Semantic Web 0 (2020) 1–35
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] *Institute 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

Editor: Harald Sack, Karlsruhe Institute of Technology, Germany
Solicited reviews: Sven Groppe, University of Lübeck, Germany; Aidan Hogan, Universidad de Chile, Chile; Maribel Acosta, Ruhr-University Bochum, Germany; One anonymous reviewer

Abstract. A knowledge graph (KG) represents real-world entities and their relationships. The represented knowledge is often context-dependent, leading to the construction of contextualized KGs. The multidimensional and hierarchical nature of context invites comparison with the OLAP cube model from multidimensional data analysis. Traditional systems for online analytical processing (OLAP) employ multidimensional models to represent numeric values for further analysis using dedicated query operations. In this paper, along with an adaptation of the OLAP cube model for KGs, we introduce an adaptation of the traditional OLAP query operations for the purposes of performing analysis over 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 – a *management summary* – of the knowledge.

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

1. Introduction

A knowledge graph (KG) represents real-world entities and their relationships. 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]. For the representation of KGs, the Resource Description Framework (RDF) [5] often serves as the data model.

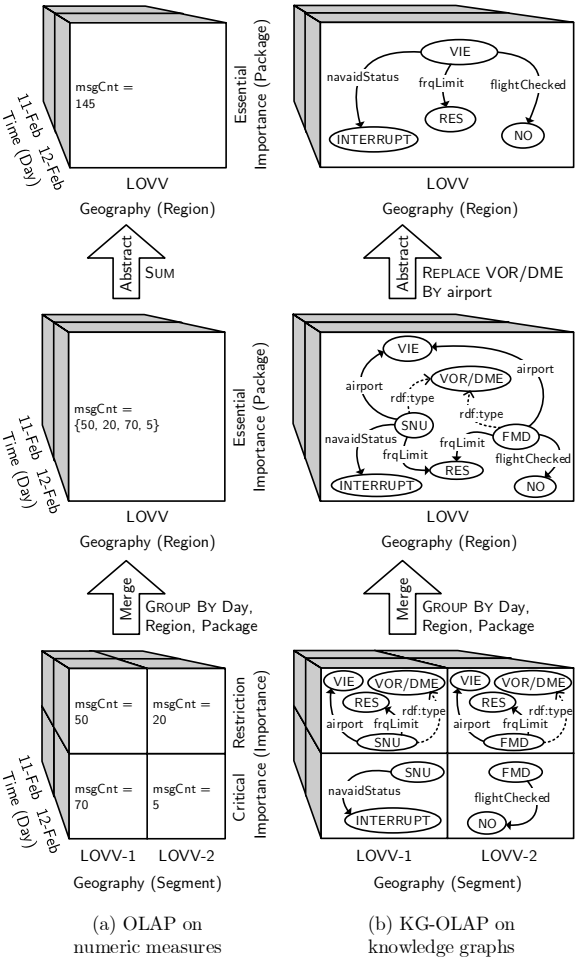
KGs present a wide range of potential applications, e.g., (web) search [6] and question-answering [7], intra-company knowledge management [8] and investment analysis [9]. Among the most popular examples of KGs are proprietary ones such as Google's Knowledge Graph [10] and Microsoft's Satori [11] as well as community-driven efforts such as DBpedia [12] and Wikidata [13]. 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 [14].

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

In order to facilitate KG management, KGs are increasingly subject to *contextualization*, i.e., the enrichment of facts with context metadata such as time and location. For example, in the aeronautics domain, the relevant knowledge for air traffic management is inherently context-dependent [15], especially with respect to time and location but also other context dimensions, e.g., importance or topic. In particular, knowledge about airport infrastructure and airspace such as operational status of runways and closure of airspace varies over time. Frameworks such as the *Contextualized Knowledge Repository* (CKR) [16] serve to organize knowledge within hierarchically ordered contexts along multiple context dimensions.

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 [17] 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.

In this paper, we introduce *Knowledge Graph OLAP* (KG-OLAP), a conceptual approach that consists of a multidimensional model and corresponding query operations for performing analysis over KGs. Extending the CKR framework [16, 18], KG-OLAP cubes collect knowledge into hierarchically ordered contexts: Each cell of a KG-OLAP cube corresponds to a context, with knowledge encoded as 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. Regarding query operations, the KG-OLAP framework distinguishes two categories: *contextual* and *graph* operations. The central types of operation in each category are *merge* and *abstraction*, respectively. In particular, a merge operation combines the knowledge from different contexts whereas an abstraction operation replaces individual entities within a context with more abstract entities; different variants for each of those types of operation exist.



(a) OLAP on numeric measures

(b) KG-OLAP on knowledge graphs

Fig. 1. Traditional OLAP operates on numeric measures whereas KG-OLAP operates on knowledge graphs

