Semantic Web 0 (2019) 1–26
IOS Press

Knowledge Graph OLAP

A Multidimensional Model and Query Operations for Contextualized Knowledge Graphs

Christoph G. Schuetz [a,*], Loris Bozzato [b], Bernd Neumayr [a], Michael Schrefl [a], and Luciano Serafini [b]

[a] *Department of Business Informatics – Data & Knowledge Engineering, Johannes Kepler University Linz, Austria*
E-mails: christoph.schuetz@jku.at, bernd.neumayr@jku.at, michael.schrefl@jku.at

[b] *Center for Information and Communication Technology, Fondazione Bruno Kessler, Italy*
E-mails: bozzato@fbk.eu, serafini@fbk.eu

Abstract. A knowledge graph (KG) represents real-world entities and their relationships with each other. The thus represented knowledge is often context-dependent, leading to the construction of contextualized KGs. Due to the multidimensional and hierarchical nature of context, the multidimensional OLAP cube model from data analysis is a natural fit for the representation of contextualized KGs. Traditional systems for online analytical processing (OLAP) employ cube models to represent numeric values for further processing using dedicated query operations. In this paper, along with an adaptation of the OLAP cube model for KGs, we introduce an adaptation of traditional OLAP query operations for the purposes of working with contextualized KGs. In particular, we decompose the roll-up operation from traditional OLAP into a merge and an abstraction operation. The merge operation corresponds to the selection of knowledge from different contexts whereas abstraction replaces entities with more general entities. The result of such a query is a more abstract, high-level view on the contextualized KG.

Keywords: Contextualized Knowledge Repository, Knowledge Graph Management System, Knowledge Graph Summarization, Resource Description Framework, Ontologies

1. Introduction

A knowledge graph (KG) serves organizations to represent real-world entities and their relationships with each other. KGs have been described as "large networks of entities, their semantic types, properties, and relationships" [1], as consisting of "a set of interconnected typed entities and their attributes" [2] with possibly arbitrary relationships [3]. The majority of a KG's contents are facts/instances or assertional knowledge (ABox) [3], although KGs may also include terminological/ontological knowledge (TBox) representing "the vocabulary used in the knowledge graph" [2] in order to allow for "ontological reasoning and query answering" [4] over the facts. Furthermore, a KG typically covers a variety of topics rather than focusing exclusively on a single aspect of the real world such as geographic terms [3]. The Resource Description Framework (RDF) is the standard representation format for KGs.

KGs present a wide range of potential applications, e.g., (web) search [5] and question-answering [6], intracompany knowledge management [7] and investment analysis [8]. Among the most popular examples of KGs are proprietary ones such as Google's Knowledge Graph [9] and Microsoft's Satori [10] as well as community-driven efforts such as DBpedia [11] and Wikidata [12]. More and more organizations follow suit with the development of KGs for their own purposes, necessitating the development of appropriate *Knowledge Graph Management Systems (KGMS)* [4] that facilitate knowledge exploitation, e.g., by providing KG summarization mechanisms [13].

In a strive for successful management, KGs are increasingly subject to *contextualization*, i.e., the enrichment of facts with context metadata information such as time and location. For example, in the aeronautics domain, the relevant knowledge for air traffic management is inherently context-dependent [14], especially with respect to time and location but also other context dimensions. In particular, knowledge about airport

*Corresponding author. E-mail: christoph.schuetz@jku.at.

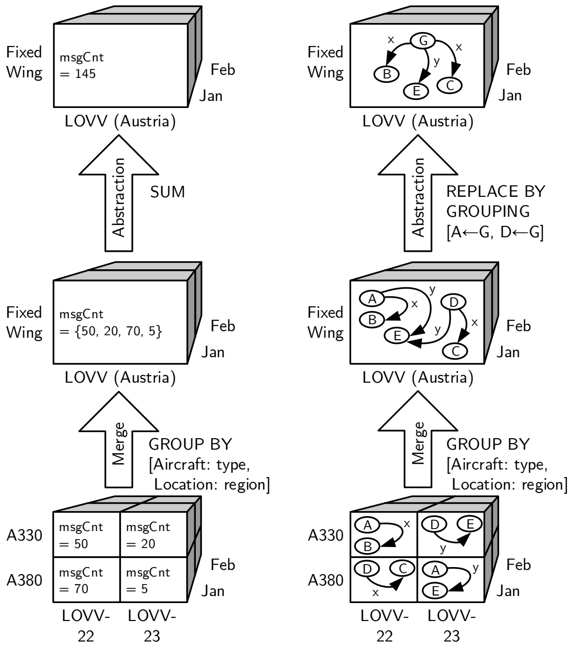
infrastructure and airspace such as operational status of runways and closure of airspace varies over time. Frameworks such as the *Contextualized Knowledge Repository* (CKR) [15] serve to organize knowledge within hierarchically ordered contexts along multiple contextual dimensions, e.g., spatial and temporal.

The multidimensional nature of context invites comparison with the multidimensional modeling approach as employed by online analytical processing (OLAP) systems for data analysis. In traditional OLAP systems, hierarchically ordered dimensions span a multidimensional space – also referred to as OLAP cube – where each point (or cell) represents an event of interest quantified by numeric measures. Similarly, context dimensions span a multidimensional space where each cell represents a context that comprises facts of a KG. OLAP systems employ multidimensional models to perform analytical queries over datasets using operations such as slice-and-dice and roll-up (see [16] for further information). In this regard, slice-and-dice refers to the selection of relevant data for the analysis whereas roll-up refers to the aggregation of the selected data in order to obtain a more abstract view on the underlying business situation. Graph OLAP [17], which is also known as InfoNetOLAP [18], extends the OLAP paradigm to structured graph data, e.g., co-author or other social graphs. In Graph OLAP, each cell of an OLAP cube contains a graph with weighted edges. Informational OLAP then refers to the combination of graphs from different cells. Topological OLAP, on the other hand, refers to the transformation of the graphs, thereby aggregating the weights of the edges. In the same vein, we extend the OLAP paradigm to KGs.

In this paper, we introduce *Knowledge Graph OLAP* (KG-OLAP), a formal framework that consists of a multidimensional model and corresponding query operations for summarizing contextualized KGs. Based on the CKR framework, KG-OLAP extends the idea of Graph OLAP to the management of contextualized KGs. Unlike Graph OLAP, which deals with more structured graphs focused on the relationships between simple entities, KG-OLAP deals with more complex, semi-structured KGs with assertional and terminological components that must be adequately dealt with. To this end, KG-OLAP cubes collect knowledge into hierarchically ordered contexts: each cell of a KG-OLAP cube corresponds to a context, with RDF triples replacing numeric measures as the contents of the cells. In KG-OLAP cubes, knowledge from the more general contexts propagates to the more specific contexts. Typically, the more general contexts establish the common terminological knowledge whereas the more specific contexts contain assertional knowledge. Central to KG-OLAP are then the notions of merge and abstraction, extending the notions of informational and topological OLAP from Graph OLAP. The merge operation combines the knowledge from different contexts whereas the abstraction operation replaces individual entities within a context with more abstract entities.

Figure 1 draws an analogy between query operations in KG-OLAP and traditional OLAP operations over numeric values. The example's setting is air traffic management, where air traffic controllers dispatch messages notifying airmen of changes in airport infrastructure and airspace characteristics. In this example, aircraft, location, and time dimensions span a three-dimensional space where each cell contains (a) numeric values in case of traditional OLAP or (b) RDF triples in case of KG-OLAP. On the left-hand side, each cell contains the number of notification messages relevant for an aircraft model in a flight information region segment for a particular month. The roll-up operation that sums up message count per aircraft type instead of individual aircraft model and flight information region instead of individual segment boils down to a sequence of merge and abstraction. First, the merge operation obtains a set of numeric values for each grouping of cells by aircraft type and flight information region. Then, the abstraction operation applies the SUM aggregation operator on the set of numeric values to obtain a single numeric value. On the right-hand side, each cell contains RDF triples representing knowledge relevant for an aircraft in a flight information region segment for a particular month. Here, the merge operation first collects the RDF triples from the individual cells that make up a grouping of cells by aircraft type and flight information region. Then, an abstraction operation replaces entities A and D by some entity G – representing the grouping of those entities A and D – in the merged graph.

We illustrate KG-OLAP using the case of (contextualized) knowledge graphs for *air traffic management (ATM)* [14, 19] in combination with the concept of ATM information cubes [20, 21]; we draw from experience in research projects on the use of semantic technologies in ATM (see [22–24]). ATM knowledge graphs potentially comprise a wide variety of topics: events, weather, flight plans, infrastructure, equipment, organizations, companies, and personnel. The running example focuses on the representation of events such as runway closures and surface contamination which affect the operational status of airport infrastructure and thus alter general ATM knowledge. In our case, merge and



(a) OLAP (b) KG-OLAP

Fig. 1. The roll-up operation as a combination of merge and abstraction in (a) traditional OLAP with numeric values and (b) Knowledge Graph OLAP

abstraction serve to obtain a *management summary* for pilot briefings or post-operational analysis, providing a more abstract view on the knowledge graph containing only the relevant ATM knowledge for a particular situation. Another potential application of KG-OLAP is the representation of business reports and their subsequent analysis using merge and abstraction [25].

In this paper, we make a number of contributions with respect to contextualized KG management:

(i). a formalization of contextualized KGs with hierarchical contexts and knowledge propagation.
(ii). a set of query operations for working with contextualized KGs.
(iii). an experimental analysis of run time performance of working with contextualized KGs.

The remainder of this paper is organized as follows. In Section 2, we present the use case that serves to illustrate KG-OLAP and motivate the approach. In Section 3, we introduce the modeling language for KG-OLAP cubes. In Section 4, we present query operations for KG-OLAP cubes. In Section 5, we discuss implementation of the approach. In Section 6, we review related work. We conclude with a discussion and an outlook on future work.

2. Use Case: Air Traffic Management

Modern air traffic management (ATM) strives to ensure safe flight operations through careful management, analysis, and advance planning of air traffic flow as well as timely provisioning of relevant information in form of messages. The exchange of data/information between ATM stakeholders is of paramount importance in order to foster common situational awareness for improved efficiency, safety, and quality in planning and operations. In this regard, situational awareness refers to a “person’s knowledge of particular task-related events and phenomena” [26], i.e., knowledge about the world relevant for ATM, which must be accurately represented and conveyed to the various stakeholders. To this end, ATM relies on a multitude of standardized data/information (exchange) models, e.g., the *Aeronautical Information Exchange Model* (AIXM), the *Flight Information Exchange Model* (FIXM), the *ICAO Meteorological Information Exchange Model* (IWXXM), and the *ATM Information Reference Model* (AIRM). Furthermore, a growing interest in the use of semantic technologies in ATM (see [27] for an overview) has led to the development of domain ontologies, e.g., the *NASA ATM Ontology* [28] and the *AIRM Ontology* [29], and knowledge graphs for ATM [19, 30].

Among the most common types of messages exchanged in ATM are Notices to Airmen. A Notice to Airmen (NOTAM) – or Digital NOTAM (DNOTAM) when in electronic form using AIXM format – is a message that conveys important information about temporary changes in flight conditions to aircraft pilots [31], e.g., aerodrome, runway, and taxiway closures, surface conditions, and construction activities (see [32] for a list of event scenarios) but also airspace restrictions. Air traffic controllers dispatch relevant DNOTAMs to pilots prior to a flight, possibly with additional annotations and further background knowledge. Automated rule-based filtering and prioritization techniques provide assistance to controllers and pilots in determining the relevance and importance of DNOTAM messages for a particular flight [22].

Messages shape the knowledge about the world as relevant for ATM. For example, a DNOTAM (Listing 1) may change the knowledge about the runways of a particular airport by announcing the temporary closure of a runway due to snow. To this end, a DNOTAM employs different *time slices*. A *baseline* timeslice defines the regular, baseline knowledge whereas a *tempdelta* timeslice announces temporary changes of the baseline knowledge. Instead of the baseline timeslice,

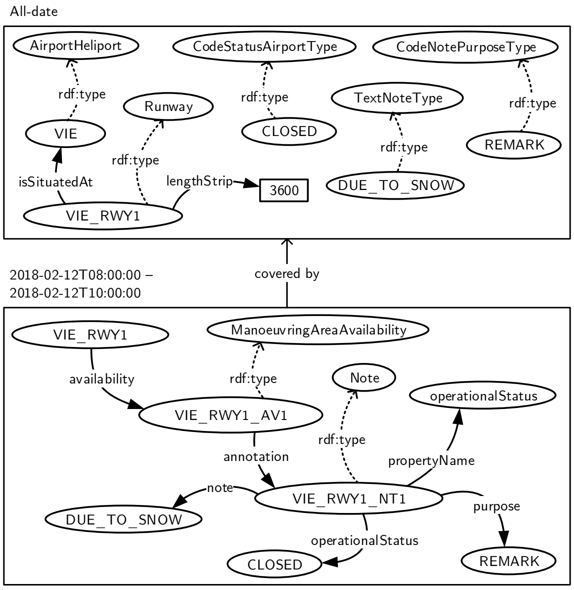


Fig. 2. A contextualized KG based on the DNOTAM in Listing 1

a DNOTAM typically employs the *snapshot* timeslice – the baseline blended with tempdelta knowledge. In the example DNOTAM in Listing 1, the encoded snapshot/baseline knowledge consists of the definition of the designator of Vienna airport (Lines 6-13) and the definition of various attributes of a runway at Vienna airport (Lines 18-28) per 12 February 2018 at 8:00 am. The tempdelta knowledge consists of the notification of a runway closure due to snow (Lines 31-50).

The knowledge encoded in DNOTAMs is more naturally represented using contextualized KGs [14]. Figure 2 illustrates the contextualized representation of the knowledge encoded in the DNOTAM from Listing 1 along a temporal dimension. The *all-date* context defines general knowledge about various infrastructure elements, which hardly changes. The temporal context for the timespan from 8:00-10:00 am on the 12 February 2018, on the other hand, defines knowledge about a temporarily reduced availability – a closed operational status – due to snow. Other context dimensions may also serve to organize ATM knowledge into contextualized KGs [14], e.g., geography, topic.

Manual and automated rule-based filtering, combination, and enrichment activities allow for the collection of individual DNOTAMs (or other messages) along with additional information and domain knowledge into *semantic containers* [23] for different contexts of relevance. For example, a container may comprise the relevant DNOTAMs in the context of a flight from Dubai to

Listing 1: An example DNOTAM in XML notifying of a runway closure in Vienna due to snow

```
1  <AIXMBasicMessage>
2   <hasMember>
3    <AirportHeliport id="VIE">
4     <timeSlice>
5      <AirportHeliportTimeSlice
6          id="VIE_TS1">
7       <validTime>
8        <TimeInstant
9            id="VIE_TS1_TI">
10         <timePosition>
11           2018-02-12T08:00:00
12      <interpretation>SNAPSHOT
13      <designator>VIE
14  <hasMember>
15   <Runway id="VIE_RWY1">
16    <timeSlice>
17     <RunwayTimeSlice
18         id="VIE_RWY1_TS1">
19      <validTime>
20       <TimeInstant
21           id="VIE_RWY1_TS1_TI">
22        <timePosition>
23          2018-02-12T08:00:00
24      <interpretation>SNAPSHOT
25      <isSituatedAt
26          xlink:href="VIE"/>
27      <designator>VIE_RWY1
28      <lengthStrip>3600
29    <timeSlice>
30     <RunwayTimeSlice
31         id="VIE_RWY1_TS2">
32      <validTime>
33       <TimePeriod
34           id="VIE_RWY1_TS2_TP">
35        <beginPosition>
36          2018-02-12T08:00:00
37        <endPosition>
38          2018-02-12T10:00:00
39      <interpretation>TEMPDELTA
40      <availability>
41       <RunwayAvailabilty
42           id="VIE_RWY1_AV1">
43        <operationalStatus>CLOSED
44        <annotation>
45         <Note
46             id="VIE_RWY1_NT1">
47          <propertyName>
48            operationalStatus
49          <purpose>REMARK
50          <note>DUE TO SNOW
```

Vienna on a particular day. Now depending on the addressee, e.g., pilot or network manager, and the task that the information serves, e.g., operational or analytical, the containers may have different *context dimensions*. For example, a pilot, in order to prepare for and safely conduct a flight, might prefer to receive DNOTAMs that have been collected into different containers per combination of flight phase, route or ground segment, event scenario, and importance. The pilot could then select the appropriate containers at the right moment without being overloaded with information. This application basically corresponds to a combination of previous work on automated rule-based filtering and prioritization of DNOTAMs [22] with semantic containers [23].

Post-operational analytical tasks may also leverage contextualized ATM knowledge. In air traffic flow and capacity management (ATFCM) – one of the core activities in ATM – a post-operations team analyzes operational events in order to identify valuable lessons learned for the benefit of future operations and produces an overview of occurred incidents [33, p. 131]. A data warehouse provides the post-operations team with statistical data about flight operations [33, p. 130]. In addition, a repository of semantic containers may comprise contextualized ATM knowledge extracted from DNOTAMs and other types of messages by temporal relevance, route or ground segment, aircraft model, and importance. By analyzing such ATM knowledge, an air traffic flow post-operations team may gain a more comprehensive picture of past air traffic operations.

