRUI.BeginTimestamp + 7) then
                   T.EffectStart
          else WRUI.BeginTimestamp + 7
          end)
          -
         (case
              when (T.DrugStart > WRUI.BeginTimestamp) then T.DrugStart
          else WRUI.BeginTimestamp
          end))
         as CostDuringInterval
from WeekRollUpInterval as WRUI, Treat2 as T
where ((T.DrugStart <= WRUT.StartPoint + 7) and
```

```
18          ( T.EffectStart >= WRUT.BeginTimestamp));
```

Listing 10. WeekRollUpIntervalCost

The `where` clause of Listing 10 requires that either the DrugStart timestamp or the EffectStart timestamp (or both of them) fall within the considered week.

Finally, the query of Listing 11 selects the tuples from the view WeekRollUpIntervalCost of Listing 10, groups them by drug (WRUIC.Drug) and by interval (WRUIC.BeginTimestamp), and sums up the CostDuringInterval. To increase readability, the query also sorts tuples by beginning timestamp and drug name (`order by` clause on WRUIC.BeginTimeStamp, WRUIA.Drug).

```
1 create view TreatRollWeek as
2 select RUIC.Drug, RUIC.BeginTimestamp, sum(CostDuringInterval) as CostDuringWeek
3 from WeekRollUpIntervalCost as RUIC
4 group by WRUIC.Drug, WRUIC.BeginTimestamp
5 order by WRUIC.BeginTimeStamp, WRUIA.Drug;
```

Listing 11. Query TreatRollWeek

7. Related Work

In this section, we will describe and discuss some proposals presented in the literature in the context of temporal models for multidimensional data, and aggregation and analysis of temporal data.

7.1. Temporal models for multidimensional data

Temporal aspects are a very important issue to deal with in the data warehouse context. The extension of conceptual data models for considering also time has been considered in different proposals [18]. Designers need to distinguish between telic and atelic data, in order to allow the system to retrieve information that accurately reflect the real world.

Khatri et al. [21] proposed a mechanism for representing telic/atelic temporal semantics at the conceptual level by using temporal annotations. Instead of defining additional constructs in the conceptual model, they proposed an annotation-based approach. They started from a conventional conceptual model for capturing the reality, then temporal features are specified on top of the initial abstraction. The authors argued that the proposed approach was not specific for the adopted conventional model,bat it can be applied to other models as well. In T+MultiDim Model, we use a hybrid technique that extends the MultiDim conceptual model and also uses annotation-based techniques. In particular, we allow the designer to annotate in a multidimensional schema the telicity properties of measures.

The T+MultiDim Model shows some features similar to those introduced in [25]. Koncilia et al. presented a formal temporal model, and its implementation, called I-OLAP, to enable users to analyze sequences interpreted as interval-based data. The model was based on several different approaches to analyze sequential data (e.g. the one discussed in [5, 19, 26, 27]). The authors considered tuples, and their attributes, from a transactional dataset: each tuple was referred to as *event*. In the model, intervals corresponded to the "gap" between two consecutive events. By considering a class of event, intervals were defined according to changes of an attribute value that quantifies an event. For instance, the event *Light* might be characterized by its status, *on* or *off*. An interval was defined over changes of such status. The I-OLAP model was based on the notions of dimensions, hierarchies, and dimension members. The authors defined the iCube (i.e., interval cube), enabling users to analyze interval-based data. iCube was characterized by user-defined functions to compute values over intervals. Indeed, these functions are similar to the adjustment function we propose in this paper, as both estimate values from two consecutive events (i.e., an interval start and end). The authors also introduced temporal operations to analyze interval data in an iCube. In particular, they defined an operator to select cells and to compute measure values over interval sequences. Indeed, such an operator is similar to the @ operator we define in this paper. Both operators allow one to select fact instances according to a temporal point or to a temporal interval. The model of Koncilia et al. differs from T+MultiDim Model since they explicitly deal with sequential data, while our proposal conceptually defines relationship between point-based data, indicating that several combinations can be considered as intervals.

7.2. Temporal data aggregation and analysis

In a data warehouse context, analysts are interested in analyzing and aggregating data (measures) w.r.t. various dimensions, which may be temporal. Performing temporal aggregation means grouping tuples according to their temporal dimension (typically their timestamp) and apply aggregate functions to measures within groups. Various kinds of temporal aggregation have been proposed in the literature [6, 10, 17, 22, 28, 31, 40]. The most relevant ones are the following:

- Instant Temporal Aggregation (ITA) computes the aggregate value at a given time instant t from the set of tuples whose timestamp contains t.