Figure 1 draws an analogy between (a) the roll-up operation in traditional OLAP on numeric measures and (b) merge and abstraction in KG-OLAP on KGs. First, the example emphasizes the fact that traditional OLAP and KG-OLAP work on distinct types of data. The example's general setting is air traffic management, where messages dispatched by air traffic control notify of temporary changes in infrastructure. In traditional OLAP, the description of real-world events is condensed into numeric measures: In this example, the OLAP cube captures the daily number of messages dispatched by air traffic control per geographic segment and message importance. In KG-OLAP, each cell of the cube con-

tains knowledge, encoded as triples, valid and relevant in the context that the cell represents: In this example, knowledge about navigation aids of varying importance, valid on a particular day, and relevant for a specific geographic segment. In particular, the KG-OLAP cube comprises knowledge about navigation aids used for approaching Vienna airport (VIE): the VOR/DME (i.e., a type of radio beacon) in Sollenau (SNU) and the VOR/DME in Fischamend (FMD), which are located in different geographic segments. On the one hand, that knowledge concerns restrictions, namely limitations of frequency (frqLimit), indicating usability of the navigation aid's frequency to be restricted (RES) to certain sectors (not shown in the example). On the other hand, there is the flight critical knowledge about a temporary change in SNU's operational status to INTERRUPT and the fact that FMD is currently not "flight checked", i.e., has not undergone mandatory, periodic security checks. To illustrate the idea of merge and abstraction – the main types of operations in KG-OLAP – the roll-up operation in traditional OLAP can be considered a sequence of merge and abstraction. In traditional OLAP, the merge operation obtains a collection of numeric values for each grouping of cells by day, flight information region, and importance package; each geographic segment belongs to a region, each importance to an importance package. Then, the abstraction operation applies the SUM aggregation operator on the collection of numeric values to obtain an aggregate numeric value. In KG-OLAP, the merge operation first combines triples from the more specific contexts into a more general context. Then, the abstraction operation replaces each VOR/DME entity by the target of its airport property to obtain a more abstract representation of the knowledge about navigation aids at Vienna airport. Just like there are different aggregation functions for numeric values, there are different variants of the abstraction operation for KGs. In addition, in case of KG-OLAP, there are different variants of the merge operation. In KG-OLAP, merge and abstraction may also be conducted repeatedly and in an arbitrary order.

We illustrate KG-OLAP following use cases of (contextualized) knowledge graphs for *air traffic management (ATM)* [15, 19]; we draw from experience in collaborative research projects on the use of semantic technologies in ATM (see [20–22]). ATM KGs 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. Merge and abstraction then serve to obtain a *management summary* of the represented knowledge for pilot briefings or post-operational analysis, providing a more abstract view on the KG which contains relevant ATM knowledge in the suitable form for a particular situation. Another potential application of KG-OLAP is the analysis of business reports formalized using KGs and business model ontologies [23], which we do not elaborate further in this paper.

Previous work [24, 25] introduced the concept of *ATM information cubes*, presenting a use case for KG-OLAP in pilot briefings to an ATM audience, without the conceptual and technological fundamentals of KGs in general or KG-OLAP in particular. The KG-OLAP framework may serve to realize ATM information cubes, which can be considered a simplified version of KG-OLAP cubes. ATM information cubes organize messages, the contents thus being considerably less complex and comprehensive than (contextualized) KGs, lacking ontological knowledge and the knowledge propagation mechanism of KG-OLAP, among other things. The work on ATM information cubes also omits the formal definition of abstraction query operations while lacking other graph operations altogether.

The contributions of this paper are:

(i). the identification of requirements for a model and query operations for performing analysis over KGs;
(ii). the formalization of a model for KGs with hierarchical contexts and knowledge propagation suitable for performing analysis;
(iii). the definition of a set of query operations for performing analysis over KGs;
(iv). an experimental analysis of run time performance for working with contextualized KGs.

The remainder of this paper is organized as follows. In Section 2, we present the use cases that serve to illustrate KG-OLAP and motivate the approach. In Section 3, we introduce the formalization for the multidimensional model of 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.

Further details about used language, reasoning methods, implementation, and experimental evaluation are provided in the separate Appendix [26].

2. Use Cases in Air Traffic Management

In this section, we give relevant background information on air traffic management (ATM) before discussing the use of (contextualized) KGs for aeronautical information management. We then describe two potential use cases of KG-OLAP in the ATM domain, from which we derive functional requirements.

2.1. Background

Modern ATM aims 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 the 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" [27], 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).

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 [28], e.g., closures of aerodromes, runways, and taxiways, surface conditions, and construction activities (see [29] for a list of airport event scenarios) but also airspace restrictions. Messages shape the knowledge about the world as relevant for ATM. For example, a DNOTAM (Listing 1) may change the knowledge about the taxiways of a particular airport by announcing the temporary closure of a taxiway due to snow removal. 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, a DNOTAM typically employs *snapshot* timeslices, i.e., 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 5-13) and the definition of various attributes of a taxiway at Vienna airport (Lines 17-27) per 12 February 2018 at 8:00 am. The tempdelta knowledge consists of the notification of a taxiway closure due to snow removal (Lines 29-50) on 12 February 2018 from 8:00 am to 10:00 am.

2.2. (Contextualized) ATM Knowledge Graphs

The knowledge encoded in DNOTAMs is more naturally represented using contextualized KGs [15]. 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 12 February 2018, on the other hand, defines knowledge about a temporarily reduced availability – a closed operational status – due to snow removal. Other context dimensions may also serve to organize ATM knowledge into contextualized KGs [15], e.g., geography, topic, and importance. Besides the knowledge derived from DNOTAMs according to the AIXM, an ATM KG may comprise knowledge from data items according to other ATM information (exchange) models, e.g., IWXXM for weather or FIXM for flight plans.

In our scenario, RDF serves as the common representation language for ATM knowledge even though XML is the native serialization 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" [30, p. 20], including the GML's object-property model [30, p. 16]. Similarly, other ATM information (exchange) models are easily translated into RDF.