In the remainder of this paper, we illustrate the KG-OLAP approach using the cases of ATM knowledge representation for pilot briefings and post-operational analysis in ATFCM. In particular, we propose to employ a KG-OLAP cube of hierarchically ordered semantic containers comprising ATM knowledge, obtained from DNOTAMs according to the AIXM standard [34] and possibly other sources, for different contexts of relevance – a *cube of ATM knowledge*. Using the *merge* operation, an air traffic controller or a post-operations team in ATFCM may combine ATM knowledge from different contexts. For example, the relevant ATM knowledge per aircraft model and importance could be combined to obtain the ATM knowledge per aircraft type and importance category; the ATM knowledge per day and geographic segment could be combined to obtain the ATM knowledge per month and geographic region. Various incarnations of the *abstraction* operation then serve to obtain a more abstract representation of the ATM knowledge. For example, instead of indicating specific closures of individual runways or taxiways, the abstract ATM knowledge would indicate closures of runways and taxiways in general for aircraft with certain characteristics. The abstract ATM knowledge constitutes a *management summary* of the detailed knowledge, providing a high-level overview. Besides DNOTAMs, other types of aeronautical information relevant to ATFCM, e.g., flight data in FIXM and meteorological messages in IWXXM, could similarly serve to populate the cube of ATM knowledge. In this regard, we have previously proposed the notion of ATM information cubes [20, 21] which, however, comprise the ATM messages themselves rather than contextualized knowledge graphs derived from possibly various different sources.

In our scenario, RDF serves as the common representation language for ATM knowledge even though XML is the native format of DNOTAMs in the AIXM standard. AIXM, however, builds on the Geography Markup Language (GML), the initial proposal of which was based directly on RDF, with subsequent editions continuing to "borrow many ideas from RDF" [35, p. 20], including the GML's object-property model [35, p. 16]. Other ATM information (exchange) models could similarly be represented using RDF, e.g., IWXXM for weather information. Furthermore, ontologies such as the AIRM Ontology [29] and the NASA ATM Ontology [19, 28, 30] could serve for the representation of ATM knowledge.

3. Multidimensional Model

In this section, we introduce the *KG-OLAP cube model* for the management of contextualized KGs. We first introduce the model informally before providing a formal definition. We define the model as a specialization of the Contextualized Knowledge Repository (CKR) framework [15, 36].

3.1. KG-OLAP Cube Model

KG-OLAP adapts the multidimensional modeling paradigm from data warehousing (see [16]) in order to organize multidimensional KGs. Hence, the *KG-OLAP cube* is the central modeling element. Following the basic structure of the CKR framework, the KG-OLAP cube consists of two distinct layers: an upper and a lower layer. The upper layer describes the structure and properties of a cube's cells; the lower layer specifies cell contents. The two layers employ distinct and possibly disjoint languages.

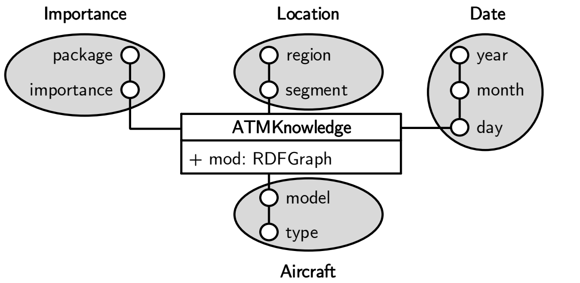


Fig. 3. A KG-OLAP cube with its dimensions and levels in DFM notation for the organization of KGs in air traffic management

A KG-OLAP cube's upper layer defines the multidimensional structure of a cube and associates specific knowledge modules with individual cube cells. Intuitively, the cube's *dimensions* (e.g., time, location) span a multidimensional space, the points of which are referred to as *cells*[1]. The dimensions are hierarchically organized into *levels*. The definition of a cube's dimensions and their hierarchical organization – the cube's multidimensional structure – into levels is referred to as *KG-OLAP cube schema*.

Example 1 (KG-OLAP cube schema)**.** Figure 3 illustrates, in Dimensional Fact Model (DFM) notation [37], a KG-OLAP cube's schema with its dimensions and levels. The presented KG-OLAP cube organizes relevant knowledge for air traffic management (ATM). The box in the center represents the cells of the cube: Each cell contains an RDF graph that encodes the contextualized ATM knowledge – the cell's knowledge module. The cube has four context dimensions that characterize the cells: *importance*, *location*, *date*, and *aircraft*. Hence, the ATM knowledge graph is partitioned by importance of the knowledge for a particular aircraft model on a certain day within a geographic segment. The importance dimension has the levels importance and package, the location dimension has segment and region, the date dimension has day, month, and year, the aircraft dimension has model and type. ◇

The dimension members (e.g., June 2016, Vienna) of a KG-OLAP cube are organized in a complete linear order, which is referred to as *roll-up* relationship. For example, the month June 2016 rolls up to the year 2016 and Vienna rolls up to Austria. Dimension members belong to *levels*, which define the granularity of the dimension members (e.g. month and year, country and city). The levels serve to aggregate individual cells of a cube (see Section 4). Levels are likewise organized in a complete linear order, which is similarly referred to as roll-up relationship. For example, month rolls up to year and city rolls up to country.

Example 2 (Dimensions and levels)**.** Figure 4 shows an ordering of dimension members and the corresponding levels, which is used in the running example cube of ATM knowledge. A tree represents each dimension, the name of the dimension depicted above the tree. Each node represents a dimension member, the caption next to each node shows the respective dimension member's name. An edge between two nodes represents a roll-up relationship between the respective dimension members, from bottom to top. On the left hand side of each tree are the levels of the dimension members, ordered from most general to most specific. Each dimension has an implicit *all* level, which is not shown in Fig. 3. For example, in the importance dimension, the FlightCritical member at the importance level rolls up to the Essential member at the package level, which rolls up to the All-importance member at the all-importance level. The hierarchical ordering of the dimension members mirrors the hierarchical ordering of the levels. ◇

The dimension members characterize the cells of a KG-OLAP cube: Each cell has a set of dimension members as identifying attributes and the dimension hierarchies organize the cells into a hierarchical structure. For example, the combination of dimension members June 2016 and Austria identifies a particular cell. The hierarchical order of dimension members determines the *coverage* relationship, which is a partial order between cells. For example, the cell identified by the combination of dimension members June 2016 and Austria covers the cell identified by the combination of dimension members 23rd June 2016 and Vienna. With each cell, a KG-OLAP cube furthermore associates a knowledge module – a set of knowledge facts.

Example 3 (KG-OLAP cube cells)**.** Figure 5 shows a set of cells according to the KG-OLAP cube schema in Fig. 3; the contents of the knowledge modules are shown in Fig. 6 (see Example 4). The c_0 cell associates the K_0 knowledge module, which contains the knowledge facts relevant for all importance categories, all locations, on all dates, and for all aircraft. The c_1 cell associates the K_1 knowledge module, which contains the knowledge facts relevant for all importance categories, the LOVV (Austria) flight information region, the year 2020, and all aircraft. The c_0 cell covers the c_1 cell,

[1] Alternatively, data warehouse literature refers to those points as *facts* – a term which in order to avoid confusion we reserve to designate the statements in a KG.

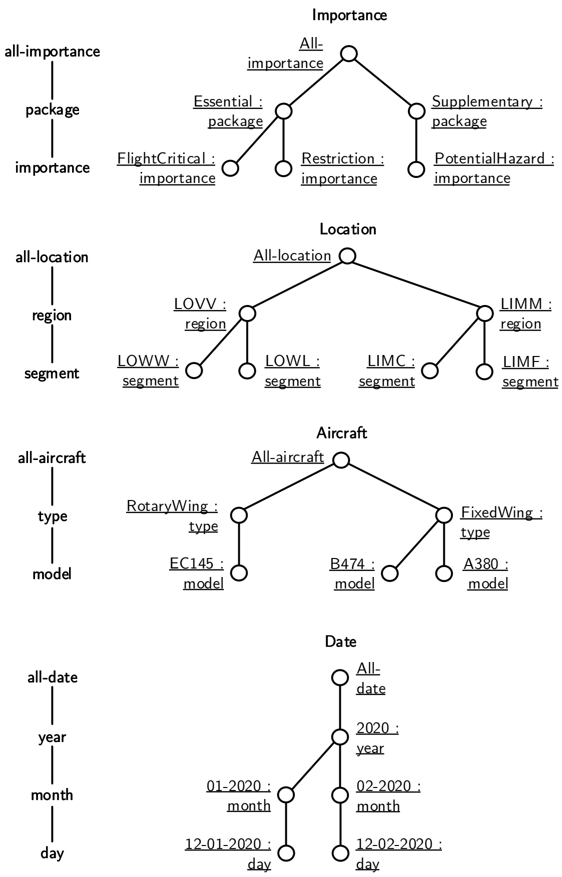


Fig. 4. Example hierarchies for the context dimension members

which is determined by the hierarchical order of the identifying dimension members: All of c_1's attribute dimension members are equal or roll up to c_0's attribute members for the respective dimensions. Context coverage indicates a sort of "extension" relationship: The covered cells inherit the knowledge in the modules of the covering cells.

The c_2 cell, which is covered by c_1, associates the K_2 knowledge module, which contains the knowledge facts relevant for the Supplementary briefing package, the LOVV region, the month 02-2020, and FixedWing aircraft. The c_3 cell, which is covered by c_1, associates the K_3 knowledge module, which contains the knowledge facts relevant for all importance categories, the LOWW (Vienna airport) segment, the year 2020, and all aircraft. The cells c_2 and c_3 are not in a coverage relationship: c_2's importance, temporal, and aircraft attributes are more general than c_3's attributes in the respective dimensions but c_3's location attribute is more general than c_2's.

The c_4 cell, which is covered by c_3, associates the K_4 knowledge module, which contains the knowledge facts relevant for the Essential briefing package, the LOWW segment, the day 12-02-2020, and the A380 aircraft model. The cells c_5 and c_6, which are covered by c_4, associate the K_5 and K_6 knowledge modules, respectively, which contain the knowledge facts of FlightCritical and Restriction importances, respectively, relevant for the LOWW segment, the day 12-02-2020, and the A380 aircraft model. The c_7 cell, which is covered by c_2 and c_3, associates the K_7 knowledge module, which contains the knowledge facts of PotentialHazard importance relevant for the LOWW segment, the day 12-02-2020, and the A380 aircraft model. ◇

A KG-OLAP cube's lower layer consists of the actual knowledge modules that are associated with the individual cells. A knowledge module contains statements valid in the context of the associated cell. The knowledge inside each module is specified using an *object language* and expresses the facts and axioms valid in the specific context defined by the cell. Furthermore, knowledge propagates downwards along the coverage relationships, from the more general to the more specific contexts.

Example 4 (Knowledge modules)**.** Figure 6 defines (in a description logics-style syntax) the contents of the knowledge modules K_0-K_7 associated with the KG-OLAP cube cells c_0-c_7 from Fig. 5. The representation of the module contents follows the AIXM standard [34] with minor modifications for illustration purposes.

The K_0 module (Rows 1-24) defines terminological knowledge valid across all contexts. In particular, the module defines Runway and Taxiway as subconcepts of RunwayTaxiway (Row 1). The isSituatedAt property links infrastructure, e.g., a RunwayTaxiway, to an AirportHeliport (Row 2). The availability property links infrastructure to a ManoeuvringAreaAvailability, which issues a warning (Row 4), e.g., of inspection activity. The warningAdjacent data property indicates whether the warning applies to an area adjacent to the infrastructure (Rows 5 and 6). The operationalStatus property (Row 7) indicates the general availability of the infrastructure, e.g., closed or limited, and the usage property (Row 8 and 9) specifies a ManoeuvringAreaUsage which indicates usageType (Row 10), e.g., allow or forbid, of a specific operation (Row 11), e.g., landing, for aircraft with certain characteristics (Rows 12 and 13), e.g., aircraft with a weight above 140 or aircraft with a wingspan below 8. In this regard, the weight property (Row 14) of the AircraftCharacteristic concept specifies a weight

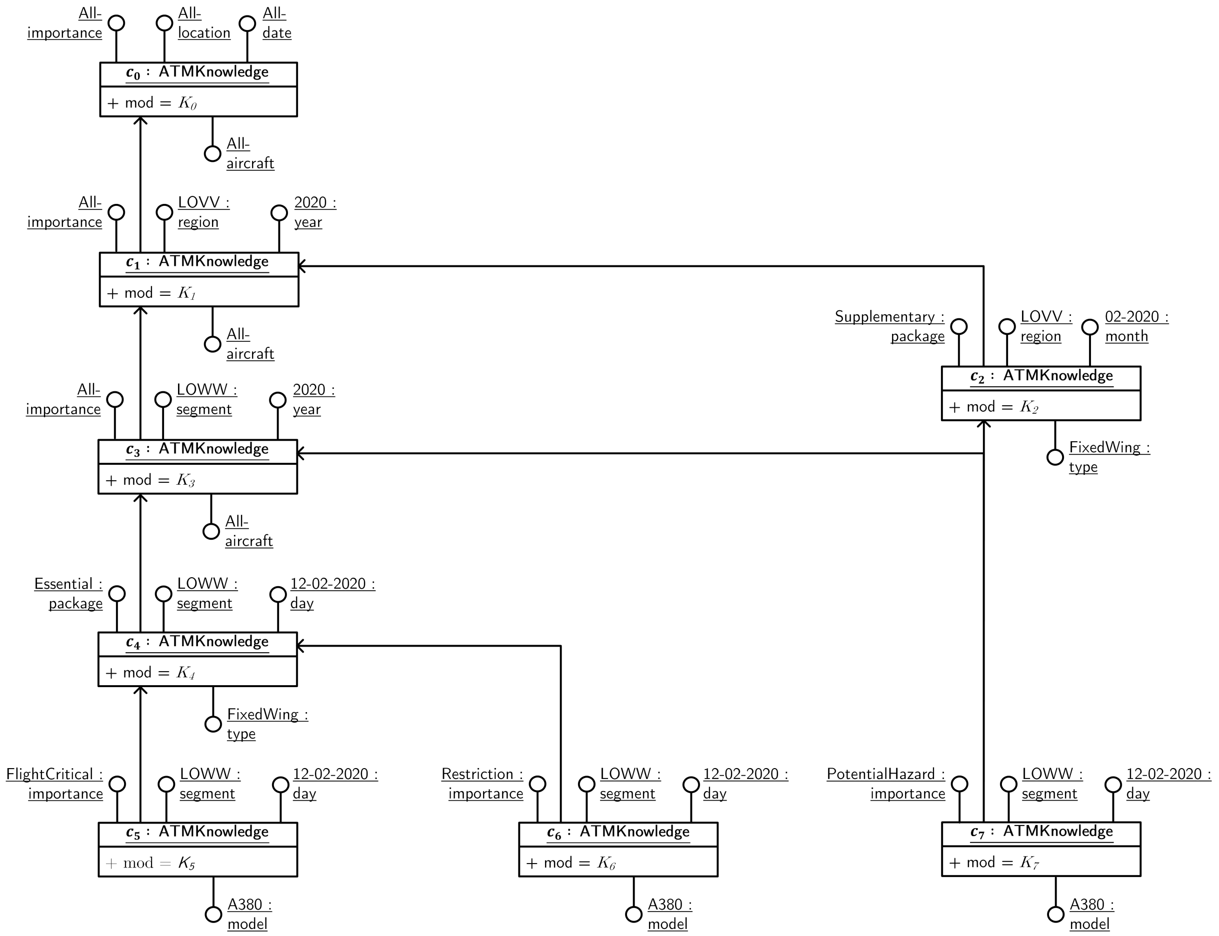


Fig. 5. An example instance of the KG-OLAP cube schema in Fig. 3. The arrows denote coverage relationships between contexts; the covered context points to the covering. The contents of the knowledge modules K_0-K_7 are defined in Fig. 6.

value and weightInterpretation specifies whether the weight value signifies an upper (weight interpretation below) or lower threshold (above); the wingspan property (Rows 16 and 17) works analogously. The contaminant property (Row 18) indicates SurfaceContamination of infrastructure, e.g., runways and taxiways. A SurfaceContamination has an overall depth (Row 19) and several SurfaceContaminationLayers specified via the layer property (Rows 20 and 21). A SurfaceContaminationLayer has a contaminationType (Row 22), e.g., compact_snow or dry_snow. The contamination types compact_snow and dry_snow, in turn, are grouped into snow as indicated by the grouping property (Rows 23 and 24). Moreover, the frequency property (Row 25) indicates the frequency of a Very High Frequency (VHF) Omni-Directional Range (VOR) navigation aid equipment used for determining an aircraft's position.

The K_1 module (Rows 26-29) defines concepts and individuals relevant for the LOVV (Austria) region in 2020. In particular, the module defines airportLOWW (Vienna airport) as an individual of the AirportHeliport class (Row 26) and the vorLNZ (VOR near the city of Linz) as an individual of the VOR class (Row 27). Furthermore, the module defines the HeavyWeight concept (Row 28) for aircraft characteristics that designate aircraft with a weight above 136 and the DeepContamination concept (Row 29) for surface contaminations with a depth greater than 0.2.

The K_2 module (Row 30) defines supplementary knowledge relevant for FixedWing aircraft in the LOVV region in February 2020. In this month, the vorLNZ navigation aid operates on a frequency of 116.8.