- Moving-Window Temporal Aggregation (MWTA) that, given a time interval w, computes the aggregate value at a given time instant t from the set of tuples whose timestamp is in the interval $[t-w, t]$.
- Span Temporal Aggregation (STA), on the contrary, allows the query to specify the time intervals for which to report result tuples, e.g., for each year from 2000 to 2005.
- Temporal Multidimensional Aggregation (TMDA) generalizes a variety of the previously proposed aggregation operators and offers orthogonal support for two aspects of aggregation: the first one is the definition of result groups, for which to report one or more aggregate values; the second one is the definition of aggregation groups.
- Parsimonious Temporal Aggregation (PTA) overcomes the major limitations and combines the best features of instant and span temporal aggregations. PTA computes compact aggregation summaries that reflect the most significant changes in the data over time. Users may specify the desired result size, in the same way they do when using STA. PTA produces a result with a predictable size, and minimizes the approximation error.

In [7], the authors pretesented a general framework for temporal aggregation that accommodates existing kinds of aggregation. The proposed framework was based on Klug's work for conventional non-temporal aggregation [23]. Like in [6], the framework of [7] provided orthogonal support for the definition of result and aggregation groups. The general temporal aggregation framework subsumed different types of temporal aggregation, such as ITA, STA, and MWTA: the authors pointed out open research challenges, such as the definition of an efficient evaluation algorithm for their framework, and applications involving higher-dimensional temporal data. In [7], the authors dealt with a point-based view, while in our proposal, the new conceptual multidimensional data model T+MultiDim allows the specification of connections between temporal dimensions. By combining temporal dimensions, we provide an interval-based semantics, instead of the instant-based one.

In [36], the author presented a general and application-independent method for temporal aggregation on user-defined granularities. The author analyzed the impact of the telic/atelic distinction on the problem of temporal aggregation between different temporal granularities. In particular, two types of information *split* were considered: information split occurs when some data can potentially be part of two different aggregates.

Classical OLAP operators limit the use of temporal dimensions as instant-based coordinates, but data analysts often need to combine time related dimensions within multidimensional cubes, to consider domain-related meaningful time periods. In this paper, we start from the MultiDim model and extend it to the new conceptual multidimensional data model T+MultiDim by introducing connections between temporal dimensions.

Literature considered different proposals for querying sequential data [4, 24]. Classical tools for data analytics allow one to analyze set-oriented data, without considering their order. By analyzing both data and their order dependencies, one could detect new knowledge. The focus of these approaches relates to sorting data, and thus temporal aspects are considered and discussed within this particular field.

In [24], the authors presented a novel approach, allowing one to analyze simple and complex sequences of events that are contained in a subcube specifically generated. Moreover, they created an additional dimension, Relative Time Axis, enabling the user to analyze data in a very flexible way. The generated

subcubes can be analyzed using standard OLAP operators. The approach proposed in [24] could be placed at logical level, since they defined an extension of the standard star schema model by introducing the concept and the definition of a relative (time) axis storing the difference between a given event and any other event. In [4], the authors proposed an SQL-like query language to analyze sequential data in an OLAP-like manner. Again, the proposed approach can be placed at logical level since it was based on a Relational Database Management System (RDBMS) and its related query language.

In our proposal, we consider both conceptual and logical levels, and allow designers to combine temporal dimensions to analyze data in a suitable way. Starting from the conceptual level, we allow designeres and analysts with a high-level representation of the considered domain, and provide an abstraction from the underlaying technology. This means that conceptual schema represented by the T+MultiDim data model can be mapped to various logical models.

A different proposal was related to sequenced temporal queries, i.e., queries that were evaluated at each time point, and aims at using an RDBMS to process sequenced queries. In [13], the authors proposed an extension to the relational database engine to implement sequenced temporal queries. The proposed approach reduced temporal queries to nontemporal ones over data with adjusted intervals, and it did not affect the processing of nontemporal queries. This was achieved by using a four-step transformation of the relations, that were considered in temporal queries, encompassing timestamp propagation, interval adjustment, attribute value scaling, and operator transformation. The query with temporal operator was transformed into a query which was able to treat intervals as atomic values, since adjusted intervals could be compared using equality, and could be processed natively by the DBMS.