In fact, ATM research has shown growing interest in the use of semantic technologies (see [31] for an overview), leading to the development of domain ontologies [32, 33], e.g., the *NASA ATM Ontology* (ATMONTO) [34] and the *AIRM Ontology* [35]. NASA's ATMGRAPH [19, 36], in turn, is a KG for the ATM domain that builds on ATMONTO and comprises knowledge about infrastructure, flights, and operating conditions. The rationale behind ATMGRAPH lies in the integration of heterogeneous data from various sources for analysis purposes. Compared to a data lake solution storing raw files from various data sources [37], which shifts the burden of data integration

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

```
<AIXMBasicMessage>
  <hasMember>
    <AirportHeliport id="VIE">
      <timeSlice>
        <AirportHeliportTimeSlice
            id="VIE_TS1">
          <validTime>
            <TimeInstant
                id="VIE_TS1_TI">
              <timePosition>
                2018-02-12T08:00:00
          <interpretation>SNAPSHOT
          <designator>LOWW
  <hasMember>
    <Taxiway id="VIE_TWY1">
      <timeSlice>
        <TaxiwayTimeSlice
            id="VIE_RWY1_TS1">
          <validTime>
            <TimeInstant
                id="VIE_RWY1_TS1_TI">
              <timePosition>
                2018-02-12T08:00:00
          <interpretation>SNAPSHOT
          <associatedAirportHeliport
              xlink:href="VIE"/>
          <designator>10/004
      <timeSlice>
        <TaxiwayTimeSlice
            id="VIE_TWY1_TS2">
          <validTime>
            <TimePeriod
                id="VIE_TWY1_TS2_TP">
              <beginPosition>
                2018-02-12T08:00:00
              <endPosition>
                2018-02-12T10:00:00
          <interpretation>TEMPDELTA
          <availability>
            <ManoeuvringAreaAvailabilty
                id="VIE_TWY1_AV1">
              <operationalStatus>CLOSED
              <annotation>
                <Note
                    id="VIE_TWY1_NT1">
                  <propertyName>
                    operationalStatus
                  <purpose>REMARK
                  <note>
                    DUE TO SNOW REMOVAL
```

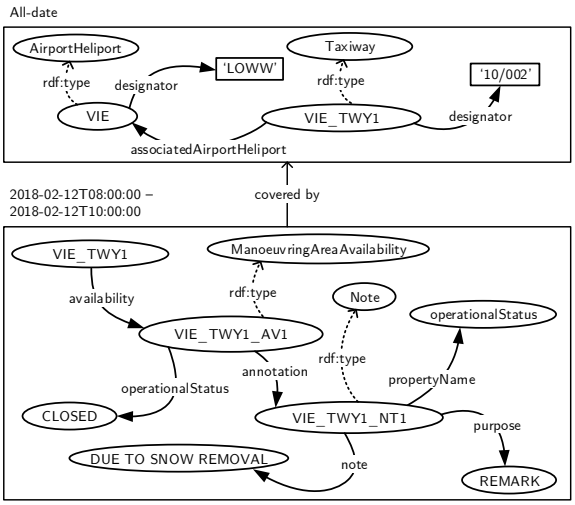


Fig. 2. A contextualized KG based on the DNOTAM in Listing 1

largely to analysts, ATMGRAPH facilitates querying across sources [38, p. 2]. Using semantic technologies was also deemed preferable over a relational implementation due to increased flexibility regarding modifications, possibility of inferencing, and reusability of the knowledge [38, p. 3]. In summary, employing (contextualized) KGs for aeronautical information management seems promising.

In the remainder of this paper, we use example KGs for illustration purposes largely following the AIXM conceptual models [39] and the FAA's airport operations scenarios [29]. In the following, we present two use cases for contextualized KGs in the ATM domain, which serve to motivate the KG-OLAP approach.

2.3. Use Case 1: Pilot Briefings

Prior to a flight, pilots receive a pre-flight information bulletin (PIB), which comprises all DNOTAMs, but also METARs, i.e., messages about weather conditions, that are even remotely relevant for the upcoming flight. Pilots then have to manually sift through an abundance of messages and decide upon the priority of each message. Among other things, the priority of a message depends on the current flight phase. For example, during take-off, a taxiway closure at the destination airport has low priority for a pilot.

Automated rule-based filtering and prioritization of DNOTAMs reduces information overload in pilot briefings [20]. Pilots formulate an *interest specification* [40], which comprises spatial, temporal, and aircraft interests [41]. The SemNOTAM system then matches

DNOTAMs against the interest specification and assigns an importance, e.g., flight critical or operational restriction, to each DNOTAM. DNOTAMs are also classifed according to geographic scope and flight phase (spatio-temporal) as well as event scenario. The general approach is also applicable to other types of ATM information, e.g., METARs, provided appropriate filtering and prioritization rules are available.

The result of DNOTAM filtering and prioritization may be packaged into a *semantic container* [21]. A semantic container collects data items into a package that comprises all the data items satisfying a certain *membership condition*, formulated in terms of an ATM domain ontology. A membership condition may have multiple facets, e.g., geography, time, aircraft, and importance. Semantic containers foster reuse of previously compiled packages of ATM information, thus reducing processing effort, and allow for replication in order to increase availability and conserve bandwidth. Raw data items, not knowledge triples, constitute the contents of a semantic container. For example, a semantic container may comprise the relevant DNOTAMs for a flight from Dubai to Vienna on a particular day, along with additional information about the importance of each individual DNOTAM.

Automated rule-based filtering and prioritization in combination with semantic containers may serve to construct ATM information cubes [24, 25]. Fine-grained semantic containers comprising data items of different geographic and temporal scopes as well as importance levels may be arranged in a multidimensional space. A pilot could then select the appropriate containers at the right moment without being overloaded with information. Individual messages could be combined into a more abstract representation in order to obtain a *management summary* of relevant events. Runway closures at the destination airport, for example, may be flight critical and pilots should be aware of the fact that there are closures in effect at the destination also in the early phases of a flight, possibly with a general indication of the reason for closure, e.g., due to snow. In detail, however, the runway closures concern the pilot only upon approach of the destination, e.g., which runway directions are closed, which types of snow are found on the runways.