The K_3 module (Rows 31-34) defines knowledge relevant in the LOWW (Vienna airport) segment of the LOVV region in 2020. That knowledge consists

K_0	Runway, Taxiway $\sqsubseteq$ RunwayTaxiway	(1)
	$\top \sqsubseteq \forall$isSituatedAt.AirportHeliport	(2)
	$\top \sqsubseteq \forall$availability.ManoeuvringAreaAvailability	(3)
	$\exists$warning.$\top \sqsubseteq$ ManoeuvringAreaAvailability	(4)
	$\exists$warningAdjacent.$\top \sqsubseteq$ ManoeuvringAreaAvailability	(5)
	$\top \sqsubseteq \forall$warningAdjacent.Boolean	(6)
	$\exists$operationalStatus.$\top \sqsubseteq$ ManoeuvringAreaAvailability	(7)
	$\exists$usage.$\top \sqsubseteq$ ManoeuvringAreaAvailability	(8)
	$\top \sqsubseteq \forall$usage.ManoeuvringAreaUsage	(9)
	$\exists$usageType.$\top \sqsubseteq$ ManoeuvringAreaUsage	(10)
	$\exists$operation.$\top \sqsubseteq$ ManoeuvringAreaUsage	(11)
	$\exists$aircraft.$\top \sqsubseteq$ ManoeuvringAreaUsage	(12)
	$\top \sqsubseteq \forall$aircraft.AircraftCharacteristic	(13)
	$\exists$weight.$\top \sqsubseteq$ AircraftCharacteristic	(14)
	$\exists$weightInterpretation.$\top \sqsubseteq$ AircraftCharacteristic	(15)
	$\exists$wingspan.$\top \sqsubseteq$ AircraftCharacteristic	(16)
	$\exists$wingspanInterpretation.$\top \sqsubseteq$ AircraftCharacteristic	(17)
	$\top \sqsubseteq \forall$contaminant.SurfaceContamination	(18)
	$\exists$depth.$\top \sqsubseteq$ SurfaceContamination	(19)
	$\exists$layer.$\top \sqsubseteq$ SurfaceContamination	(20)
	$\top \sqsubseteq \forall$layer.SurfaceContaminationLayer	(21)
	$\exists$contaminationType.$\top \sqsubseteq$ SurfaceContaminationLayer	(22)
	grouping(compact_snow, snow)	(23)
	grouping(dry_snow, snow)	(24)
	$\exists$frequency.$\top \sqsubseteq$ VOR	(25)
K_1	AirportHeliport(airportLOWW)	(26)
	VOR(vorLNZ)	(27)
	HeavyWeight $\sqsubseteq \exists$weight.$\{\geqslant 136\} \sqcap \exists$weightInterpretation.$\{$above$\}$	(28)
	DeepContamination $\sqsubseteq \exists$depth.$\{\geqslant 0.2\}$	(29)
K_2	frequency(vorLNZ, 116.8)	(30)
K_3	Runway(runway16/34)	(31)
	isSituatedAt(runway16/34, airportLOWW)	(32)
	Taxiway(taxiway10/004)	(33)
	isSituatedAt(taxiway10/004, airportLOWW)	(34)
K_4	contaminant(runway16/34, runway16/34-contam#265)	(35)
	depth(runway16/34-contam#265, 0.2)	(36)
	layer(runway16/34-contam#265, runway16/34-layer#265-1)	(37)
	contaminationType(runway16/34-layer#265-1, dry_snow)	(38)
	contaminant(taxiway10/004, taxiway10/004-contam#343)	(39)
	depth(taxiway10/004-contam#343, 0.4)	(40)
	layer(taxiway10/004-contam#343, taxiway10/004-layer#343-1)	(41)
	contaminationType(taxiway10/004-layer#343-1, compact_snow)	(42)
K_5	availability(runway16/34, runway16/34-avail#241)	(43)
	operationalStatus(runway16/34-avail#241, closed)	(44)
	usage(runway16/34-avail#241, runway16/34-usage#241-1)	(45)
	usageType(runway16/34-usage#241-1, forbid)	(46)
	operation(runway16/34-usage#241-1, landing)	(47)
	aircraft(runway16/34-usage#241-1, characteristic#556)	(48)
	weight(characteristic#556, 140)	(49)
	weightInterpretation(characteristic#556, above)	(50)
K_6	availability(taxiway10/004, taxiway10/004-avail#352)	(51)
	operationalStatus(taxiway10/004-avail#352, closed)	(52)
	usage(taxiway10/004-avail#352, taxiway10/004-usage#352-1)	(53)
	usageType(taxiway10/004-usage#352-1, forbid)	(54)
	aircraft(taxiway10/004-usage#352-1, characteristic#677)	(55)
	weight(characteristic#677, 150)	(56)
	weightInterpretation(characteristic#677, above)	(57)
	usage(taxiway10/004-avail#352, taxiway10/004-usage#352-2)	(58)
	usageType(taxiway10/004-usage#352-2, allow)	(59)
	aircraft(taxiway10/004-usage#352-2, characteristic#723)	(60)
	wingspan(characteristic#723, 8)	(61)
	wingspanInterpretation(characteristic#723, below)	(62)
K_7	availability(runway16/34, runway16/34-avail#528)	(63)
	warning(runway16/34-avail#528, inspection)	(64)
	warningAdjacent(runway16/34-avail#528, true)	(65)

Fig. 6. Example contents of the KG-OLAP cube cells in Fig. 5

of the definition of individuals representing a runway (runway16/34) and a taxiway (taxiway10/004), which are both situated at Vienna airport (airportLOWW).

The K_4 module (Rows 35-42) defines knowledge essential for aircraft of the FixedWing type in the LOWW segment on the 12th February 2020. On that day, runway16/34 is covered by a contaminant (Row 35) with a depth of 0.2 (Row 36) consisting of one layer (Row 37) of dry_snow (Row 38). Moreover, taxiway10/004 is covered by a contaminant (Row 39) with a depth of 0.4 (Row 40) consisting of one layer (Row 41) of compact_snow (Row 42).

The K_5 module (Rows 43-50) defines knowledge of flight critical importance for A380 aircraft in the LOWW segment on the 12th February 2020. On that day, runway16/34's availability (Row 43) indicates a closed operational status (Row 44) where usage is forbidden (Rows 45 and 46) for landing aircraft (Row 47) with a weight above 140 (Rows 48-50).

The K_6 module (Rows 51-62) defines knowledge about restrictions relevant for A380 aircraft in the LOWW segment on the 12th February 2020. On that day, taxiway10/004's availability (Row 51) indicates a closed operational status (Row 52) where usage is forbidden (Rows 53 and 54) for all aircraft with a weight above 150 (Rows 55-57). Usage of taxiway10/004 is allowed (Rows 58 and 59) for all aircraft with a wingspan below 8 (Rows 60-62).

The K_7 module (Rows 63-65) defines knowledge about potential hazard relevant for A380 aircraft in the LOWW segment on the 12th February 2020. A warning notifies of an inspection adjacent to runway16/34. $\diamond$

The knowledge from the higher-level cells propagates to the covered lower-level cells; on the other hand, the knowledge associated to such lower-levels cells specialize the more general knowledge inherited from the higher levels. This organization facilitates the combination of knowledge across cells in the course of data analysis: the higher-level facts contain a shared conceptualization of business terms that may be extended by lower-level facts. On the other hand, the actual contents of lower-level cells are defined in terms of the shared conceptualization provided by the higher-level facts. The propagated knowledge is also available for reasoning.

Example 5 (Inference)**.** Given the cells in Fig. 5 and the corresponding knowledge modules in Fig. 6, in the K_4 module, runway16/34-contam#265 (Rows 35 and 36) and taxiway10/004-contam#343 (Rows 39 and 40) can be classified as DeepContamination according to the definition of this concept in the K_1 module (Row 29), which is inherited by the lower-level cells. Furthermore, characteristic#556 in the K_5 module (Rows 48-50) and the characteristic#677 in the K_6 module (Rows 55-57) can be classified as HeavyAircraft according to the definition of this concept in the K_1 module (Row 28). ◇

3.2. Formalization

In the following, we adapt and extend the definitions of the CKR framework – building on the CKR definition [36, 38] in a generic description logic (DL) language [39] – in order to fit the needs of KG-OLAP and its query operations (see Section 4).

3.2.1. Basic Definitions

We first define the basic notions of a KG-OLAP cube before relating the KG-OLAP cube definitions to the CKR framework. The multidimensional structure is expressed using a *cube vocabulary* Ω, which is a DL signature. Ω is composed of the mutually disjoint sets NR_Ω of atomic roles, NC_Ω of atomic concepts, and NI_Ω of individual names. The vocabulary further specifies a set $\mathsf{F} \subseteq \mathrm{NI}_\Omega$ of *cell names*, a set $\mathsf{D} \subseteq \mathrm{NR}_\Omega$ of *dimensions*, a set $\mathsf{L} \subseteq \mathrm{NI}_\Omega$ of *levels*, a set $\mathsf{I} \subseteq \mathrm{NI}_\Omega$ of *dimension members*, and for every dimension $\mathsf{E} \in \mathsf{D}$, a set $\mathsf{D}_\mathsf{E} \subseteq \mathsf{I}$ of dimension members of E (cf. dimensional structure in [15]). The *cube language* $\mathcal{L}_\Omega$ for expressing a KG-OLAP cube's multidimensional structure is thus a DL language over cube vocabulary Ω.

For every dimension $\mathsf{A} \in \mathsf{D}$, we define the role $\prec_\mathsf{A}$ of *dimensional ordering* for A as a strict partial order relation over dimension members D_A, i.e., an irreflexive, transitive and antisymmetric role over couples $\langle d, d' \rangle \in \mathsf{D}_\mathsf{A} \times \mathsf{D}_\mathsf{A}$. In the following, we also employ the non-strict dimensional ordering $\preceq_\mathsf{A}$ over D_A. In general, we assume that each dimension is ordered in a simple hierarchy (or tree). Thus, if we denote with $\dot{\prec}_\mathsf{A}$ the direct successor relation in the dimensional ordering, we require that $d \dot{\prec}_\mathsf{A} e_1$ and $d \dot{\prec}_\mathsf{A} e_2$ implies $e_1 = e_2$, i.e., $\dot{\prec}_\mathsf{A}$ is functional, and we assume that, for every D_A, there is a maximum, i.e., an *all* level with one *all* member. We further formally define for every dimension $\mathsf{A} \in \mathsf{D}$ its set $\mathsf{L}_\mathsf{A} \subseteq \mathsf{L}$ of levels. We define the role $\prec^\mathsf{L}_\mathsf{A}$ as a strict order relation over L_A and a role lev associating dimension members in D_A to levels in L_A. For example, in Fig. 4, the Date dimension has dimension member ordering 12-02-2020 $\prec$ 02-2020 $\prec$ 2020 $\prec$ All-date. The Date dimension further has the hierarchical order of levels day $\prec^\mathsf{L}_\mathsf{Date}$ month $\prec^\mathsf{L}_\mathsf{Date}$ year $\prec^\mathsf{L}_\mathsf{Date}$ all-date.

In order to define the hierarchical order of cells, we adapt the definition of dimensional vector and context coverage from the CKR definition in [15]. Let $|\mathsf{D}| = k$, we define a *dimensional vector* as the set

$$\mathbf{d} = \{A_1 := d_1, \ldots, A_k := d_k\}$$

s.t. for j with $1 \leqslant j \leqslant k$, $d_j \in \mathsf{D}_{\mathsf{A_j}}$. We call *multidimensional space* $\mathfrak{D}_\Omega$ the set of all dimensional vectors of Ω. We denote with d_A the value given by $\mathbf{d}$ to the dimension A. For example, given the dimensional vector $\mathbf{d} = \{$Importance := All-importance, Location := LOVV, Time := 2020, Aircraft := All-aircraft$\}$, d_{Location} is equal to LOVV.

Given a dimensional vector, we associate with it a *cell name* using the function $\mathsf{cn} : \mathfrak{D}_\Omega \to \mathsf{F}$. We require cn to be bijective, that is, each cell name is associated with a point in the multidimensional space and, conversely, the cell name can be interpreted as the unique identifier of the corresponding dimensional vector. For example, in Fig. 5, given the dimensional vector $\mathbf{e} = \{$Importance := FlightCritical, Location := LOWW, Time := 12-02-2020, Aircraft := A380$\}$, we have $\mathsf{cn}(\mathbf{e}) = \mathsf{c}_5$. We denote with cn^- the inverse function of cn.

Let $\mathbf{d}, \mathbf{e} \in \mathfrak{D}_\Omega$, we say that $\mathbf{d} \preceq \mathbf{e}$ iff $d_\mathsf{A} \preceq e_\mathsf{A}$ for each $\mathsf{A} \in \mathsf{D}$. Similarly, given $\mathsf{c}_1, \mathsf{c}_2 \in \mathsf{F}$, we say that c_2 *covers* c_1 and we write $\mathsf{c}_1 \preceq \mathsf{c}_2$ iff $\mathsf{cn}(\mathbf{d}) = \mathsf{c}_1$ and $\mathsf{cn}(\mathbf{e}) = \mathsf{c}_2$ and, for every $\mathsf{A} \in \mathsf{D}$, $d_\mathsf{A} \preceq e_\mathsf{A}$. For example, given the dimension hierarchies from Fig. 4, for the dimensional vectors $\mathbf{d} = \{$Importance := All-importance, Location := LOVV, Time := 2020, Aircraft := All-aircraft$\}$ and $\mathbf{e} = \{$Importance := FlightCritical, Location := LOWW, Time := 12-02-2020, Aircraft := A380$\}$, we have $\mathbf{e} \preceq \mathbf{d}$. Then, given the cells in Fig. 5, where $\mathsf{cn}(\mathbf{d}) = \mathsf{c}_1$ and $\mathsf{cn}(\mathbf{e}) = \mathsf{c}_5$, we have $\mathsf{c}_5 \preceq \mathsf{c}_1$.

The knowledge represented in each cell is expressed in a DL language $\mathcal{L}_\Sigma$ called *object language* which in turn is based on a DL object vocabulary $\Sigma = \mathrm{NC}_\Sigma \uplus \mathrm{NR}_\Sigma \uplus \mathrm{NI}_\Sigma$. Note that the expressivity of languages at the meta and at the object level may be different. In our examples, however, we assume that meta and object level employ the same logic.

3.2.2. Extending the CKR Framework

We now define a KG-OLAP cube as a special kind of CKR with hierarchically-ordered dimensions and cells as well as knowledge propagation from higher to lower-level cells. In the following, in order to formalize the semantics of a KG-OLAP cube, we employ the OLAP cube vocabulary Ω as a CKR meta-knowledge vocabulary and extend the definitions of the CKR core. In particular, we extend the definitions that express propagation of knowledge along the coverage relation and we allow contexts with empty knowledge contents [36]. We provide only basic definitions of the CKR core and refer to previous work [15, 36] for an exhaustive presentation of the CKR framework.

A CKR is a two-layered structure composed of (1) the *global context* $\mathfrak{G}$, consisting of a knowledge base that contains *meta-knowledge*, i.e. the structure and properties of contexts, and *global (context-independent) knowledge*, i.e., knowledge that applies to every context; (2) a set of *(local) contexts* that contain locally valid knowledge.

The meta-knowledge of a CKR is expressed in a DL language containing the elements that define the contextual structure. A *meta-vocabulary* Γ is a DL vocabulary that consists of a set of *context names* $\mathbf{N} \subseteq \mathrm{NI}_\Gamma$, a set of *module names* $\mathbf{M} \subseteq \mathrm{NI}_\Gamma$, a set of *context classes* $\mathbf{C} \subseteq \mathrm{NC}_\Gamma$, including the classes Ctx and Null, a set of *contextual relations* $\mathbf{R} \subseteq \mathrm{NR}_\Gamma$, a set of *contextual attributes* $\mathbf{A} \subseteq \mathrm{NR}_\Gamma$, and for every attribute $\mathsf{A} \in \mathbf{A}$, a set $\mathsf{D}_\mathsf{A} \subseteq \mathrm{NI}_\Gamma$ of *attribute values* of A. The role mod defined over $\mathbf{N} \times \mathbf{M}$ expresses associations between contexts and modules. Intuitively, modules represent pieces of knowledge specific to a context or context class; attributes describe contextual properties (e.g., time, location, provenance) identifying a context (or class). The context class Ctx defines the class of all contexts, while the Null class defines the contexts with empty knowledge modules, the latter being useful for deliberately ruling out inapplicable combinations of dimensions known to lack relevant knowledge content. It is then easy to relate the KG-OLAP cube language (Sec. 3.2) to the CKR core languages: we have that $\mathbf{F} \subseteq \mathbf{N}$ (i.e. cells are a kind of context), $\mathbf{D} \subseteq \mathbf{A}$ (i.e. dimensions are a kind of contextual attributes) and context coverage is a partial order relation in $\mathbf{R}$.

The *meta-language* $\mathcal{L}_\Gamma$ of a CKR is a then DL language over Γ. The knowledge inside contexts of a CKR is expressed via a DL *object language* $\mathcal{L}_\Sigma$ over object vocabulary Σ. The expressions of the object language are evaluated locally to each context, i.e., contexts can interpret each symbol independently. The local evaluation corresponds to the local knowledge of each cell in the KG-OLAP cube. Based on the meta- and object languages, a *Contextualized Knowledge Repository (CKR)* is defined (cf. [36]) as follows.

Definition 1 (Contextualized Knowledge Repository)**.** *A* Contextualized Knowledge Repository (CKR) *over a meta-vocabulary* Γ *and an object vocabulary* Σ *is a structure* $\mathfrak{K} = \langle \mathfrak{G}, \mathrm{K}_\mathbf{M} \rangle$ *where:*

– $\mathfrak{G}$ *is a DL knowledge base over* $\mathcal{L}_\Gamma \cup \mathcal{L}_\Sigma$*, and*
– $\mathrm{K}_\mathbf{M} = \{\mathrm{K}_\mathsf{m}\}_{\mathsf{m} \in \mathbf{M}}$ *where every* K_m *is a DL knowledge base over* $\mathcal{L}_\Sigma$*, for each module name* $\mathsf{m} \in \mathbf{M}$.

In particular, in the following we call $\mathfrak{K}$ a *(KG-OLAP) cube* if its metaknowledge is based (following the above relations) on a cube language $\mathcal{L}_\Omega$.

The CKR semantics basically follows the two-layered structure of the CKR framework: A CKR interpretation is composed by a DL interpretation for the global context and a DL interpretation for every context.