8. Discussion and Conclusions

Data analysts often need to combine time related dimensions within multidimensional cubes, to consider domain-related meaningful time periods. However, data analysts are limited by OLAP operators in using temporal dimensions as instant-based coordinates, only. In this paper, we showed that the inclusion of interval-based temporal dimensions in the navigation of data cubes provides the analysts with the capability to view and analyze data that otherwise would not be available. We proposed the new conceptual multidimensional data model T+MultiDim that allows the specification of connections between temporal dimensions. These links indicate that users can use an interval-based semantics, instead of the usual instant-based one, by combining connected temporal dimensions. Such a combination generates interval-based dimensions. Moreover, T+MultiDim allows the user to specify whether measures are either *telic* or *atelic* on the specific interval dimension. Standard OLAP operators do not provide the capabilities to exploit such dimensions. New temporal operators able to support interval-based analyses have been proposed and defined. Such operators allow the derivation of new telic/atelic measure values from the existing ones, possibly considering new interval values associated to the derived measures. Moreover, they have also been specified at a (relational) logical level in SQL, providing different examples in the healthcare domain. Such approach may be applied to different technological architectures and could be also useful for the design and representation of temporal data that need to be analyzed/classified according to AI-based techniques [14, 20].

As an overall view of the main features of our proposal, we argue that our data model addresses and provides some original contribution in five of the eleven modeling requirements for multidimensional data models, Pedersen et al. discussed in [32].

- *Explicit hierarchies in dimensions:* dimension hierarchies should be captured explicitly by the schema. Our model fulfills this point, since they are intuitively represented both in the schema and in the graphical representation.
- *Multiple hierarchies in each dimension:* a single dimension could have several paths for aggregating data. Our model fulfills this point, since multiple paths may be modeled. A fact may also use a dimension multiple times (by playing different roles), or consider a dimension starting from any of its level.

- *Many-to-many relationships between facts and dimensions:* relationships between fact and dimensions may not always be the classical many-to-one. Our model fulfills this point, since fact-dimension and fact-fact relationships may be one-to-one, many-to-one, and many-to-many.

- *Support for aggregation semantics*: a data model is required to act as a "safety net" that catches queries that might give as results erroneous data, avoids double-counting, and allows users to specify which aggregations to consider. Our proposal fulfills this point, and also takes into account measure telic/atelic distinction. The supported aggregation semantics have been extended by T+MultiDim. Indeed, T+MultiDim supports the derivation of new aggregated telic/atelic measure values from existing telic/atelic ones. Deriving telic measures from atelic ones (and viceversa) over different intervals, by also considering aggregation, is a new relevant feature of T+MultiDim.

- *Handling change and time*: a data model is required to allow users to perform meaningful analysis across time when data change. It should be possible to easily combine data across changes over time. Our proposal partially fulfills this point, since changes are not taken into account, but we do allow users to perform meaningful analysis across various time interval-based dimensions. On this matter, the model proposed by [32] allows one to combine two time dimensions. Intuitively, the *union* is performed on member sets of the two dimensions. In our model, instead, the combination of two point-based time dimensions spawns an interval-based one.

As for ongoing work, we are applying our approach to the analysis of pharmacovigilance data [11, 12], similar to the one we used in the proposed motivating scenario. More specifically we are defining some suitable dashboards allowing users to specify temporal analyses based on the proposed OLAP operators. In this direction we are going to consider different possible implementations of OLAP operations based on temporal aggregation, evaluating the performances of both standard SQL specifications and of ad-hoc prototype extensions for sequenced temporal queries [13], within OLAP tools. A further research direction we are following deals with the application of such proposal to store and query trends, derived from clinical data to support medical decision-making tasks [11].

Acknowledgments

This work has been partially funded by the University of Verona, within 2017-2019 RIBA project "Extending OLAP data analysis with temporal and statistical operators".

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.
[2] Aristotle. *Categories. On Interpretation. Prior Analytics.* Harvard University Press, Cambridge, MA, 1938.
[3] Carlo Batini, Stefano Ceri, and Shamkant B. Navathe. *Conceptual Database Design: An Entity-Relationship Approach.* Benjamin/Cummings, 1992.
[4] Bartosz Bebel, Tomasz Cichowicz, Tadeusz Morzy, Filip Rytwiński, Robert Wrembel, and Christian Koncilia. Sequential data analytics by means of seq-sql language. In Qiming Chen, Abdelkader Hameurlain, Farouk Toumani, Roland Wagner, and Hendrik Decker, editors, *Database and Expert Systems Applications*, pages 416–431, Cham, 2015. Springer International Publishing.