In this paper, rather than considering ATM information cubes with messages collected into semantic containers, we consider *cubes of ATM knowledge*. More general contexts establish a common schema and business term vocabulary, which is inherited and extended by the more specific contexts. The messages represent a source of knowledge about the state of the world with limited spatial, temporal, and content scope. The information contained in the messages translates into knowledge triples that are collected into contexts based on the scope of the messages.

2.4. Use Case 2: Post-Operational Analysis

Air traffic flow and capacity management (ATFCM), which represents one of the core activities in ATM, has post-operations teams analyzing operational events in order to identify valuable lessons learned for the benefit of future operations and produces an overview of occurred incidents [42, p. 131]. A data warehouse provides the post-operations team with statistical data about past flight operations [42, p. 130].

Post-operational analytical tasks in ATFCM may also leverage contextualized ATM knowledge. The rationale is similar to ATMGRAPH's [38], a KG having richer semantics while being more flexible and versatile than simple statistical data organized in a data warehouse. In addition to the data warehouse, the post-operations team may employ a KG-OLAP cube of contextualized ATM knowledge extracted from DNOTAMs and other types of messages, organized 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. Using the *merge* operation, the post-operations team may combine ATM knowledge from different contexts. For example, the relevant ATM knowledge per day and geographic segment could be combined to obtain the ATM knowledge per month and geographic region and month. 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. 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 the remainder of this paper, we employ contextualized KGs for the ATM domain to illustrate the KG-OLAP approach. In particular, we focus mainly on ATM knowledge derived from DNOTAMs. The example contextualized KGs are plausible for both pilot briefings and post-operational analysis.

2.5. Functional Requirements

Based on the presented use cases from the ATM domain, we identify functional requirements that a system for performing data analysis over KGs must fulfill. Although derived from use cases in ATM, the requirements also apply to other use cases, e.g., the analysis of business situations formalized using business model ontologies [23].

Requirement 1 (Heterogeneity)**.** The system must cope with heterogeneity in the knowledge representation. Heterogeneity is inherent to KGs, which are not simple graphs. In particular, heterogeneity in KGs manifests itself as follows:

(a). Multiple types of entities. In the ATM domain, for example, a KG comprises knowledge about various kinds of infrastructure but also flight plans, airspace, weather events, and more.
(b). Multiple types of relationships. Unlike simple graphs, the entities in a KG are connected via various relationships. For example, one type of relationship serves to indicate the airport that a runway is situated at while another associates a runway with a usage restriction.
(c). Schema variability, i.e., entities of the same type may have different properties and relationships. For example, runway availability may warn of inspection adjacent to a runway. But, runway availability may also announce runway closure, e.g., due to snow, or mention the characteristics of aircraft that are prohibited to land, which in turn may depend on weight, wingspan, etc.

Requirement 2 (Ontological knowledge)**.** The system must handle ontological knowledge that describes relationships between classes and employs logic for the definition of domain-specific terms. ATM information (exchange) models, for example, comprise a multitude of generalization/specialization relationships and define associations between classes, which potentially translate into domain and range constraints. ATM technical language is rife with domain-specific terms, e.g., heavy wake-turbulence aircraft, the meaning of which can be defined using an ontology, which facilitates use of business terms in queries [40].

Requirement 3 (Self-describing data)**.** The system must store metadata along with the instance data in order to facilitate interpretation. As a general trait of semantic systems, definitions of classes and properties are a flexible part of the data, not encoded in the system's physical schema. The RDF data format, for example, serves for the representation of instance data and class-level data alike. Changes in the data model do not culminate in a refactoring of the database schema in the same way as in relational databases. Furthermore, query operations may directly operate on the metadata. A query operation may, for example, obtain an overview of relationships where individual entities are replaced by their classes.

Requirement 4 (Modularization)**.** The system must allow for the modularization of knowledge according to different criteria. Knowledge that belongs together can be represented together. Separate knowledge with different scopes can be split into multiple modules. For example, some knowledge, e.g., a particular runway closure, may be relevant for LOWW airport while other knowledge, e.g., reduced quality of a specific navigation aid, may be relevant for LOWL airport.

Requirement 5 (General and specific knowledge)**.** The system must support both modules with more general scope and modules with more specific scope. For example, in some cases, associating particular knowledge with individual geographic segments is impossible. Instead, the knowledge could be relevant for the entire LOVV region and not only LOWW airport. When selecting relevant knowledge for the LOWW airport, however, relevant knowledge for the entire region should also be included.

Requirement 6 (Knowledge selection and combination)**.** The system must provide query operations for selecting and combining knowledge from different modules. For example, in a pilot briefing, depending on the flight phase, knowledge with a specific geographic applicability is relevant, which must be selected from multiple modules.

Requirement 7 (Knowledge abstraction)**.** The system must provide query operations that allow to obtain a more abstract view on the represented knowledge. For example, in some situations, the knowledge about various layers of dry snow, compact snow, etc. on different runways of an airport, may be accurately summarized by the fact that the runways at LOWW airport have snow contamination.

In the following, we formulate the KG-OLAP cube model and query operations guided by the identified functional requirements.