Definition 2 (CKR interpretation)**.** *A* CKR interpretation *for* $\langle \Gamma, \Sigma \rangle$ *is a structure* $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ *s.t.:*

(i). $\mathcal{M}$ *is a DL interpretation of* $\Gamma \cup \Sigma$ *s.t., for every* $\mathsf{c} \in \mathbf{N}$, $\mathsf{c}^\mathcal{M} \in \mathsf{Ctx}^\mathcal{M}$ *and, for every* $\mathsf{C} \in \mathbf{C}$, $\mathsf{C}^\mathcal{M} \subseteq \mathsf{Ctx}^\mathcal{M}$*;*
(ii). for every $x \in \mathsf{Ctx}^\mathcal{M}$, $\mathcal{I}(x)$ *is a DL interpretation over* Σ *s.t.* $\Delta^{\mathcal{I}(x)} = \Delta^\mathcal{M}$ *and, for* $a \in \mathrm{NI}_\Sigma$, $a^{\mathcal{I}(x)} = a^\mathcal{M}$.

The interpretation of ordinary DL expressions in $\mathcal{M}$ and each $\mathcal{I}(x)$ is defined as in the CKR core [39]. We then extend as follows the original definition of CKR model [36] with new conditions for the intended interpretation of the multidimensional structure.

Definition 3 (KG-OLAP cube model)**.** *A CKR interpretation* $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ *is a* KG-OLAP cube model *of* $\mathfrak{K}$ *iff the following conditions hold:*

(i). for $\alpha \in \mathcal{L}_\Sigma \cup \mathcal{L}_\Gamma$ *in* $\mathfrak{G}$, $\mathcal{M} \models \alpha$*;*
(ii). for $\langle x, y \rangle \in \mathsf{mod}^\mathcal{M}$ *with* $y = \mathsf{m}^\mathcal{M}$ *and* $x \notin \mathsf{Null}^\mathcal{M}$, $\mathcal{I}(x) \models \mathrm{K}_\mathsf{m}$*;*
(iii). for $\alpha \in \mathfrak{G} \cap \mathcal{L}_\Sigma$ *and* $x \in \mathsf{Ctx}^\mathcal{M} \setminus \mathsf{Null}^\mathcal{M}$, $\mathcal{I}(x) \models \alpha$.
(iv). if $\mathsf{c}_1, \mathsf{c}_2 \in \mathbf{F}$*, and for every* $\mathsf{A} \in \mathbf{D}$, $\mathcal{M} \models \mathsf{A}(\mathsf{c}_1, d)$ *and* $\mathcal{M} \models \mathsf{A}(\mathsf{c}_2, d)$ *then* $\mathsf{c}_1 = \mathsf{c}_2$.
(v). for $\mathbf{d} \in \mathfrak{D}_\Omega$ *and* $\mathsf{cn}(\mathbf{d}) = \mathsf{c} \in \mathbf{F}$*, then* $\mathcal{M} \models \mathsf{A}(\mathsf{c}, d_\mathsf{A})$ *for each* $\mathsf{A} \in \mathbf{D}$.
(vi). if $\mathsf{c}_1, \mathsf{c}_2 \in \mathbf{F}$*, if* $\mathcal{M} \models \mathsf{c}_1 \preceq \mathsf{c}_2$ *and* $\mathcal{M} \models \mathsf{mod}(\mathsf{c}_2, \mathsf{m})$*, then* $\mathcal{M} \models \mathsf{mod}(\mathsf{c}_1, \mathsf{m})$.

Intuitively, while the conditions (i) and (ii) of Definition 3 impose that $\mathfrak{I}$ verifies the contents of global and local modules associated to contexts, condition (iii) states that global knowledge has to be propagated to local contexts. Note that the contexts in the Null class have no local knowledge associated to them. Condition (iv) states that contexts are identified by the values of their dimension attribute values. Condition (v) basically states that dimensional vectors are a compact way to represent assertions of the kind $A(\mathsf{c}, d_A)$ in the meta-knowledge. Finally, Condition (vi) defines the propagation of modules associated with more general contexts to the covered contexts.

Given a CKR $\mathfrak{K}$ over $\langle \Gamma, \Sigma \rangle$ and $\mathsf{c} \in \mathbf{N}$, an axiom $\alpha \in \mathcal{L}_\Sigma$ is c-*entailed* by $\mathfrak{K}$ (denoted $\mathfrak{K} \models \mathsf{c} : \alpha$) if $\mathcal{I}(\mathsf{c}^{\mathcal{M}}) \models \alpha$ for every model $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ of $\mathfrak{K}$. We say that an axiom α is *globally entailed* by $\mathfrak{K}$ (denoted $\mathfrak{K} \models \alpha$) if: (i) $\alpha \in \mathcal{L}_\Sigma$ and $\mathfrak{K} \models \mathsf{c} : \alpha$ for every $\mathsf{c} \in \mathbf{N}$, or (ii) $\alpha \in \mathcal{L}_\Gamma$ and $\mathcal{M} \models \alpha$ for every cube model $\mathfrak{I} = \langle \mathcal{M}, \mathcal{I} \rangle$ of $\mathfrak{K}$.

3.2.3. Reasoning in OWL-RL Cubes

As in the original formulation of CKR in [36], we can formalize instance-level reasoning inside a KG-OLAP cube using a *materialization calculus* [40]. As in [36], the calculus is provided for cubes using a specific DL language[2], that we call $\mathcal{SROIQ}$-RL, which corresponds to the OWL-RL fragment [41]. The definition of the calculus and of the $\mathcal{SROIQ}$-RL DL language are provided in the Appendix.

Intuitively, the materialization calculus is based on a translation to Datalog. The axioms of the input cube $\mathfrak{K}$ are translated into Datalog atoms (by *input rules I*), and Datalog rules (called *deduction rules P*) are added to the translation in order to encode the global and local inference rules. Instance checking is then performed by translating (by *output rules O*) the ABox assertion to be verified into a Datalog fact and verifying whether this fact is entailed by the CKR program $PK(\mathfrak{K})$.

With respect to the calculus for OWL-RL CKRs presented in [36], it is necessary to introduce additional rules and translation steps in order to express the computation of the coverage relation and the propagation of object knowledge. In particular, regarding the translation rules, we need to introduce global input rules that encode the coverage in the level and dimensional hierarchies: the following global deduction rule then provides the propagation of modules (corresponding to condition (vi) in the definition of KG-OLAP model), where gm denotes the context name of global meta-knowledge:

$$\texttt{triple}(\mathsf{c}_1, \mathsf{covers}, \mathsf{c}_2, \mathsf{gm}), \texttt{triple}(\mathsf{c}_1, \mathsf{mod}, m, \mathsf{gm}) \\ \rightarrow \texttt{triple}(\mathsf{c}_2, \mathsf{mod}, m, \mathsf{gm})$$

Then, the translation procedure is extended from the one presented for CKR [36] by introducing new steps in which the cell coverage relation is computed from the dimensional coverage in the global program.

We can show (see the Appendix, extending the results in [36]) that the presented rules and translation process provide a sound and complete calculus for instance checking for $\mathcal{SROIQ}$-RL KG-OLAP cubes.

Theorem 1. Given $\mathfrak{K} = \langle \mathfrak{G}, \mathrm{K}_\mathbf{M} \rangle$ a consistent KG-OLAP cube in $\mathcal{SROIQ}$-RL normal form, $\alpha \in \mathcal{L}_\Sigma$ an atomic concept or role assertion and $\mathsf{c} \in \mathbf{F}$ s.t. $O(\alpha, \mathsf{c})$ is defined, then $PK(\mathfrak{K}) \models O(\alpha, \mathsf{c})$ iff $\mathfrak{K} \models \mathsf{c} : \alpha$. $\square$

4. Query Operations

In this section, we introduce a set of query operations for working with KG-OLAP cubes. We distinguish between *contextual* and *graph* operations. Contextual operations alter the multidimensional structure of a cube. Graph operations modify the RDF graph in the knowledge modules of the cells. Formally, the operations are defined as transformations of KG-OLAP cubes.

4.1. Contextual Operations

The contextual operations select and combine cells of a KG-OLAP cube using its dimensions and levels. The *slice-and-dice* operation allows for the selection of a set of facts whereas the *merge* operation combines cells at finer granularities into aggregated cells at a coarser granularity, merging the contents of the modules from the finer-grained cells.

4.1.1. Slice and Dice

The *slice-and-dice* operation restricts a cube to a set of cells with a specific subset of dimension attribute values; the operation selects a subcube of an input KG-OLAP cube. The slice-and-dice operation selects a partition of the cube for subsequent manipulation. Note that slice-and-dice operations in data warehousing literature and practice come in various fashions. The definition in this section establishes a basic notion of slice-and-dice for KG-OLAP cubes. Future work may well extend this notion to provide rich query mechanisms in order to filter contexts based on complex conditions in an expressive domain ontology.

Definition 4 (Slice and dice). *Given a cube $\mathfrak{K} = \langle \mathfrak{G}, \mathrm{K}_\mathbf{M} \rangle$ and a dimensional vector $\overline{\mathbf{d}}$ which defines the dice coordinates, we define the* slice-and-dice oper-

[2]Note that, otherwise, definitions for KG-OLAP cube and CKR are independent of the DL language used at the meta and object levels.

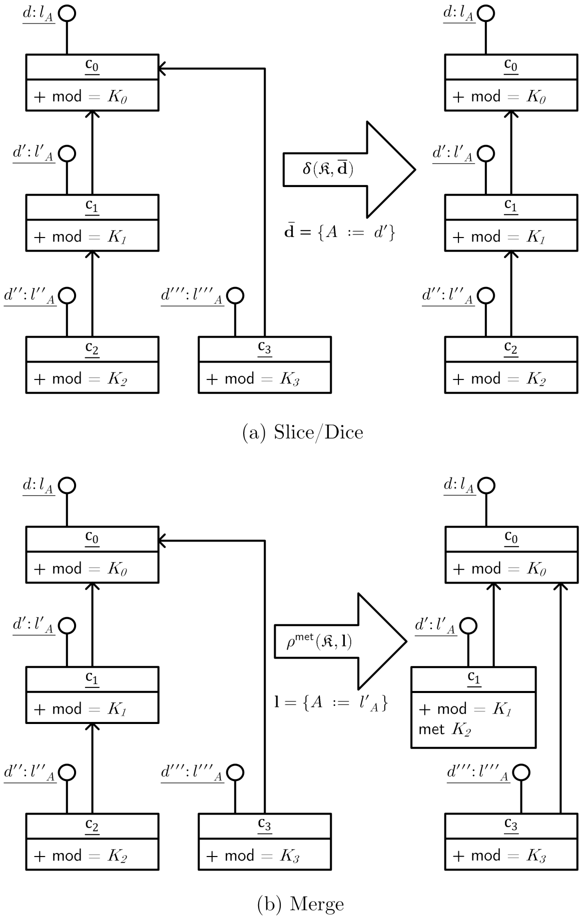


Fig. 7. Illustration of contextual operations definitions

ation $\delta(\mathfrak{K}, \overline{\mathbf{d}})$ *of* $\mathfrak{K}$ *with respect to* $\overline{\mathbf{d}}$ *as a new cube* $\mathfrak{K}' = \langle \mathfrak{G}', \mathrm{K}_{\mathbf{M}'} \rangle$ *over* $\langle \Gamma', \Sigma \rangle$*, such that*[3]*:*

– $\mathbf{M}' = \mathbf{M}$, $\mathbf{D}' = \mathbf{D}$, *and for each* $\mathsf{A} \in \mathbf{D}$, $\mathsf{L}'_\mathsf{A} = \mathsf{L}_\mathsf{A}$*;*
– *For each* $\mathsf{A} \in \mathbf{D}$, $\mathsf{D}'_\mathsf{A} = \{\mathsf{d}'_\mathsf{A} \in \mathsf{D}_\mathsf{A} \mid \mathsf{d}'_\mathsf{A} \preceq \mathsf{d}_\mathsf{A}$ *or* $\mathsf{d}_\mathsf{A} \preceq \mathsf{d}'_\mathsf{A}$, *with* $\mathsf{d}_A \in \overline{\mathbf{d}}\}$*;*
– $\mathbf{F}' = \{\mathsf{c} \in \mathbf{F} \mid$ *for each* $\mathsf{d}_\mathsf{A} \in \mathsf{cn}^-(\mathsf{c}), \mathsf{d}_\mathsf{A} \in \mathsf{D}'_\mathsf{A}\}$*;*
– $\mathfrak{G}' = \mathfrak{G}_\Sigma \cup \mathfrak{G}_{\Gamma'}$ *(i.e., metaknowledge in* $\mathfrak{G}'$ *is equal to the formulas in* $\mathfrak{G}_\Gamma$ *that have only symbols in* Γ'*).*

Figure 7a depicts the definition of slice-and-dice operation on a one-dimensional cube. Intuitively, the slice-and-dice operation takes as argument the coordinates of a point in the cube, i.e., a dimensional vector $\overline{\mathbf{d}}$, and produces a new cube by extracting all cells, along with their associated knowledge modules, at points underneath the argument point as well as the cells that are in a coverage relationship with those cells at points underneath the argument point.

Example 6 (Slice and dice)**.** Figure 8 illustrates the application of the slice-and-dice operation on the KG-OLAP cube from Fig. 5, which we denote by $\mathfrak{K}_{\mathsf{ATM}}$. The context shown as shaded box represents the dice coordinates $\{\mathsf{Importance} := \mathsf{Essential}, \mathsf{Location} := \mathsf{LOWW}, \mathsf{Date} := \mathsf{All\text{-}date}, \mathsf{Aircraft} := \mathsf{FixedWing}\}$. Only cells that are underneath the point identified by the dice coordinates, i.e., $\mathsf{c}_4, \mathsf{c}_5$, and c_6, or cells that are in a coverage relationship with $\mathsf{c}_4, \mathsf{c}_5$, and c_6, i.e., $\mathsf{c}_0, \mathsf{c}_1$, and c_3 are kept in the result cube $\mathfrak{K}'_{\mathsf{ATM}}$; the disregarded cells are shown in gray color. ◇

4.1.2. Merge

The *merge* (or *contextual roll-up*) operation changes the granularity of a cube and its dimensions. Given an argument granularity specified as a vector of dimension levels $\mathbf{l}$, the merge operation combines the contents of knowledge modules at granularities that are more specific than the given granularity.

Formally, we define a *level vector* as a set: $\mathbf{l} = \{l_1, \ldots, l_k\}$ s.t. for $\mathsf{j} \in \{1, \ldots, k\}$, $l_\mathsf{j} \in \mathsf{L}_{\mathsf{A}_\mathsf{j}}$. We define restrictions of dimensional space $\mathfrak{D}_\Omega$ given w.r.t. a level vector $\mathbf{l}$ as follows:

$$\mathfrak{D}^{\mathbf{l}}_{\Omega} = \{\mathbf{d} \in \mathfrak{D}_\Omega \mid \text{for } \mathsf{d} \in \mathsf{D}_\mathsf{A}, \mathsf{lev}(\mathsf{d}, l) \text{ with } l \in \mathbf{l}\}$$
$$\mathfrak{D}^{\mathbf{l}\succeq}_{\Omega} = \{\mathbf{d} \in \mathfrak{D}_\Omega \mid \mathbf{e} \preceq \mathbf{d}, \text{with } \mathbf{e} \in \mathfrak{D}^{\mathbf{l}}_{\Omega}\}$$

Intuitively, the subspace $\mathfrak{D}^{\mathbf{l}}_{\Omega}$ identifies all the vectors *exactly* at the level specified by the level vector $\mathbf{l}$, while $\mathfrak{D}^{\mathbf{l}\succeq}_{\Omega}$ defines the vectors *above* (or equal to) the specified level vector.

Let $\mu(\mathsf{c}) = \bigcup_{\mathsf{c}' \prec \mathsf{c}} \{\mathsf{m} \in \mathbf{M} \mid \mathfrak{G} \models \mathsf{mod}(\mathsf{c}', \mathsf{m})\}$. The set $\mu(\mathsf{c})$ then contains all module names of the initial cube associated to contexts c' that are more specific than the input context c (with respect to the coverage relation).

Definition 5 (Merge)**.** *Given a cube* $\mathfrak{K} = \langle \mathfrak{G}, \mathrm{K}_\mathbf{M} \rangle$ *and a level vector* $\mathbf{l}$*, we define the* merge operation $\rho^{\mathsf{met}}(\mathfrak{K}, \mathbf{l})$ *of* $\mathfrak{K}$ *with respect to the level vector* $\mathbf{l}$ *as a new cube* $\mathfrak{K}' = \langle \mathfrak{G}', \mathrm{K}_{\mathbf{M}'} \rangle$ *over* $\langle \Gamma', \Sigma' \rangle$ *s.t.*

– $\mathbf{F}' = \{\mathsf{c} \in \mathbf{F} \mid \mathsf{cn}^-(\mathsf{c}) \in \mathfrak{D}^{\mathbf{l}\succeq}_{\Omega}\}$*;*
– $\mathbf{D}' = \mathbf{D}$*;*
– $\mathbf{M}' = \mathbf{M} \cup \{\mathsf{mg}(\mathsf{c}) \mid \mathsf{c} \in \mathbf{F}'$ *with* $\mathsf{cn}(\mathsf{c})^- \in \mathfrak{D}^{\mathbf{l}}_{\Omega}\}$ *with each* $\mathsf{mg}(\mathsf{c})$ *a new module name;*
– *For each* $A \in \mathbf{D}$, $\mathsf{L}'_\mathsf{A} = \{\mathsf{l}'_A \in \mathsf{L}_\mathsf{A} \mid \mathsf{l}_A \preceq \mathsf{l}'_A, \mathsf{l}_A \in \mathbf{l}\}$*;*
– *For each* $A \in \mathbf{D}$, $\mathsf{D}'_\mathsf{A} = \{\mathsf{d}'_A \in \mathsf{D}_\mathsf{A} \mid \mathsf{lev}(\mathsf{d}'_A, \mathsf{l}_A), \mathsf{l}_A \in \mathsf{L}'_\mathsf{A}\}$*;*

[3]In the definition of operations, for simplicity of notation, we assume that components of the cube $\mathfrak{K}'$ and languages Γ', Σ' are recognized with a prime superscript, e.g., Γ' contains $\mathbf{M}', \mathbf{F}', \mathbf{D}'$, etc.