[5] Bartosz Bebel, Mikolaj Morzy, Tadeusz Morzy, Zbyszko Królikowski, and Robert Wrembel. OLAP-like analysis of time point-based sequential data. In Silvana Castano, Panos Vassiliadis, Laks V. S. Lakshmanan, and Mong-Li Lee, editors, *Advances in Conceptual Modeling - ER 2012 Workshops CMS, ECDM-NoCoDA, MoDIC, MORE-BI, RIGiM, SeCoGIS, WISM, Florence, Italy, October 15-18, 2012. Proceedings*, volume 7518 of *Lecture Notes in Computer Science*, pages 153–161. Springer, 2012.
[6] Michael H. Böhlen, Johann Gamper, and Christian S. Jensen. Multi-dimensional aggregation for temporal data. In Yannis E. Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Michael Hatzopoulos, Klemens Böhm, Alfons Kemper, Torsten Grust, and Christian Böhm, editors, *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, volume 3896 of *Lecture Notes in Computer Science*, pages 257–275. Springer, 2006.
[7] Michael H. Böhlen, Johann Gamper, and Christian S. Jensen. Towards general temporal aggregation. In W. Alex Gray, Keith G. Jeffery, and Jianhua Shao, editors, *Sharing Data, Information and Knowledge, 25th British National Conference on Databases, BNCOD 25, Cardiff, UK, July 7-10, 2008. Proceedings*, volume 5071 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2008.
[8] Peter Pin-Shan Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36, 1976.

[9] Carlo Combi, Massimo Franceschet, and Adriano Peron. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 14(1):51–77, 2004.

[10] Carlo Combi, Matteo Mantovani, Alberto Sabaini, Pietro Sala, Francesco Amaddeo, Ugo Moretti, and Giuseppe Pozzi. Mining approximate temporal functional dependencies with pure temporal grouping in clinical databases. *Comp. in Bio. and Med.*, 62:306–324, 2015.

[11] Carlo Combi, Giuseppe Pozzi, and Rosalba Rossato. Querying temporal clinical databases on granular trends. *Journal of Biomedical Informatics*, 45(2):273–291, 2012.

[12] Carlo Combi, Margherita Zorzi, Gabriele Pozzani, Ugo Moretti, and Elena Arzenton. From narrative descriptions to MedDRA: automagically encoding adverse drug reactions. *Journal of Biomedical Informatics*, 84:184 – 199, 2018.

[13] Anton Dignös, Michael H. Böhlen, Johann Gamper, and Christian S. Jensen. Extending the kernel of a relational DBMS with comprehensive support for sequenced temporal queries. *ACM Trans. Database Syst.*, 41(4):26:1–26:46, November 2016.

[14] Cao-Tri Do, Ahlame Douzal Chouakria, Sylvain Marié, Michèle Rombaut, and Saeed Varasteh. Multi-modal and multi-scale temporal metric learning for a robust time series nearest neighbors classification. *Inf. Sci.*, 418:272–285, 2017.

[15] Andrew Gemino and Yair Wand. Evaluating modeling techniques based on models of learning. *Commun. ACM*, 46(10):79–84, 2003.

[16] Matteo Golfarelli and Stefano Rizzi. Temporal data warehousing: Approaches and techniques. In David Taniar and Li Chen, editors, *Integrations of Data Warehousing, Data Mining and Database Technologies - Innovative Approaches.*, pages 1–18. Information Science Reference, 2011.

[17] Juozas Gordevicius, Johann Gamper, and Michael H. Böhlen. Parsimonious temporal aggregation. *VLDB J.*, 21(3):309–332, 2012.

[18] Heidi Gregersen and Christian S. Jensen. Temporal entity-relationship models - A survey. *IEEE Trans. Knowl. Data Eng.*, 11(3):464–497, 1999.

[19] Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W. Wah, Jianyong Wang, and Y. Dora Cai. Stream cube: An architecture for multi-dimensional analysis of data streams. *Distributed and Parallel Databases*, 18(2):173–197, 2005.

[20] Bobo Huang, Li Jin, Zhihui Lu, Ming Yan, Jie Wu, Patrick C. K. Hung, and Qifeng Tang. Rdma-driven mongodb: An approach of RDMA enhanced nosql paradigm for large-scale data processing. *Inf. Sci.*, 502:376–393, 2019.