3. Multidimensional Model

In this section, we introduce the *KG-OLAP cube model* for the representation of contextualized KGs. We first introduce the model informally before providing a formal definition. We define the model as an extension of the Contextualized Knowledge Repository (CKR) framework [16, 18].

3.1. KG-OLAP Cube Model

KG-OLAP adapts the multidimensional modeling paradigm from data warehousing (see [17]) 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 the contents of the cells. The two layers employ distinct and possibly disjoint languages.

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 [43], 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. ◇

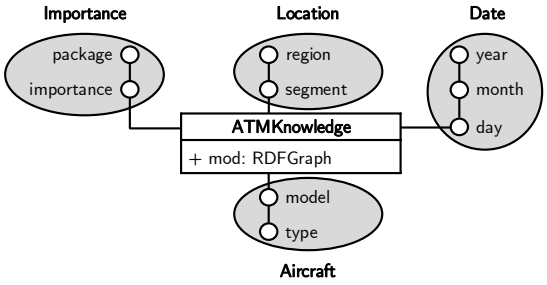


Fig. 3. A KG-OLAP cube with its dimensions and levels in DFM notation for the organization of KGs in air traffic management

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. Each dimension is represented as a tree. The dimension's name is 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 hierar-

[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.

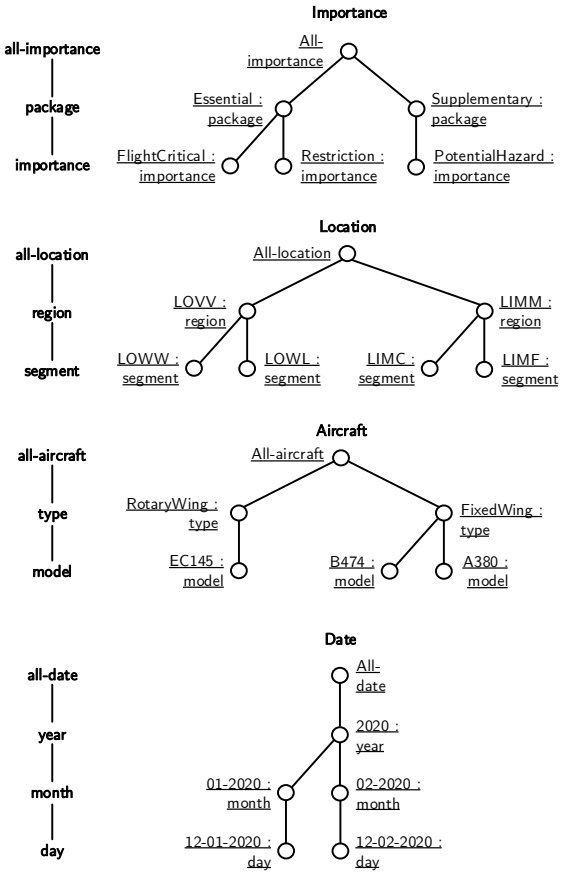


Fig. 4. Example hierarchies for the context dimension members

chies organize the cells into a hierarchical structure. For example, the combination of dimension members June 2016 and Austria identifies a particular cell in a two-dimensional cube with time and location dimensions. The hierarchical order of dimension members then 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 23 June 2016 and Vienna. With each cell, a KG-OLAP cube then associates a knowledge module, which comprises facts of knowledge valid in the respective context.

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, 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 also 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_3's importance, temporal, and aircraft attributes are more general than c_2's attributes in the respective dimensions but c_3's location attribute is more specific 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 FixedWing aircraft. 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. In the examples, we employ $\mathcal{SROIQ}$-RL as the object language, which corresponds to the OWL 2