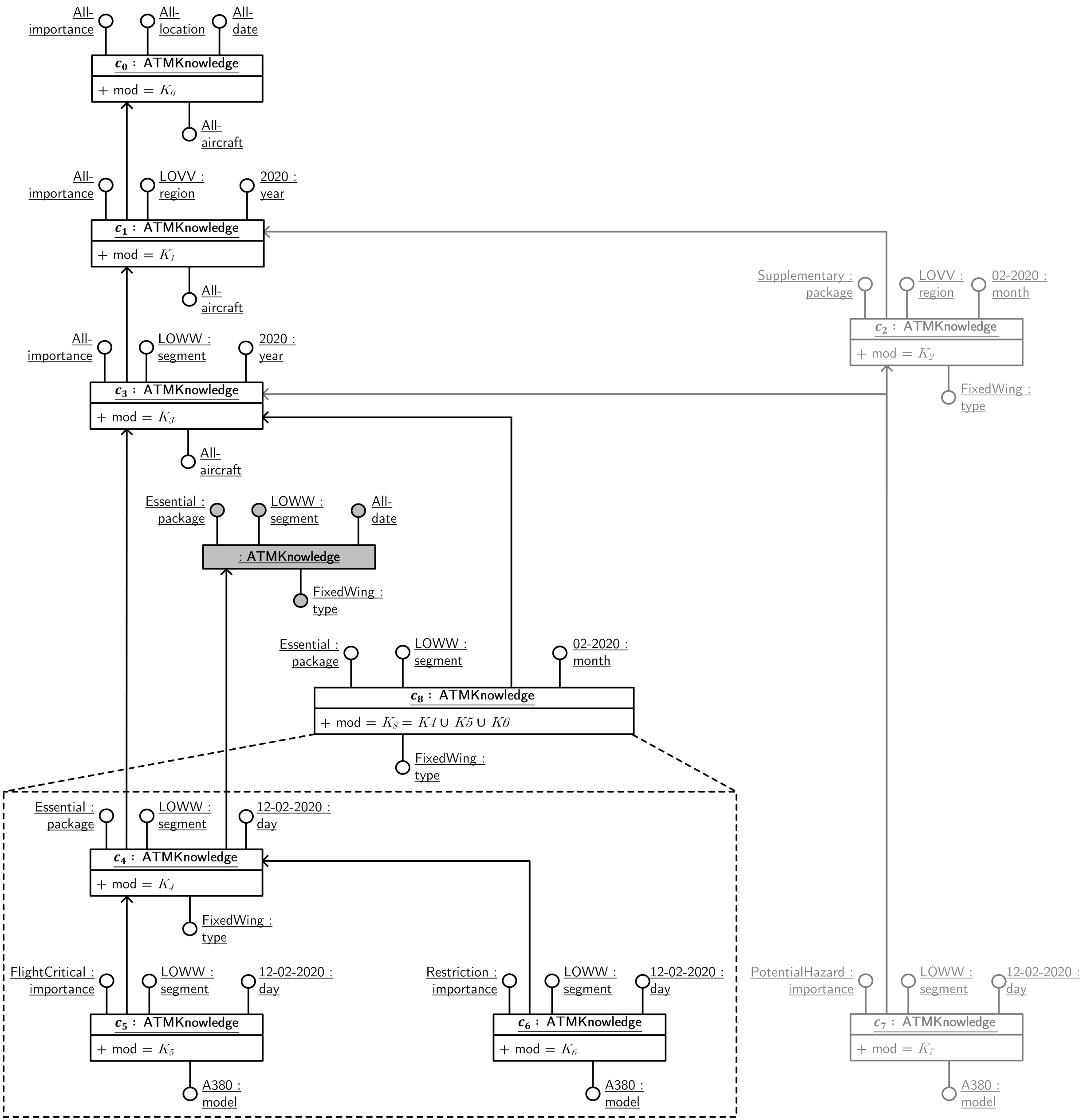


Fig. 8. Applying slice-and-dice and merge operations on the KG-OLAP cube instance from Fig. 5. Gray lines denote contexts that are disregarded by the slice-and-dice operation $\delta(\mathfrak{K}_{\mathsf{ATM}}, \{\mathsf{Importance} := \mathsf{Essential}, \mathsf{Location} := \mathsf{LOWW}, \mathsf{Date} := \mathsf{All\text{-}date}, \mathsf{Aircraft} := \mathsf{FixedWing}\})$, with the unnamed context shown as shaded box denoting the dice coordinates, and the dashed box denotes a merge of contexts $\rho^{\cup}(\mathfrak{K}'_{\mathsf{ATM}}, \{\mathsf{package}, \mathsf{segment}, \mathsf{month}, \mathsf{type}\})$ into the c_8 context.

– $\mathfrak{G}' = \mathfrak{G}_\Sigma \cup \mathfrak{G}_{\Gamma'} \cup \{\mathsf{mod}(\mathsf{c}, \mathsf{mg}(\mathsf{c})) \mid \mathsf{c} \in \mathbf{F}'$ *with* $\mathsf{cn}(\mathsf{c})^- \in \mathfrak{D}^{\mathbf{l}}_\Omega\}$*;*
– ***Union merge*** ($\mathsf{met} = \cup$)*: knowledge module* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *for* c *is added to* $\mathrm{K}_{\mathbf{M}'}$ *with:* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})} = \bigcup_{\mathsf{m} \in \mu(\mathsf{c})} \mathrm{K}_\mathsf{m}$
– ***Intersection merge*** ($\mathsf{met} = \cap$)*: knowledge module* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *for* c *is added to* $\mathrm{K}_{\mathbf{M}'}$ *with:* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})} = \bigcap_{\mathsf{m} \in \mu(\mathsf{c})} \mathrm{K}_\mathsf{m}$

Figure 7b shows an illustration of the merge operation definition. Intuitively, the merge operation is a transformation over the original cube that combines the knowledge from lower-level cells into higher-level cells in the contextual hierarchy (by adding a new module $\mathsf{mg}(\mathsf{c})$ containing the merged knowledge) and "cuts" the contexts below the level defined by the input level vector $\mathbf{l}$. The roll-up operation employs a specific combination method $\mathsf{met} \in \{\cup, \cap\}$, which specifies the kind of combination of knowledge inside the merged cells.

Example 7 (Merge)**.** In Fig. 8, the c_8 context is the result of a union merge to the $\{\mathsf{package}, \mathsf{segment}, \mathsf{month}, \mathsf{type}\}$ dice level of the result cube $\mathfrak{K}'_{\mathsf{ATM}}$ from the previous slice-and-dice operation. The c_8 cell is at the dice level, its knowledge module being the union of the knowledge modules from the covered cells. ◇

4.2. Graph Operations

Graph operations – abstraction, pivoting, and reification – alter the structure of the RDF graphs inside the knowledge modules of a cell. Abstraction replaces sets of entities with individual and more abstract entities. Pivoting moves metaknowledge (contextual information) inside the modules. Reification allows to represent relations as individuals.

4.2.1. Abstraction

Abstraction serves as an umbrella term for a class of graph operations that, broadly speaking, replace entities in an RDF graph with more abstract entities. This abstraction is based on various types of ontological information, e.g., class membership and grouping properties. We also refer to abstraction as *ontological roll-up*.

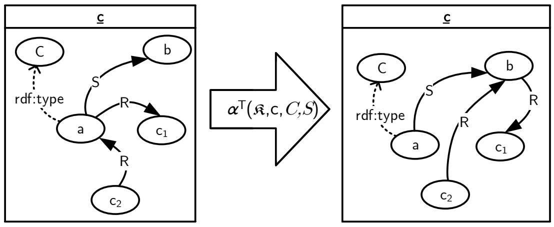
We distinguish three types of abstraction: (a). *triple-generating abstraction* generates new triples from existing triples, where an existing individual acts as abstraction of a set of other resources; (b). *individual-generating abstraction* generates a new individual that acts as abstraction of a set of resources; (c). *value-generating abstraction* computes a new value using some aggregation operation on a set of values.

Consider the set of asserted and inherited modules of a cell c: $\mathsf{mod}(\mathsf{c}) = \{\mathsf{m} \in \mathbf{M} \mid \mathfrak{G} \models \mathsf{mod}(\mathsf{c}, \mathsf{m})\}$. We then denote the local knowledge base of cell c as:

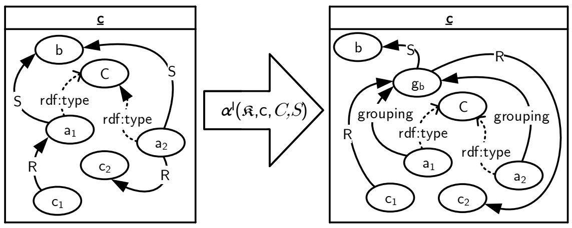
$$\mathrm{K}_{\mathsf{mod}(\mathsf{c})} = \bigcup_{\mathsf{m} \in \mathsf{mod}(\mathsf{c})} \mathrm{K}_\mathsf{m}$$

Definition 6 (Abstraction)**.** *Given a cube* $\mathfrak{K} = \langle \mathfrak{G}, \mathrm{K}_\mathbf{M} \rangle$*, a context name* $\mathsf{c} \in \mathbf{F}$*, a (possibly complex) concept* C *of* $\mathcal{L}_\Sigma$ *restricting abstraction to a subset of individuals, a (possibly complex) role* S *of* $\mathcal{L}_\Sigma$ *– the grouping property – we define the* abstraction operation $\alpha^{\mathsf{met}}(\mathfrak{K}, \mathsf{c}, C, S)$ *as a new cube* $\mathfrak{K}' = \langle \mathfrak{G}', \mathrm{K}'_\mathbf{M} \rangle$ *over* $\langle \Gamma', \Sigma' \rangle$*, with* $\mathsf{met} \in \{T, I, V(op)\}$ *for the specific abstraction method (triple, individual or value generation), where the local knowledge module* $\mathrm{K}_{\mathsf{mod}(\mathsf{c})}$ *is modified as follows, depending on the abstraction method:*

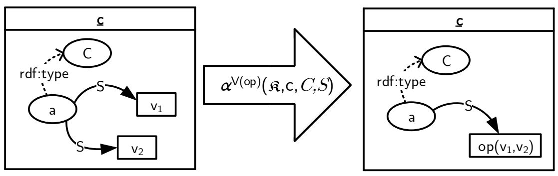
– $\mathbf{M}' = \mathbf{M} \cup \{\mathsf{mg}(\mathsf{c})\} \setminus \overline{\mathsf{mod}}(\mathsf{c})$*, with* $\mathsf{mg}(\mathsf{c})$ *a new module name and* $\overline{\mathsf{mod}}(\mathsf{c})$ *the set of asserted modules of* c *in the original cube;*
– $\mathfrak{G}' = \mathfrak{G} \cup \{\mathsf{mod}(\mathsf{c}, \mathsf{mg}(\mathsf{c}))\} \setminus \{\mathsf{mod}(\mathsf{c}, \mathsf{m}) \mid \mathsf{m} \in \overline{\mathsf{mod}}(\mathsf{c})\}$
– $\mathrm{K}_{\mathsf{mg}(\mathsf{c})} = \mathrm{K}_{\overline{\mathsf{mod}}(\mathsf{c})}$ *and* $\mathrm{K}_{\mathbf{M}'} = \mathrm{K}_\mathbf{M} \cup \{\mathrm{K}_{\mathsf{mg}(\mathsf{c})}\} \setminus \{\mathrm{K}_{\overline{\mathsf{mod}}(\mathsf{c})}\}$
– ***triple generation*** T*: for* $b \in \mathrm{NI}_\Sigma$ *with* $\mathrm{K}_{\mathsf{mod}(\mathsf{c})} \models C(a)$*, let* $S^-(b) = \{a \in \mathrm{NI}_\Sigma \mid \mathrm{K}_{\mathsf{mod}(\mathsf{c})} \models S(a, b)\}$*; then:*
  – *for every role assertion* $R(a, c) \in \mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *with* $a \in S^-(b)$ *and* $R \neq S$*, add* $R(b, c)$ *to* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *and remove* $R(a, c)$ *from* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$*;*
  – *for every role assertion* $R(c, a) \in \mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *with* $a \in S^-(b)$ *and* $R \neq S$*, add* $R(c, b)$ *to* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *and remove* $R(c, a)$ *from* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$*;*
– ***individual generation*** I*: for* $a \in \mathrm{NI}_\Sigma$ *with* $\mathrm{K}_{\mathsf{mod}(\mathsf{c})} \models C(a)$*, let* $S(a) = \{b \in \mathrm{NI}_\Sigma \mid \mathrm{K}_{\mathsf{mod}(\mathsf{c})} \models S(a, b)\}$*; then:*
  – *for every* $b \in S(a)$*, add* $\mathsf{grouping}(a, g_b)$ *to* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *with* $g_b \in \mathrm{NI}_{\Sigma'}$ *a new individual name (associated to the grouping individual* b*).*
  – *for every role assertion* $R(a, c) \in \mathrm{K}_{\mathsf{mg}(\mathsf{c})}$*, for every* $b \in S(a)$*, add* $R(g_b, c)$ *to* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *and remove* $R(a, c)$ *from* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$*;*
  – *resp. for every* $R(c, a)$ *and* $C(a) \in \mathrm{K}_{\mathsf{mg}(\mathsf{c})}$
– ***value generation*** $V(op)$*: for* $a \in \mathrm{NI}_\Sigma$ *with* $\mathrm{K}_{\mathsf{mod}(\mathsf{c})} \models C(a)$*, considering the operation* op *on values in the range of* S*, let* $S(a) = \{v \in \mathrm{NI}_\Sigma \mid S(a, v) \in \mathrm{K}_{\mathsf{mg}(\mathsf{c})}\}$*, then:*
  – *add to* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *the assertion* $S(a, op(v_1, \ldots, v_m))$ *with* $\{v_1, \ldots, v_m\} = S(a)$*;*
  – *remove every* $S(a, v) \in \mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *with* $v \in S(a)$*;*



(a) Triple-generating Abstraction



(b) Individual-generating Abstraction



(c) Value-generating Abstraction

Fig. 9. Illustration of abstraction operations definitions

Note that for simplicity we treat literal values as individuals and we do not distinguish roles across individuals and values in our language. We note that rdf:type may serve as grouping property, provided that (newly introduced) grouping individuals represent the concepts employed for grouping and that the management of these grouping individuals is taken care of (cf. OWL "punning" [42]). Individual-generating abstraction may be extended for multiple grouping properties. Moreover, we note that the grouping role S is allowed to be a complex role expression, thus permitting to group, e.g., along role compositions.

Figure 9 shows a graphical representation of the abstraction operations definition. Intuitively, the abstraction operation takes as input the single cell on the knowledge module of which the operation is applied, the class C of individuals to be abstracted and a property S, which represents the grouping relation along which the elements have to be abstracted. The kind of manipulation on the cell's knowledge then depends on the abstraction type: (a) in triple-generating abstraction, for every instance $C(b)$, if there is some relation of the kind $S(a, b)$ (i.e. a is grouped by b), then all of the role assertions of the kind $R(a, c)$ or $R(c, a)$ are redirected to the grouping individual b; (b) in individual-generating abstraction, for every instance $C(a)$, if there is some relation of the kind $S(a, b)$[4] then a new grouping individual g_b and assertion $\mathsf{grouping}(a, g_b)$ are added and, as above, all of the ABox assertions of the kind $R(a, c), R(c, a)$ and $A(a)$ are redirected to g_b; (c) in value-generating abstraction, for every element $C(a)$, we consider all of the values $v_1, \ldots, v_m$ that are related to a by role S and we add their aggregation $op(v_1, \ldots, v_m)$ by a parameter operator op as a new S value for a.

In the following, we illustrate the different variants of abstraction using examples from the ATM domain. We start with an example combining triple-generating, individual-generating, and value-generating abstraction before separately looking at triple-generating and individual-generating abstraction in more detail.

Example 8 (Abstraction)**.** Figure 10 illustrates the different variants of abstraction on the running example of ATM knowledge graphs. The RDF graph on the bottom shows (part of) the triples of the K_4 module of the c_4 context of $\mathfrak{K}_{\text{ATM}}$ from Fig. 6 indicating runway and taxiway contaminations. A triple-generating abstraction $\alpha^T(\mathfrak{K}_{\text{ATM}}, c_4, \top, \mathsf{grouping})$ leads to the replacement of individuals dry_snow and compact_snow with snow, using grouping as grouping property. On the result of that abstraction, $\mathfrak{K}'_{\text{ATM}}$, an individual-generating abstraction $\alpha^I(\mathfrak{K}'_{\text{ATM}}, c_4, \mathsf{RunwayTaxiway}, \mathsf{contaminationType})$ groups all RunwayTaxiway individuals with the same contaminationType property value; the grouping property indicates which individuals have been grouped. The new individual assumes the place of the grouped individuals in the graph. Another individual-generating abstraction $\alpha^I(\mathfrak{K}''_{\text{ATM}}, c_4, \mathsf{SurfaceContamination}, \mathsf{layer})$ then groups all SurfaceContamination individuals with the same layer target. A value-generating abstraction $\alpha^{V(avg)}(\mathfrak{K}'''_{\text{ATM}}, c_4, \mathsf{SurfaceContamination}, \mathsf{depth})$, in turn, replaces multiple depth property values (0.2 and 0.4) by the average depth (0.3). The result graph thus indicates, at an abstract level, the presence of snow contamination at runways and taxiways along with the average depth. ◇

[4] Note that the definition can be easily extended for multiple grouping properties. In this case, every individual a with $S_1(a, b)$ and $S_2(a, c)$, for example, could be replaced by a grouping g_{bc}, which allows for more complex group-by operations. See the appendix for a SPARQL-based implementation that allows for multiple grouping properties to be specified.