[21] Vijay Khatri, Sudha Ram, Richard T. Snodgrass, and Paolo Terenziani. Capturing telic/atelic temporal data semantics: Generalizing conventional conceptual models. *IEEE Trans. Knowl. Data Eng.*, 26(3):528–548, 2014.

[22] Nick Kline and Richard T. Snodgrass. Computing temporal aggregates. In Philip S. Yu and Arbee L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 222–231. IEEE Computer Society, 1995.

[23] Anthony C. Klug. Equivalence of relational algebra and calculus query languages having aggregate functions. *J. ACM*, 29(3):699–717, 1982.

[24] Christian Koncilia, Johann Eder, and Tadeusz Morzy. Analyzing sequential data in standard olap architectures. In Yannis Manolopoulos, Goce Trajcevski, and Margita Kon-Popovska, editors, *Advances in Databases and Information Systems*, pages 56–69, Cham, 2014. Springer International Publishing.

[25] Christian Koncilia, Tadeusz Morzy, Robert Wrembel, and Johann Eder. Interval OLAP: Analyzing interval data. In Ladjel Bellatreche and Mukesh K. Mohania, editors, *Data Warehousing and Knowledge Discovery - 16th International Conference, DaWaK 2014, Munich, Germany, September 2-4, 2014. Proceedings*, volume 8646 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2014.

[26] Mo Liu and Elke A. Rundensteiner. Event sequence processing: New models and optimization techniques. In *Proceedings of the Fourth SIGMOD PhD Workshop on Innovative Database Research*, IDAR '10, pages 7–12, New York, NY, USA, 2010. ACM.

[27] Mo Liu, Elke A. Rundensteiner, Kara Greenfield, Chetan Gupta, Song Wang, Ismail Ari, and Abhay Mehta. E-Cube: multi-dimensional event sequence analysis using hierarchical pattern query sharing. In Timos K. Sellis, Renée J. Miller, Anastasios Kementsietsidis, and Yannis Velegrakis, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 889–900. ACM, 2011.

[28] Inés Fernando Vega López, Richard T. Snodgrass, and Bongki Moon. Spatiotemporal aggregate computation: a survey. *IEEE Trans. Knowl. Data Eng.*, 17(2):271–286, 2005.

[29] Riccardo Lora, Alberto Sabaini, Carlo Combi, and Ugo Moretti. Designing the reconciled schema for a pharmacovigilance data warehouse through a temporally-enhanced ER model. In *Proceedings of the 2012 international workshop on Smart health and wellbeing*, pages 17–24. ACM, 2012.

[30] Elzbieta Malinowski and Esteban Zimányi. *Advanced Data Warehouse Design - From Conventional to Spatial and Temporal Applications*. Data-Centric Systems and Applications. Springer, 2008.

[31] Bongki Moon, Inés Fernando Vega López, and Vijaykumar Immanuel. Efficient algorithms for large-scale temporal aggregation. *IEEE Trans. Knowl. Data Eng.*, 15(3):744–759, 2003.

[32] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Inf. Syst.*, 26(5):383–423, 2001.

[33] Yoav Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artif. Intell.*, 33(1):89–104, 1987.

[34] Arie Shoshani. Summarizability. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 2880–2884. Springer US, 2009.

[35] Armin Skrbo, Begler Begovi, and Selma Skrbo. [classification of drugs using the atc system (anatomic, therapeutic, chemical classification) and the latest changes]. *Medicinski arhiv*, 58(1 Suppl 2):138141, 2004.

[36] Paolo Terenziani. Temporal aggregation on user-defined granularities. *J. Intell. Inf. Syst.*, 38(3):785–813, 2012.

[37] Paolo Terenziani and Richard T. Snodgrass. Reconciling point-based and interval-based semantics in temporal relational databases: A treatment of the telic/atelic distinction. *IEEE Trans. Knowl. Data Eng.*, 16(5):540–551, 2004.

[38] Francesco Di Tria, Ezio Lefons, and Filippo Tangorra. Cost-benefit analysis of data warehouse design methodologies. *Inf. Syst.*, 63:47–62, 2017.

[39] Alejandro A. Vaisman and Esteban Zimányi. *Data Warehouse Systems - Design and Implementation.* Data-Centric Systems and Applications. Springer, 2014.

[40] Jun Yang and Jennifer Widom. Incremental computation and maintenance of temporal aggregates. In Dimitrios Georgakopoulos and Alexander Buchmann, editors, *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 51–60. IEEE Computer Society, 2001.