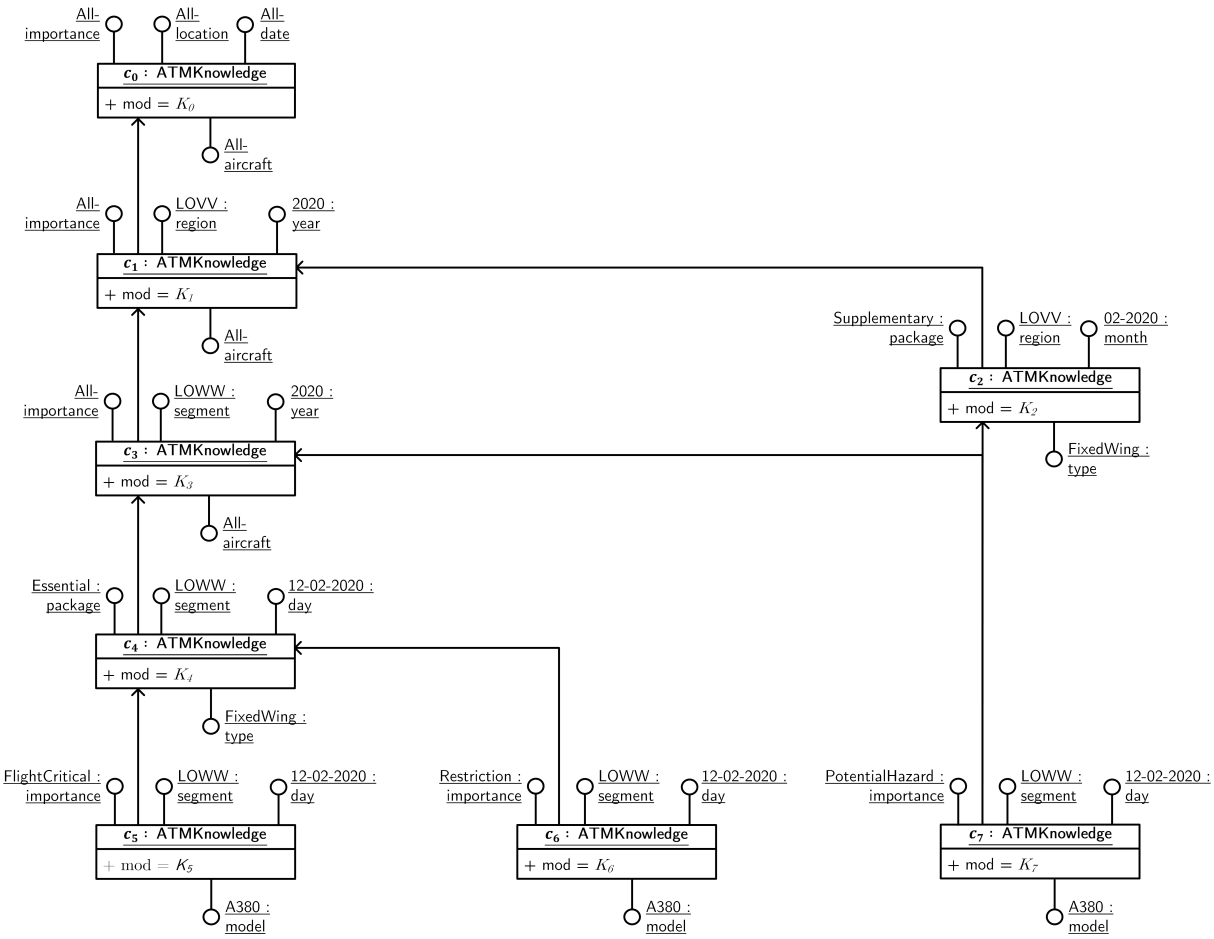


Fig. 5. An example instance of the KG-OLAP cube schema in Fig. 3. The arrows denote coverage relationships between contexts; the covered context points to the covering. The contents of the knowledge modules K_0-K_7 are defined in Fig. 6.

RL profile, as it will be the language of reference for our formal definition of reasoning in Sect. 3.2.3. We note, however, that our approach is agnostic to the language chosen for the representation of local knowledge.

Example 4 (Knowledge modules)**.** Figure 6 defines (using DL syntax) contents for the knowledge modules K_0-K_7 associated with the KG-OLAP cube cells c_0-c_7 from Fig. 5. The representation of module contents generally follows the AIXM standard [39] and the FAA's airport operations scenarios [29], with minor modifications for illustration purposes.

The K_0 module (Rows 1-35) defines concepts and terminological axioms valid across all contexts but also concept and role assertions of general interest. In particular, the module defines a designator property, which assigns an identifying string to an infrastructure element (Row 1), e.g., an airport/heliport, runway, or taxiway. The isSituatedAt property links an infrastructure element to an AirportHeliport (Row 2).

The availability property links a Runway or Taxiway element to a ManoeuvringAreaAvailability individual (Row 3), which typically indicates, via the operationalStatus property (Row 4), a temporal change in the runway or taxiway's operational status, e.g., closed or limited (Row 5). A ManoeuvringAreaAvailability individual may be annotated with a Note (Row 6) for a certain purpose (Rows 7 and 8), e.g., as a remark (Row 9) that gives an additional explanation via the note property (Row 10) for the value of the property referred to by the note's propertyName attribute (Row 11). For example, a DueToNote individual attached to the Note via the note property may indicate why the operationalStatus property assumes the value closed, e.g., due-