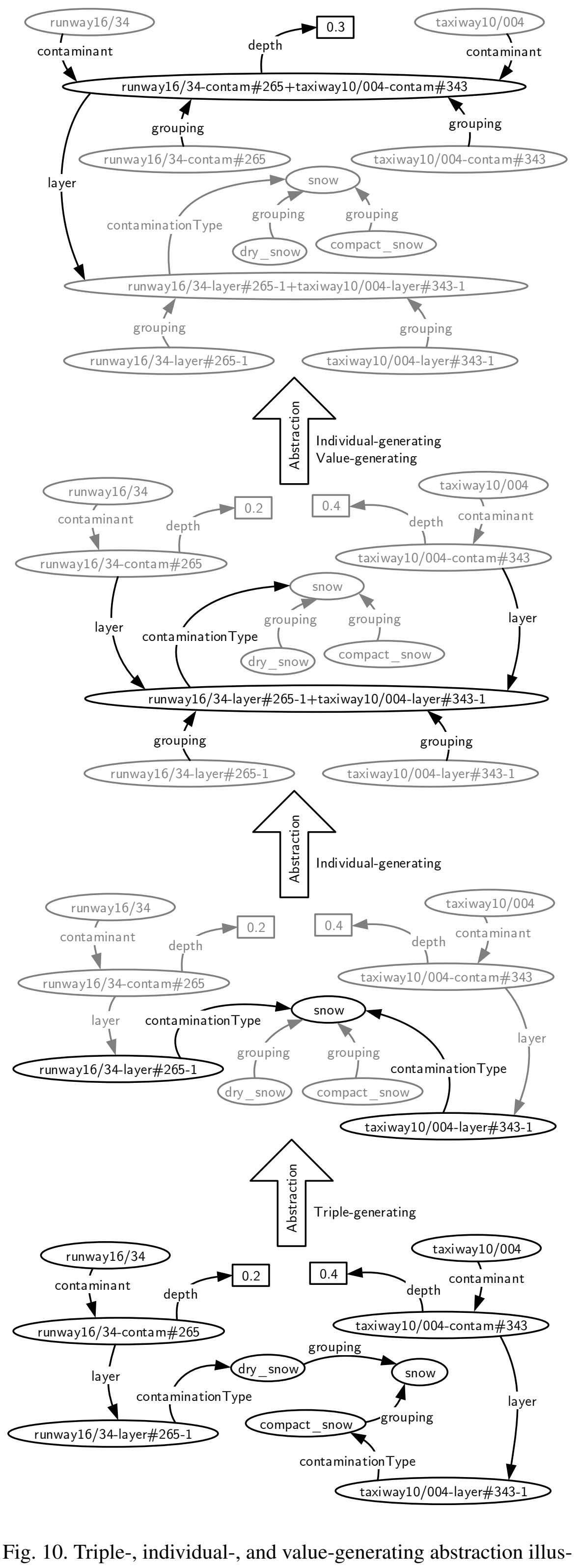


Fig. 10. Triple-, individual-, and value-generating abstraction illustrated on the knowledge module K_4 of the c_4 context (see Fig. 6)

Example 9 (Triple-generating abstraction)**.** Figure 11 shows two other applications of triple-generating abstraction. The example abstraction basically operates on the merge of knowledge modules K_5 and K_6 from Fig. 6 – extended with a grouping property to heavyWeight for characteristic#556 and characteristic#677. Assume that the grouping property was added based on the classification of characteristic#556 and characteristic#677 as HeavyWeight prior to the abstraction, according to the definition of that concept in the K_1 module. Let $\mathfrak{K}'_{\text{ATM}}$ denote the KG-OLAP cube from Fig. 8 after applying the merge, then the lower part of Fig. 11 shows the contents of K_8 (omitting the facts from K_4) of the c_8 cell generated by a previous merge. The triple-generating abstraction $\mathfrak{K}''_{\text{ATM}} := \alpha^T(\mathfrak{K}'_{\text{ATM}}, c_8, \text{AircraftCharacteristic}, \text{grouping})$ then replaces the aircraft characteristics characteristic#556 and characteristic#677 by their grouping heavyWeight. In the result of that operation, the triple-generating abstraction $\alpha^T(\mathfrak{K}''_{\text{ATM}}, c_8, \text{RunwayTaxiway}, \text{isSituatedAt})$ replaces runway16/34 and taxiway10/004 by airport-LOWW. Thus, the graph after execution of those operations (upper part of Fig. 11) shows restricted availabilities for heavy-weight aircraft in Vienna. ◇

Example 10 (Individual-generating abstraction)**.** Figure 12 shows two other applications of individual-generating abstraction. Let $\mathfrak{K}_{\text{ATM}}$ denote the KG-OLAP cube from Fig. 11 after applying the triple-generating abstraction, the individual-generating abstraction $\alpha^I(\mathfrak{K}_{\text{ATM}}, c_8, \text{ManoeuvringAreaUsage}, \text{aircraft})$ groups all individuals of ManoeuvringAreaUsage that refer to the same aircraft, effectively replacing runway16/34-usage#241-1 and taxiway10/004-usage#352-1 by the individual runway16/34-usage#241-1+taxiway10/004-usage#352-1; $\mathfrak{K}'_{\text{ATM}}$ denotes the result cube of that abstraction. The second individual-generating abstraction $\alpha^I(\mathfrak{K}'_{\text{ATM}}, c_8, \text{ManoeuvringAreaAvailability}, \text{usage})$ then groups all individuals of ManoeuvringAreaAvailability that refer to the same ManoeuvringAreaUsage, effectively replacing runway16/34-avail#241 and taxiway10/004-avail#352 by the individual runway16/34-avail#241+taxiway10/004-usage#352. Thus, the graph after execution of those operations (upper part of Fig. 12) is a compact representation of availabilities and usage restrictions in Vienna. ◇

4.2.2. Pivoting

The *pivoting* operation attaches dimensional properties (dimension attribute values) of a cell to a specified set of individuals inside the cell's object knowl-

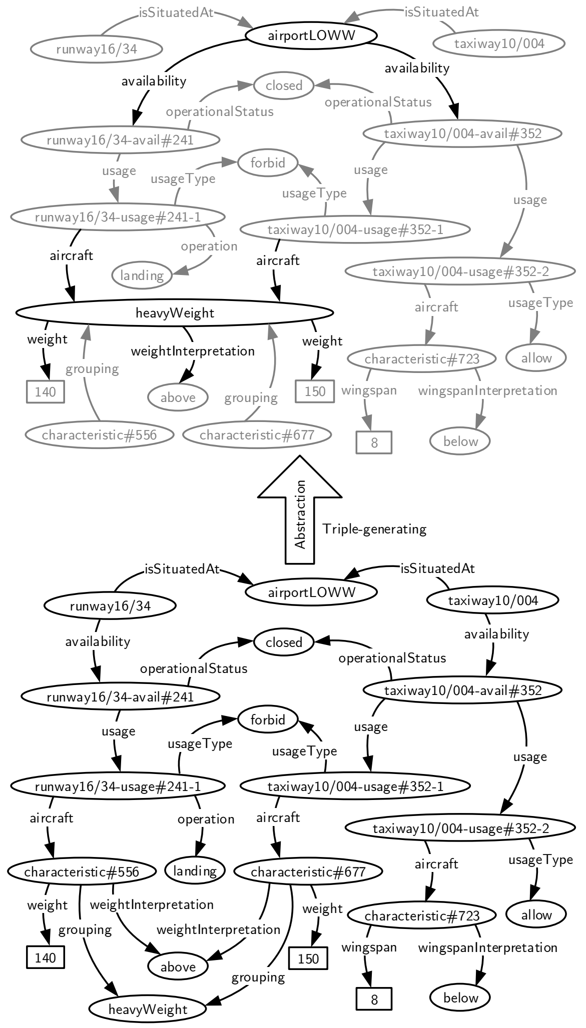


Fig. 11. Triple-generating abstraction illustrated on the merge of knowledge from the knowledge modules K_5 and K_6 (see Fig. 6)

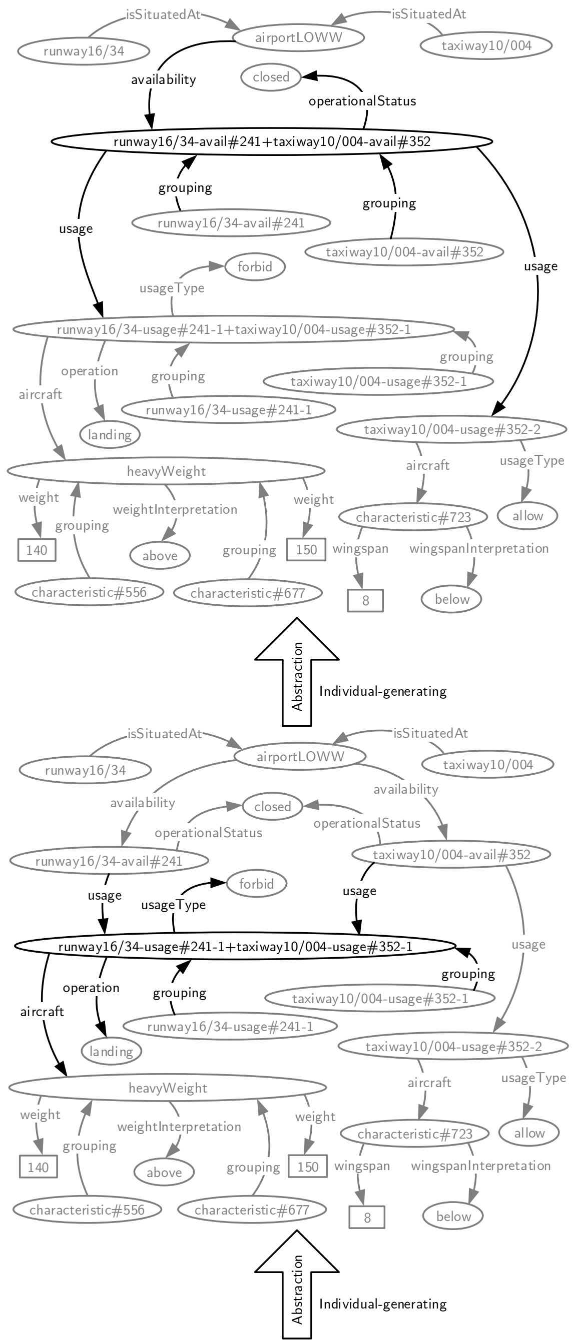


Fig. 12. Individual-generating abstractions starting from the result of the triple-generating abstraction from Fig. 11

edge. Pivoting allows for the preservation of contextual knowledge in case of a merge operation.

Definition 7 (Pivoting). *Given a cube* $\mathfrak{K} = \langle \mathfrak{G}, \mathrm{K}_{\mathbf{M}} \rangle$*, a cell name* $\mathsf{c} \in \mathbf{F}$*, a (possibly complex) concept* C *of* $\mathcal{L}_{\Sigma}$ *of the objects to be labeled, and a set* $D = \{\mathsf{A}_1, \ldots, \mathsf{A}_n\} \subseteq \mathbf{D}$ *of the selected set of dimension labels, we define the* pivoting operation $\pi(\mathfrak{K}, \mathsf{c}, C, D)$ *as a new cube* $\mathfrak{K}' = \langle \mathfrak{G}', \mathrm{K}_{\mathbf{M}'} \rangle$ *over* $\langle \Gamma', \Sigma' \rangle$ *s.t.*

- $\mathbf{M}' = \mathbf{M} \cup \{\mathsf{mg}(\mathsf{c})\}$*, with* $\mathsf{mg}(\mathsf{c})$ *a new module name;*
- $\mathfrak{G}' = \mathfrak{G} \cup \{\mathsf{mod}(\mathsf{c}, \mathsf{mg}(\mathsf{c}))\}$
- $\mathrm{K}_{\mathbf{M}'} = \mathrm{K}_{\mathbf{M}} \cup \{\mathrm{K}_{\mathsf{mg}(\mathsf{c})}\}$
- *for every* $e \in \mathrm{NI}_{\Sigma}$ *with* $\mathrm{K}_{\mathsf{mod}(\mathsf{c})} \models C(e)$*, we add to* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *the set of assertions* $\mathsf{A}_1(e, d_{\mathsf{A}_1}), \ldots, \mathsf{A}_n(e, d_{\mathsf{A}_n})$ *if* $\mathfrak{G} \models \mathsf{A}_1(\mathsf{c}, d_{\mathsf{A}_1}), \ldots, \mathsf{A}_n(\mathsf{c}, d_{\mathsf{A}_n})$.

Note that we have to admit that $\Sigma \cap \Gamma \neq \emptyset$ in order to use metaknowledge symbols in the local object knowledge. Figure 13 shows an illustration of the pivoting operation definition. Intuitively, the pivoting operation takes as input a cell c and as parameters a class C as

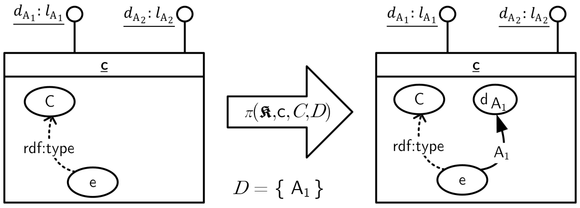


Fig. 13. Illustration of the pivoting operation definition

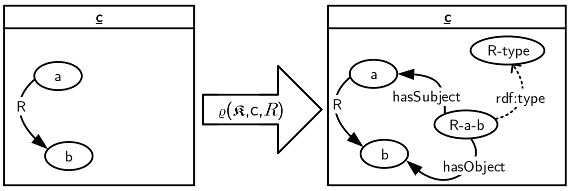


Fig. 14. Illustration of the reification operation definition

well as a set of dimensions D. The operation associates with c an additional knowledge module $\mathsf{mg}(\mathsf{c})$ that contains, for each element e of argument class C in c, a set of assertions that label the element e with the dimension attribute values of c associated with the argument dimensions in $D \subseteq \mathbf{D}$.

Example 11 (Pivoting)**.** Consider the K_4 knowledge module of the c_4 context from Fig. 6 (see Fig. 10 for an illustration of the cell contents). The pivoting operation $\pi(\mathfrak{K}, c_4, \mathsf{SurfaceContamination}, \{\mathsf{Importance}, \mathsf{Date}\})$ then returns a new cube $\mathfrak{K}'$ with a knowledge module $\mathsf{mg}(\mathsf{c}_4)$ that contains the additional assertions Importance(runway16/34-contam#265, Essential) and Date(runway16/34-contam#265, 12-02-2020) as well as Importance(taxiway10/004-contam#343, Essential) and Date(taxiway10/004-contam#343, 12-02-2020). $\diamond$

4.2.3. Reification

The *reification* operation takes "triples" in the object knowledge of a cell and creates individuals that represent such triples. Reification allows for the preservation of duplicates in case of a union merge, which facilitates subsequent counting of occurrences in the course of the analysis. Furthermore, in combination with pivoting, the reification operation allows for attaching contextual information to context-dependent knowledge, preserving information about the context of a triple in case of a merge union.

Consider the set of asserted modules of cell c, $\overline{\mathsf{mod}}(\mathsf{c}) = \{\mathsf{m} \in \mathbf{M} \mid \mathfrak{G} \models \mathsf{mod}(\mathsf{c}, \mathsf{m})$ and, for every $\mathsf{c}' \neq \mathsf{c}$ where $\mathsf{c} \preceq \mathsf{c}', \mathfrak{G} \not\models \mathsf{mod}(\mathsf{c}', \mathsf{m})\}$. We denote the local knowledge base of cell c as $\mathrm{K}_{\overline{\mathsf{mod}}(\mathsf{c})} = \bigcup_{\mathsf{m} \in \overline{\mathsf{mod}}(\mathsf{c})} \mathrm{K}_{\mathsf{m}}$.

Definition 8 (Reification)**.** *Given a cube* $\mathfrak{K} = \langle \mathfrak{G}, \mathrm{K}_{\mathbf{M}} \rangle$*, a cell name* $\mathsf{c} \in \mathbf{F}$*, a role* R *of* $\mathcal{L}_\Sigma$ *(i.e. the reified property), we define the* reification operation $\varrho(\mathfrak{K}, \mathsf{c}, R)$ *as a new cube* $\mathfrak{K}' = \langle \mathfrak{G}', \mathrm{K}'_{\mathbf{M}} \rangle$ *over* $\langle \Gamma', \Sigma' \rangle$ *s.t.:*

- *a module name* $\mathsf{mg}(\mathsf{c})$ *is added to* $\mathbf{M}'$, $\mathsf{mod}(\mathsf{c}, \mathsf{mg}(\mathsf{c}))$ *is added to* $\mathfrak{G}'$ *and* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *is added to* $\mathrm{K}_{\mathbf{M}'}$*;*
- *a concept* R-type $\in \mathrm{NC}'_\Sigma$ *(representing the reified role type) is added to* $\langle \Gamma', \Sigma' \rangle$*;*
- *for every* $a, b \in \mathrm{NI}_\Sigma$ *s.t.* $R(a, b) \in \mathrm{K}_{\overline{\mathsf{mod}}(\mathsf{c})}$*, a new individual* R-a-b *is added to* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})}$ *with the following set of assertions (associating the subject and object to the reified role assertion):*

$$\mathsf{hasSubject}(R\text{-}a\text{-}b, a) \quad \mathsf{hasObject}(R\text{-}a\text{-}b, r) \quad R\text{-}\mathsf{type}(R\text{-}a\text{-}b)$$

Figure 14 shows an illustration of the reification operation definition. Intuitively, the reification operation considers a single cell c with its asserted knowledge $\mathrm{K}_{\overline{\mathsf{mod}}(\mathsf{c})}$ and a role R to be reified: For each asserted instance of property R, a new individual representing that instance is added to the cell's knowledge. Formally, as with pivoting, the new knowledge is added as a new knowledge module $\mathsf{mg}(\mathsf{c})$.

Example 12 (Reification)**.** Consider the K_4 knowledge module of the c_4 context from Fig. 6 (see Fig. 10 for an illustration of the cell contents). The reification operation $\varrho(\mathfrak{K}, \mathsf{c}_4, \mathsf{depth})$ then returns a new cube $\mathfrak{K}'$ with a knowledge module $\mathsf{mg}(\mathsf{c}_4)$ containing the assertions hasSubject(depth-runway16/34-contam#265-0.2, runway16/34-contam#265), hasObject(depth-runway 16/34-contam#265-0.2, 0.2), and depth-type(depth-runway16/34-contam#265-0.2).

5. Proof-of-Concept Implementation

In this section we sketch the foundations of a proof-of-concept implementation of a KG-OLAP system using off-the-shelf quad stores[5]. We refer to the appendix for additional information on the implementation.

5.1. Architecture, Model, and Operations

A mapping of the formal language to an actual RDF representation allows for the storage of KG-OLAP cubes in off-the-shelf quad stores with SPARQL realizations of the query operations. Context-aware rules serve to materialize roll-up relationships for levels and cells as well as inference and propagation of knowledge.

[5] http://kg-olap.dke.uni-linz.ac.at/

Architecture. For each KG-OLAP cube, a base repository in a quad store comprises the cube knowledge (structure) and object knowledge (contents) for the cube. Using the slice-and-dice operation, the user selects a subset of the base data into a temporary repository, which then contains a working copy of the original data that can be modified using merge and abstraction.

Multidimensional model. The definition of the KG-OLAP model primitives (e.g., cell/fact, dimension, dimension members) can be easily defined in terms of RDF/OWL classes and properties; we refer to the appendix for details. The two-layered structure of the KG-OLAP system with a global context and multiple local contexts – as in the CKR core [36] – is realized in RDF using different RDF graphs – one graph for the global knowledge and one graph for each knowledge module as well as a graph for the materialized inferred knowledge of each module.

Materialization. The reasoning procedure presented in Section 3.2.3 and the appendix, analogously to the CKR core [36], can be implemented using SPARQL-based forward rules that materialize the inferences, including coverage relationships between contexts. The KG-OLAP implementation employs the *RDFpro* framework [43] as library for the execution of rules. RDFpro allows for the specification of queries across different graphs, a feature needed for the reasoning inside individual cells as well as across different cells.

OLAP operations. The query operations introduced in Section 4 can be implemented using SPARQL queries. In particular, we implement the query operations as SPARQL `SELECT` statements that return "delta" tables which consist of quads along with an indication of the operation (insert or delete). The delta tables can then be applied to the temporary repository. We refer to the appendix for details on the implementation of query operations. We note that performance optimization was not a goal of this work.

5.2. *Performance Evaluation*

In the following, we analyze performance of the core set of KG-OLAP cube operations – i.e., slice-and-dice, merge union, and abstraction – based on experimental results. Specifically, we look at median run times for the computation of the query operations' delta statements, i.e., the statements that must be inserted or deleted in order to perform the respective query operation, measured over multiple iterations, relative to repository size (number of statements), context size (number of contexts), and delta size (number of computed delta statements). We do not include the duration of actual insertions or deletions of the delta statements in the run times since these are not specific to contextualized KGs. We refer to the appendix for more information, including an analysis of reification and pivoting performance.

In the performance experiments, we employed synthetic datasets based on the ATM use case, which allowed us to vary the number of dimensions, contexts, and statements while keeping the graphs similar. Hence, we tested query operations on three-dimensional and four-dimensional datasets with 1 365, 2 501, and 3 906 contexts, respectively. For each dimensionality and context size, we had three different repository sizes. In addition, we used "baseline" datasets for abstraction operations, which consisted of a single context, in order to investigate the impact of contextualization.

The performance experiments were conducted on a virtual CentOS 6.8 machine with four cores of an Intel Xeon CPU E5-2640 v4 with 2.4 GHz, hosting a GraphDB[6] 8.9 instance. The Java Virtual Machine (JVM) of the GraphDB instance ran with 100 GB heap space. The JVM of the KG-OLAP cube, which conducts rule evaluation and caches query results, ran with 20 GB heap space.

The GraphDB instance comprised two repositories – base and temporary – with the following configuration (see [44] for further information). The entity index size was 30 000 000 and the entity identifier size was 32 bits. Context index, predicate list, and literal index were enabled. Reasoning and inconsistency checks were disabled; the KG-OLAP implementation takes care of reasoning via RDFpro rule evaluation.

Figure 15a shows run times of the slice-and-dice operation. The plot on the left shows run time relative to repository size per dimensionality. The plot in the middle shows run time relative to repository size per context size. The plot on the right shows run time relative to the size of the delta table computed by the query operation. Hence, performance of the slice-and-dice operation primarily depends on the number of delta statements in the query result, i.e., the number of selected cells/statements from the base repository. In fact, in this example, the slice-and-dice operation performs better on the large context size (3 906 contexts) due to fewer delta statements being computed. Dimensionality does not play a role here.

[6]http://graphdb.ontotext.com/

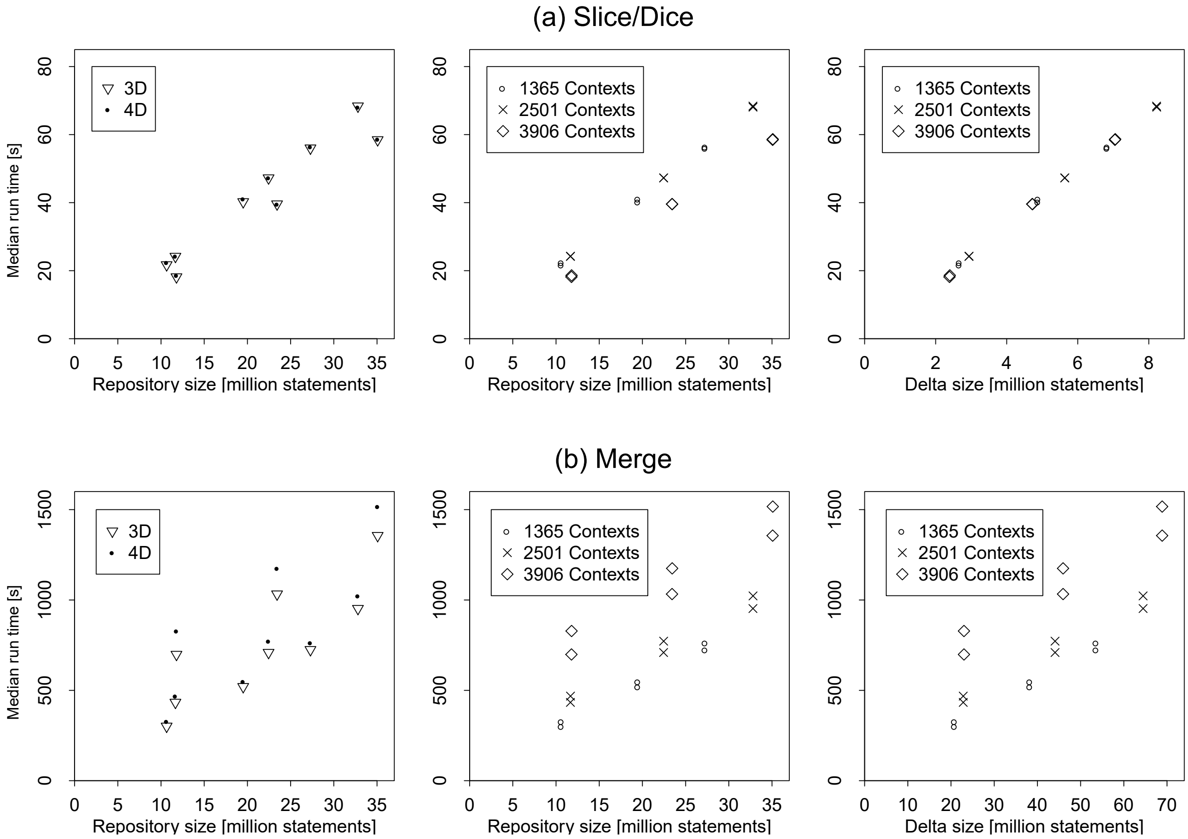


Fig. 15. Performance of contextual operations

In case of the merge operation, we deliberately chose a query that results in a massive reorganization of the KG-OLAP cube, causing a vast number of lower-level cells to be merged, in order to study worst-case performance. The run time of the merge union operation (Figure 15b) primarily depends on the number of contexts. For each context size, we observe a linear increase in run time with respect to the repository size. For the large context size (3 906 contexts) there is also a marked influence of dimensionality on run time.

For abstraction operations, we employ baseline datasets which consist of a single context comprising all statements in order to gain an understanding of the inherent complexity of these operations regardless contextualization. Such abstraction operations on a single context correspond to the formalization. In addition, we perform abstraction operations in a variant that applies to each cell at a particular granularity level. Similar to the merge operation, we deliberately choose a setting where the query operations affect a large number of statements in order to study worst-case performance.

Figure 16a shows run times of triple-generating abstraction. Run time grows linearly with repository size. Run times for individual-generating abstraction with a single grouping property look similar (Figure 16b). For triple-generating abstraction, the baseline datasets had significantly higher run times. The difference was less pronounced for individual-generating abstraction.

For value-generating abstraction, the queries resulted in smaller sizes of the computed delta tables. Figure 16c shows that the run time of value-generating abstraction grows about linearly with respect to repository size with a small influence of context size and little influence of dimensionality. Run times for the baseline datasets were smaller. We note that the run times of value-generating abstraction were quite low in general.

For results of reification and pivoting operations we refer to the appendix. In summary, the reification operation grows linearly with the repository size. For the pivoting operation we observe context size as the main factor influencing run time.

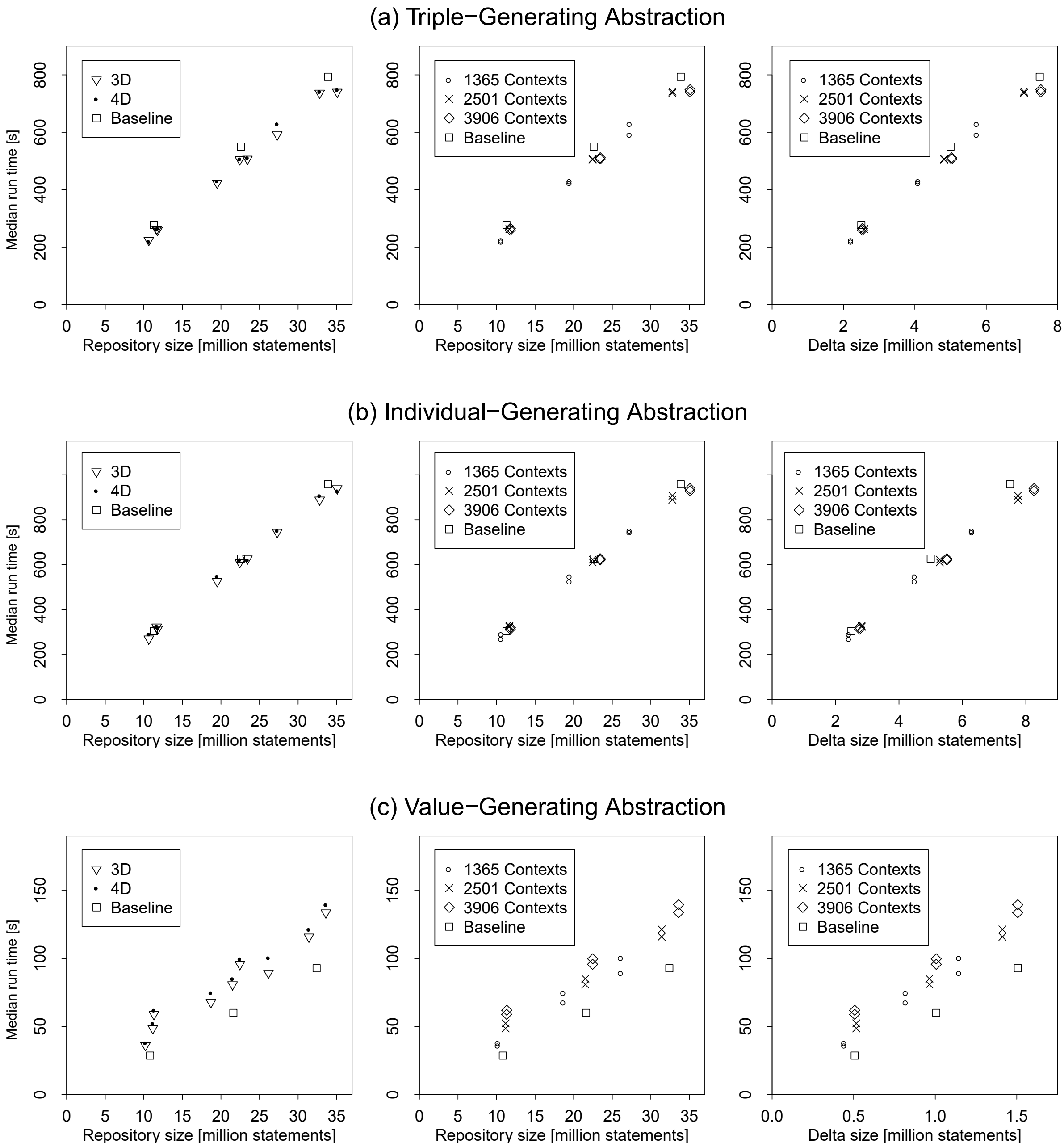


Fig. 16. Performance of abstraction operations

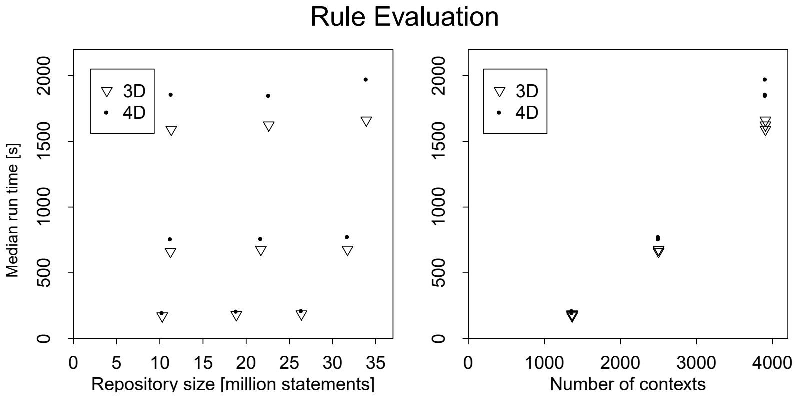


Fig. 17. Performance of rule evaluation

The proof-of-concept implementation relies on materialization of coverage relationships and inferences, which requires evaluation of a set of rules over the base repository in order to initialize the KG-OLAP cube. Figure 17 shows run times for rule evaluation relative to repository size and number of contexts for three-dimensional and four-dimensional KG-OLAP cubes. Reasoning in performance experiments was conducted with the entire repository being loaded into main memory first. Reasoning was limited to a subset of RDFS inference rules and thus the main influence for run time turned out to be the number of contexts. As with the merge operation, for the large context size, there is a marked influence of dimensionality on run time.

Precomputation of coverage relationships and inferences shifts the burden away from the individual query operations. Merge and abstraction (but not slice/dice) require rule evaluation after the operation has been performed in order to allow for possibly new inferences to be uncovered. In this regard, future work will provide an implementation that performs local reasoning only on the contexts that have been subject to change.

Concerning the ATM use case, we note that the single-machine implementation allows for employing KG-OLAP cubes for pilot briefings and post-operational analysis in individual regions. Maintaining a large-scale contextualized KG for ATM in Europe with tens of thousands of contexts in a single KG-OLAP cube, however, would require a different implementation strategy. In this regard, future work will investigate a clustered implementation with cells distributed on different server nodes and parallel computation of query operations. Another possibility is the introduction of the *metacube* [21] concept, i.e., a cube of cubes, in conjunction with the drill-across operation.

6. Related Work

Semantic technologies have been used for a variety of tasks in the context of OLAP (see [45] for an overview). Related to KG-OLAP are techniques for data analysis over RDF data. The RDF data cube vocabulary (QB) [46] and its extension, QB4OLAP [47], provide an RDF representation format for publishing traditional OLAP cubes with numeric measures on the semantic web, with often SPARQL-based operators that emulate traditional OLAP queries for analyzing multidimensional data in QB [48] and QB4OLAP [49, 50]. Such statistical linked data are just different serialization and publication formats of traditional OLAP cubes rather than KGs. Other work has suggested “lenses” over RDF data [51] for the purpose of RDF data analysis, i.e., analytical schemas which can be used for OLAP queries on RDF data. Similarly, superimposed multidimensional schemas [52] define a mapping between a multidimensional model and a KG in order to allow for the formulation of OLAP queries. Contrary to these approaches, KG-OLAP focuses on RDF graphs as the “measures” of OLAP cubes rather than numeric measures that are aggregated using aggregation operators such as SUM and AVG.