K_0	$\exists$designator$^-$.$\top \sqsubseteq$ String	(1)
	$\exists$isSituatedAt$^-$.$\top \sqsubseteq$ AirportHeliport	(2)
	Runway $\sqcup$ Taxiway $\sqsubseteq \forall$availability.ManoeuvringAreaAvailability	(3)
	ManoeuvringAreaAvailability $\sqsubseteq \forall$operationalStatus.StatusAirport	(4)
	StatusAirport(closed), StatusAirport(limited)	(5)
	$\exists$annotation$^-$.$\top \sqsubseteq$ Note	(6)
	$\exists$purpose.$\top \sqsubseteq$ Note	(7)
	$\exists$purpose$^-$.$\top \sqsubseteq$ NotePurpose	(8)
	NotePurpose(remark)	(9)
	$\exists$note.$\top \sqsubseteq$ Note	(10)
	$\exists$propertyName.$\top \sqsubseteq$ Note	(11)
	DueToNote(dueToSnowRemoval)	(12)
	$\exists$warning$^-$.$\top \sqsubseteq$ AirportWarning	(13)
	AirportWarning(inspection)	(14)
	$\exists$warningAdjacent$^-$.$\top \sqsubseteq$ Boolean	(15)
	ManoeuvringAreaAvailability $\sqsubseteq \forall$usage.ManoeuvringAreaUsage	(16)
	ManoeuvringAreaUsage $\sqsubseteq \forall$operation.OperationManoeuvring	(17)
	OperationManoeuvring(landing)	(18)
	$\exists$limitationType$^-$.$\top \sqsubseteq$ UsageLimitation	(19)
	UsageLimitation(forbid)	(20)
	$\exists$aircraft$^-$.$\top \sqsubseteq$ AircraftCharacteristic	(21)
	HeavyWakeCharacteristic $\sqsubseteq$ AircraftCharacteristic	(22)
	$\exists$contaminant$^-$.$\top \sqsubseteq$ SurfaceContamination	(23)
	$\exists$depth.$\top \sqsubseteq$ SurfaceContamination	(24)
	$\exists$layer.$\top \sqsubseteq$ SurfaceContamination	(25)
	$\exists$layer$^-$.$\top \sqsubseteq$ SurfaceContaminationLayer	(26)
	$\exists$contaminationType.$\top \sqsubseteq$ SurfaceContaminationLayer	(27)
	$\exists$contaminationType$^-$.$\top \sqsubseteq$ ContaminationType	(28)
	ContaminationType(drySnow)	(29)
	kindOf(drySnow, snow)	(30)
	$\exists$usedForHomingAt.$\top \sqsubseteq$ Navaid	(31)
	$\exists$usedForHomingAt$^-$.$\top \sqsubseteq$ AirportHeliport	(32)
	VOR $\sqcup$ DME $\sqsubseteq$ Navaid	(33)
	VOR $\sqcap$ DME $\equiv$ VOR/DME	(34)
	$\exists$frequency$^-$.$\top \sqsubseteq$ Number	(35)
K_1	Airport(airportLOWW)	(36)
	designator(airportLOWW, 'LOWW')	(37)
	VOR/DME(vor/dmeFMD)	(38)
	designator(vor/dmeFMD, 'FMD')	(39)
	usedForHomingAt(vor/dmeFMD, airportLOWW)	(40)
	HeavyWakeCharacteristic $\sqsubseteq \exists$weightCategory.{above136t}	(41)
K_2	frequency(vor/dmeFMD, 110.8)	(42)
K_3	Taxiway(taxiway10/004)	(43)
	designator(taxiway10/004, '10/004')	(44)
	isSituatedAt(taxiway10/004, airportLOWW)	(45)
	Runway(runway16/34)	(46)
	designator(runway16/34, '16/34')	(47)
	isSituatedAt(runway16/34, airportLOWW)	(48)
K_4	contaminant(taxiway10/004, taxiway10/004-contam#101)	(49)
	depth(taxiway10/004-contam#101, 0.4)	(50)
	layer(taxiway10/004-contam#101, taxiway10/004-layer#101)	(51)
	contaminationType(taxiway10/004-layer#101, drySnow)	(52)
K_5	availability(runway16/34, runway16/34-avail#101)	(53)
	operationalStatus(runway16/34-avail#101, limited)	(54)
	usage(runway16/34-avail#101, runway16/34-usage#101)	(55)
	limitationType(runway16/34-usage#101, forbid)	(56)
	operation(runway16/34-usage#101, landing)	(57)
	aircraft(runway16/34-usage#101, characteristic#101)	(58)
	HeavyWakeCharacteristic(characteristic#101)	(59)
K_6	availability(taxiway10/004, taxiway10/004-avail#201)	(60)
	operationalStatus(taxiway10/004-avail#201, closed)	(61)
	annotation(taxiway10/004-avail#201, taxiway10/004-note#201)	(62)
	purpose(taxiway10/004-note#201, remark)	(63)
	propertyName(taxiway10/004-note#201, operationalStatus)	(64)
	note(taxiway10/004-note#201, dueToSnowRemoval)	(65)
K_7	availability(taxiway10/004, taxiway10/004-avail#301)	(66)
	warning(taxiway10/004-avail#301, inspection)	(67)
	warningAdjacent(taxiway10/004-avail#301, true)	(68)

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

ToSnowRemoval (Row 12); this example corresponds to the DNOTAM shown in Listing 1.

In some cases, a ManoeuvringAreaAvailability individual may indicate nothing more than a warning via the warning property (Row 13), which refers to a cause for caution, e.g., inspection (Row 14). The boolean warningAdjacent property (Row 15) indicates whether or not the warning refers to an area adjacent to the infrastructure element that the ManoeuvringAreaAvailability individual belongs to.

A ManoeuvringAreaAvailability individual may also indicate a usage restriction for a runway or taxiway, which forbids certain operations, e.g., landing, possibly conditional to certain aircraft characteristics. In that case, the usage property links a ManoeuvringAreaAvailability individual to a ManoeuvringAreaUsage individual (Row 16). The operation property, in turn, links a ManoeuvringAreaUsage individual to an OperationManoeuvring individual (Row 17), e.g., landing (Row 18). The limitationType property (Row 19) then indicates the type of limitation, e.g., forbid (Row 20), imposed for the previously specified operation. The aircraft property (Row 21) specifies characteristics of aircraft which the usage restriction applies to. HeavyWakeCharacteristic is a kind of AircraftCharacteristic (Row 22), referring to aircraft with heavy wake turbulence.

The contaminant property (Row 23) indicates SurfaceContamination for an infrastructure element, e.g., a runway or taxiway. A SurfaceContamination has an overall depth (Row 24) and several contamination layers, specified via the layer property linking a SurfaceContamination to SurfaceContaminationLayer individuals (Rows 25 and 26). A SurfaceContaminationLayer has a contaminationType property (Row 27), indicating the ContaminationType (Row 28) present on the surface, e.g., drySnow (Row 29). The drySnow contamination type represents a kind of snow (Row 30); the kindOf property defines a grouping of individuals.

A navigation aid (Navaid) may be used for homing at an AirportHeliport (Rows 31 and 32). Very High Frequency Omni-Directional Range (VOR) and Distance Measureing Equipment (DME) are special types of Navaid (Row 33), with VOR/DME representing navigation aids that act both as VOR and DME (Row 34). The frequency property (Row 35) indicates the frequency which a piece of equipment, e.g., a navigation aid, operates at.