Fusion cubes [53] supplement traditional OLAP cubes with external data in RDF format, particularly linked open data where typically the data are not owned by the analyst. Fusion cubes are traditional OLAP cubes with numeric measures that can be populated dynamically with statistical data from RDF sources. Fusion cubes store contextual information about provenance and data quality of the external sources. Other similar work [54] extracts traditional OLAP cubes with numeric measures from RDF data sources and ontologies, which analysts may then query using a traditional OLAP language, namely MDX. The semCockpit project [55] employed ontologies for the definition of a shared understanding of business terms and analysis situations among business analysts. With respect to these approaches, KG-OLAP cubes may be considered a structured data lake approach (see [56] for more information), which stores the data of interest in a semantically richer format than plain numeric measures and provides dedicated query operations.

Closely related to KG-OLAP is Graph OLAP (also known as InfoNetOLAP) [17, 18], which through its informational and topological OLAP queries provides rich query facilities suitable for graph analysis. In Graph OLAP, graphs are associated with dimensional attributes, which yields a graph cube. The edges of the graphs themselves are weighted; the weights represent the measures to be analyzed. Typical applications of Graph OLAP are analysis of co-author and similar social graphs from different time periods, geographic locations, and so on. Graph OLAP distinguishes between informational roll-up and topological roll-up, which corresponds to the distinction between contextual and graph operations in KG-OLAP. Graph OLAP, however, is not suitable for working with heterogeneous KGs. Rather, Graph OLAP is another means of data analysis for a certain type of numeric measures, e.g., how many times two researchers have collaborated on a paper. The focus of Graph OLAP are weighted directed graphs with highly structured and homogeneous data. RDF data, on the other hand, have rich semantics and a more heterogeneous structure, therefore requiring specific query operators. Unlike KG-OLAP, Graph OLAP does not consider knowledge propagation and inferencing over contextualized KGs.

The KG-OLAP query operations also invite comparison with *(knowledge) graph summarization* techniques [57, 58], which aim at making KGs more accessible to end users and applications by providing a condensed view on the represented knowledge. Use cases for KG summarization include visualization and exploration of KGs as well as facilitating query formulation and processing. Broadly speaking, KG (or RDF) summarization techniques may be divided into structural summarization, mining-based, and statistical summarization [58]. Statistical summarization computes quantitative measures that characterize a graph whereas mining-based (or pattern-based) summarization employs graph mining to extract frequent patterns that act as a summary. Structural summarization aims at finding a summary graph that preserves characteristics of the original graph while considerably reducing the size of the graph, making the graph easier to handle and comprehend.

Among the structural summarization approaches for RDF graphs, quotient RDF summaries represent a common type of summaries that produce an RDF graph where multiple nodes from the source graph are replaced by a single summary node in the RDF summary. Accordingly, the results of abstraction operations in KG-OLAP may be considered *structural quotient RDF summaries* [58]. Unlike most structural approaches towards RDF summarization, KG-OLAP allows for ad hoc summarization based on user-specified, application-specific summarization criteria.

Unlike KG-OLAP, existing work on graph and KG summarization largely ignores contextuality in KGs. In fact, existing work on KG summarization is orthogonal to the KG-OLAP approach. Consequently, future work may adapt summarization algorithms to serve as graph operators in KG-OLAP.

7. Conclusion

In this paper, we presented KG-OLAP for working with KGs. Hence, we extended the multidimensional modeling paradigm from online analytical processing (OLAP) for the representation of contextualized KGs. We then introduced specific query operations: First, contextual operations for selecting and merging contexts and then graph operations for summarizing the graphs within individual contexts. We illustrated KG-OLAP using a real-world use case from the air traffic management (ATM) domain [20, 21]. A proof-of-concept implementation using off-the-shelf quad stores and SPARQL queries demonstrates feasibility. In this regard, we conducted an experimental evaluation of the performance of working with contextualized KGs. Continuing from here, future work may investigate the following:

- the extension of KG-OLAP with defeasible axioms similar to previous work [38, 59].
- potential applications of KG-OLAP in KG refinement (see [3] for more information).
- the extension of graph operations with common RDF summarization techniques.
- distributed, parallelized implementation of a KG-OLAP system, including the concept of metacube and the drill-across operation [21], in order to support big KGs.

References

[1] M. Krötzsch and G. Weikum, Editorial for special section on knowledge graphs, *Journal of Web Semantics* **37-38** (2016), 53–54. doi:10.1016/j.websem.2016.04.002.

[2] J.M. Gomez-Perez, J.Z. Pan, G. Vetere and H. Wu, Enterprise Knowledge Graph: An Introduction, in: *Exploiting Linked Data and Knowledge Graphs in Large Organisations*, J.Z. Pan, G. Vetere, J.M. Gomez-Perez and H. Wu, eds, Springer, 2017, pp. 1–14.

[3] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic Web* **8**(3) (2017), 489–508.

[4] L. Bellomarini, G. Gottlob, A. Pieris and E. Sallinger, Swift Logic for Big Data and Knowledge Graphs, in: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2–10. doi:10.24963/ijcai.2017/1.

[5] A.Y. Halevy, F. Korn, N.F. Noy, C. Olston, N. Polyzotis, S. Roy and S.E. Whang, Managing Google's data lake: an overview of the Goods system, *IEEE Data Engineering Bulletin* **39**(3) (2016), 5–14.

[6] W. Tunstall-Pedoe, True Knowledge: Open-Domain Question Answering Using Structured Knowledge and Inference, *AI Magazine* **31**(3) (2010), 80–92. doi:10.1609/aimag.v31i3.2298.

[7] V. Penela, G. Álvaro, C. Ruiz, C. Córdoba, F. Carbone, M. Castagnone, J.M. Gómez-Pérez and J. Contreras, miKrow: Semantic Intra-enterprise Micro-Knowledge Management System, in: *The Semanic Web: Research and Applications*, G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. De Leenheer and J. Pan, eds, Springer, 2011, pp. 154–168.

[8] T. Ruan, L. Xue, H. Wang, F. Hu, L. Zhao and J. Ding, Building and Exploring an Enterprise Knowledge Graph for Investment Analysis, in: *ISWC 2016*, P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck and Y. Gil, eds, LNCS, Vol. 9982, Springer, 2016.

[9] Google, Introducing the Knowledge Graph: things, not strings, 2012, https://search.googleblog.com/2012/05/introducing-knowledge-graph-things-not.html (Accessed: 05 August 2018).

[10] R. Qian, Understand your world with Bing, 2013, https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/ (Accessed: 05 August 2018).

[11] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer and C. Bizer, DBpedia – a large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6**(2) (2015), 167–195.

[12] D. Vrandečić and M. Krötzsch, Wikidata: A Free Collaborative Knowledgebase, *Communications of the ACM* **57**(10) (2014), 78–85. doi:10.1145/2629489.

[13] H. Wu, R. Denaux, P. Alexopoulos, Y. Ren and J.Z. Pan, Understanding Knowledge Graphs, in: *Exploiting Linked Data and Knowledge Graphs in Large Organisations*, J.Z. Pan, G. Vetere, J.M. Gomez-Perez and H. Wu, eds, Springer, 2017, pp. 147–180.

[14] C.G. Schuetz, B. Neumayr, M. Schrefl, E. Gringinger, A. Vennesland and S. Wilson, The Case for Contextualized Knowledge Graphs in Air Traffic Management, in: *CKG 2018*, CEUR Workshop Proceedings, Vol. 2317, CEUR-WS.org, 2018. http://ceur-ws.org/Vol-2317/article-10.pdf.

[15] L. Serafini and M. Homola, Contextualized Knowledge Repositories for the Semantic Web, *Journal of Web Semantics* **12** (2012), 64–87.

[16] A. Vaisman and E. Zimányi, *Data Warehouse Systems – Design and Implementation*, Springer, Berlin Heidelberg, 2014.

[17] C. Chen, X. Yan, F. Zhu, J. Han and P.S. Yu, Graph OLAP: a multi-dimensional framework for graph data analysis, *Knowledge and Information Systems* **21**(1) (2009), 41–63.

[18] C. Chen, F. Zhu, X. Yan, J. Han, P. Yu and R. Ramakrishnan, InfoNetOLAP: OLAP and mining of information networks, in: *Link Mining: Models, Algorithms, and Applications*, P.S. Yu, J. Han and C. Faloutsos, eds, Springer, 2010, pp. 411–438.

[19] R.M. Keller, Building a knowledge graph for the air traffic management community, in: *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 700–704. doi:10.1145/3308560.3317706.

[20] C.G. Schuetz, B. Neumayr, M. Schrefl, E. Gringinger and S. Wilson, Semantics-Based Summarization of ATM Data to Manage Information Overload in Pilot Briefings, in: *Proceedings of the 31st Congress of the International Council of the Aeronautical Sciences*, 2018. http://www.icas.org/ICAS_ARCHIVE/ICAS2018/data/papers/ICAS2018_0763_paper.pdf.

[21] C.G. Schuetz, B. Neumayr, M. Schrefl, E. Gringinger and S. Wilson, Semantics-based summarisation of ATM information: Managing information overload in pilot briefings using semantic data containers, *The Aeronautical Journal* (2019), To appear. doi:10.1017/aer.2019.74.

[22] D. Steiner, I. Kovacic, F. Burgstaller, M. Schrefl, T. Friesacher and E. Gringinger, Semantic enrichment of DNOTAMs to reduce information overload in pilot briefings, in: *Proceedings of the 16th Integrated Communications Navigation and Surveillance (ICNS) Conference*, 2016. doi:10.1109/ICNSURV.2016.7486359.

[23] B. Neumayr, E. Gringinger, C.G. Schuetz, M. Schrefl, S. Wilson and A. Vennesland, Semantic data containers for realizing the full potential of system wide information management, in: *Proceedings of the 36th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, 2017, pp. 1–10. doi:10.1109/DASC.2017.8102002.

[24] E. Gringinger, C. Schuetz, B. Neumayr, M. Schrefl and S. Wilson, Towards a value-added information layer for SWIM: The semantic container approach, in: *Proceedings of the 18th Integrated Communications Navigation and Surveillance (ICNS) Conference*, 2018, pp. 3–113114. doi:10.1109/ICNSURV.2018.8384870.

[25] C.G. Schütz, B. Neumayr and M. Schrefl, Business Model Ontologies in OLAP Cubes, in: *CAiSE 2013*, C. Salinesi, M.C. Norrie and O. Pastor, eds, LNCS, Vol. 7908, Springer, 2013, pp. 514–529. doi:10.1007/978-3-642-38709-8.

[26] Skybrary, Situational Awareness, https://www.skybrary.aero/index.php/Situational_Awareness (Accessed: 05 August 2019).

[27] R.M. Keller, Ontologies for aviation data management, in: *Proceedings of the IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016.

[28] R.M. Keller, The NASA Air Traffic Management Ontology (atmonto) – Release dated March 2018, Technical Report, National Aeronautics and Space Administration, 2018, Accessed: 05 August 2018. https://data.nasa.gov/ontologies/atmonto/.

[29] A. Vennesland, B. Neumayr, C. Schuetz, A. Savulov, S. Wilson, E. Gringinger and J. Gorman, AIRM-O – ATM Information Reference Model Ontology, 2017, https://w3id.org/airm-o/ontology.

[30] R.M. Keller, The NASA Air Traffic Management Ontology (atmontoPlus) – Release dated March 2018, Technical Report, National Aeronautics and Space Administration, 2018, Accessed: 05 August 2018. https://data.nasa.gov/ontologies/atmontoPlus/.

[31] International Civil Aviation Organization, *Annex 15 to the Convention on International Civil Aviation: aeronautical information services*, 13 edn.

[32] Federal Aviation Administration, *Federal NOTAM system airport operations scenarios*, 2010, Accessed: 05 August 2018. https://notams.aim.faa.gov/FNSAirportOpsScenarios.pdf.

[33] S. Niarchakou and J. Simón Selva, *ATFCM operations manual – network operations handbook*, 21.0 edn, 2017, Accessed: 05 August 2019. http://www.eurocontrol.int/sites/default/files/content/documents/nm/network-operations/HANDBOOK/ATFCM-Operations-Manual-next.pdf.

[34] AIXM 5.1.1 - Data Model (UML), Accessed: 05 August 2019. http://aixm.aero/document/aixm-511-data-model-uml.

[35] R. Lake, D.S. Burggraf, M. Trninić and L. Rae, *Geography Mark-Up Language: foundation for the geo-web*, John Wiley & Sons, 2004.

[36] L. Bozzato and L. Serafini, Materialization calculus for contexts in the Semantic Web, in: *DL 2013*, CEUR Workshop Proceedings, Vol. 1014, CEUR-WS.org, 2013. http://ceur-ws.org/Vol-1014/paper_51.pdf.

[37] M. Golfarelli, D. Maio and S. Rizzi, The dimensional fact model: a conceptual model for data warehouses, *International Journal of Cooperative Information Systems* **7**(2–3) (1998), 215–247.

[38] L. Bozzato, T. Eiter and L. Serafini, Enhancing context knowledge repositories with justifiable exceptions, *Artificial Intelligence* **257** (2018), 72–126.

[39] F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P. Patel-Schneider (eds), *The Description Logic Handbook*, Cambridge University Press, 2003.

[40] M. Krötzsch, Efficient Inferencing for OWL EL, in: *JELIA 2010*, LNCS, Vol. 6341, Springer, 2010, pp. 234–246. doi:10.1007/978-3-642-15675-5_21.

[41] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue and C. Lutz, OWL 2 Web Ontology Language Profiles (Second Edition) – W3C Recommendation 11 December 2012, Technical Report, W3C, 2009, https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/.

[42] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P.F. Patel-Schneider and U. Sattler, OWL 2: The next step for OWL, *Journal of Web Semantics* **6**(4) (2008), 309–322.

[43] F. Corcoglioniti, M. Rospocher, M. Mostarda and M. Amadori, Processing Billions of RDF Triples on a Single Machine Using Streaming and Sorting, in: *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015, pp. 368–375. doi:10.1145/2695664.2695720.

[44] Ontotext, Configuring a repository, http://graphdb.ontotext.com/documentation/8.9/free/configuring-a-repository.html (Accessed: 31 July 2019).

[45] A. Abello, O. Romero, T.B. Pedersen, R. Berlanga, V. Nebot, M.J. Aramburu and A. Simitsis, Using semantic web technologies for exploratory OLAP: a survey, *IEEE Transactions on Knowledge and Data Engineering* **27**(2) (2015), 571–588.

[46] R. Cyganiak and D. Reynolds, The RDF Data Cube Vocabulary – W3C Recommendation 16 January 2014, Technical Report, W3C, 2014, https://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/.

[47] L. Etcheverry and A.A. Vaisman, QB4OLAP: A Vocabulary for OLAP Cubes on the Semantic Web, in: *COLD 2012*, CEUR Workshop Proceedings, Vol. 905, CEUR-WS.org, 2012.

[48] M. Meimaris, G. Papastefanatos, P. Vassiliadis and I. Anagnostopoulos, Efficient Computation of Containment and Complementarity in RDF Data Cubes, in: *Proceedings of the 19th Conference on Extending Database Technology (EDBT 2016)*, 2016, pp. 281–292. doi:10.5441/002/edbt.2016.27.

[49] L. Etcheverry, A.A. Vaisman and E. Zimányi, Modeling and Querying Data Warehouses on the Semantic Web Using QB4OLAP, in: *DaWaK 2014*, LNCS, Vol. 8646, Springer, 2014, pp. 45–56.

[50] J. Varga, L. Etcheverry, A.A. Vaisman, O. Romero, T.B. Pedersen and C. Thomsen, QB2OLAP: Enabling OLAP on Statistical Linked Open Data, in: *Proceedings of the 32nd IEEE International Conference on Data Engineering (ICDE 2016)*, 2016, pp. 1346–1349. doi:10.1109/ICDE.2016.7498341.

[51] D. Colazzo, F. Goasdoué, I. Manolescu and A. Roatiş, RDF Analytics: Lenses over Semantic Graphs, in: *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 467–478. doi:10.1145/2566486.2567982.

[52] M. Hilal, C.G. Schuetz and M. Schrefl, Using superimposed multidimensional schemas and OLAP patterns for RDF data analysis, *Open Computer Science* **8**(1) (2018), 18–37. doi:10.1515/comp-2018-0003.

[53] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J. Mazón, F. Naumann, T.B. Pedersen, S. Rizzi, J. Trujillo, P. Vassiliadis and G. Vossen, Fusion Cubes: Towards Self-Service Business Intelligence, *International Journal of Data Warehousing and Mining* **9**(2) (2013), 66–88.

[54] V. Nebot and R. Berlanga Llavori, Building data warehouses with semantic web data, *Decision Support Systems* **52**(4) (2012), 853–868.

[55] T. Neuböck, B. Neumayr, M. Schrefl and C. Schütz, Ontology-Driven Business Intelligence for Comparative Data Analysis, in: *eBISS 2013*, LNBIP, Vol. 172, Springer, 2014, pp. 77–120.

[56] P. Russom, Data Lakes: Purposes, Practices, Patterns, and Platforms, 2017, Accessed: 05 August 2019. https://tdwi.org/research/2017/03/best-practices-report-data-lakes.

[57] Y. Liu, T. Safavi, A. Dighe and D. Koutra, Graph Summarization Methods and Applications: A Survey, *ACM Computing Surveys* **51**(3) (2018). doi:10.1145/3186727.

[58] Š. Čebirić, F. Goasdoué, H. Kondylakis, D. Kotzinos, I. Manolescu, G. Troullinou and M. Zneika, Summarizing semantic graphs: a survey, *The VLDB Journal* **28**(3) (2019), 295–327. doi:10.1007/s00778-018-0528-3.

[59] L. Bozzato, L. Serafini and T. Eiter, Reasoning with Justifiable Exceptions in Contextual Hierarchies, in: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference (KR 2018)*, M. Thielscher, F. Toni and F. Wolter, eds, AAAI Press, 2018, pp. 329–338. https://aaai.org/ocs/index.php/KR/KR18/paper/view/18032.