The K_1 module (Rows 36-41) 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 Airport class

(Row 36) and vor/dmeFMD (VOR/DME in Fischamend near Vienna) as an individual of the VOR/DME class (Row 38) with designator 'FMD' (Row 39) that is used for homing at Vienna airport (Row 40). Furthermore, the module defines the HeavyWakeCharacteristic concept (Row 41) by maximum take-off weight, referring to aircraft that fall into the weightCategory of aircraft with a maximum take-off weight above 136 tonnes.

The K_2 module (Row 42) defines supplementary knowledge relevant for FixedWing aircraft in the LOVV region in February 2020. In this particular month, the vor/dmeFMD navigation aid operates at a frequency of 110.8 MHz.

The K_3 module (Rows 43-48) defines general knowledge relevant in the LOWW (Vienna airport) segment of the LOVV region in 2020. That knowledge consists 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 49-52) defines knowledge essential for aircraft of type FixedWing in the LOWW segment on 12 February 2020. On that day, taxiway10/004 was covered by a contaminant (Row 49) with a depth of 0.4 (Row 50) consisting of one contamination layer (Row 51) of drySnow (Row 52).

The K_5 module (Rows 53-59) defines knowledge of flight critical importance for A380 aircraft in the LOWW segment on 12 February 2020. On that day, runway16/34's availability (Row 53) indicates a limited operational status (Row 54) where usage is forbidden (Rows 55 and 56) for landing aircraft (Row 57) with heavy wake turbulence (Rows 58-59).

The K_6 module (Rows 60-65) defines knowledge about restrictions relevant for A380 aircraft in the LOWW segment on 12 February 2020. On that day, taxiway10/004's availability (Row 60) indicates a closed operational status (Row 61). An annotation (Rows 62-63) remarks that the change in operationalStatus (Row 64) is due to snow removal (Row 65).

The K_7 module (Rows 66-68) defines knowledge about potential hazards relevant for A380 aircraft in the LOWW segment on 12 February 2020. A warning notifies of an inspection adjacent to taxiway10/004. ◇

Example 4 illustrates how knowledge representation in KG-OLAP fulfills the functional requirements defined in the previous section. First, the represented KGs are heterogeneous (Requirement 1) regarding both the types of entities and relationships; schema variability is exemplified by the different variants of ManoeuvringAreaAvailability. The KGs comprise ontological knowledge (Requirement 2), mainly in K_0, which defines properties and classes, but notably also in K_1, which contains the definition of a business term, namely HeavyWakeCharacteristic, denoting the characteristic of aircraft with heavy wake turbulence. The contents are self-describing (Requirement 3), the schema information and ontological knowledge being a flexible part of the data. Knowledge is modularized (Requirement 4), namely along the context dimensions. The modules have different levels of generality (Requirement 5): K_0 comprises knowledge that is generally applicable, the other modules comprise gradually more specific knowledge. For example, K_1 comprises quite general knowledge for the LOVV region whereas K_2-K_7 have more limited scopes of applicability.

The knowledge from higher-level cells propagates to the covered lower-level cells; the knowledge associated with the lower-levels cells also specializes the more general knowledge inherited from the higher levels. The hierarchical 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. 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)**.** Consider the cells in Fig. 5 and the corresponding knowledge modules in Fig. 6. In the K_5 module, characteristic#101 is asserted to be a HeavyWakeCharacteristic (Fig. 6, Row 59). Using the knowledge inherited from K_0, it can be inferred that characteristic#101 is an AircraftCharacteristic (Row 22). Using the knowledge inherited from K_1, it can also be inferred that characteristic#101 has a weightCategory property with value above136t (Row 41). ◇

3.2. Formalization

In the following, we adapt and extend the definitions of the CKR framework – building on the CKR definition [18, 44] in a generic description logic (DL) language [45] – 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 $\mathbf{D} \subseteq \mathrm{NR}_\Omega$ 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 $\mathsf{E} \in \mathbf{D}$, a set $\mathsf{D}_\mathsf{E} \subseteq \mathsf{I}$ of dimension members of E (cf. dimensional structure in [16]). Dimensions are defined as atomic roles: The dimensions serve to link individual cells to dimension members that identify and qualify the situation (or context) described by the contents of the cell's knowledge module. 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 $\mathsf{A} \in \mathbf{D}$, we define the role $\prec_\mathsf{A}$ – the *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), which could be extended to support more complex hierarchies. 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 $\mathsf{A} \in \mathbf{D}$ 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, with $\mathsf{lev}(\text{12-02-2020}, \text{day})$, $\mathsf{lev}(\text{02-2020}, \text{month})$, $\mathsf{lev}(\text{2020}, \text{year})$, and $\mathsf{lev}(\text{All-date}, \text{all-date})$. 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 original CKR definition [16]. Let $|\mathbf{D}| = k$, we define a *dimensional vector* as the set

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

where every $d_j \in \mathsf{D}_{\mathsf{A}_j}$, with $1 \leqslant j \leqslant k$. The value given by $\mathbf{d}$ to the dimension A is denoted with d_A. For example, given the dimensional vector $\mathbf{d} = \{\mathsf{Importance} := \mathsf{All\text{-}importance}, \mathsf{Location} := \mathsf{LOVV}, \mathsf{Time} := \text{2020}, \mathsf{Aircraft} := \mathsf{All\text{-}aircraft}\}$, d_Location is equal to LOVV, with $\mathsf{LOVV} \in \mathsf{D}_\mathsf{Location}$. We refer to the set of all dimensional vectors of the cube vocabulary Ω as the *multidimensional space* $\mathfrak{D}_\Omega$.

Given a dimensional vector, we associate with that vector a *cell name* using the function $\mathsf{cn} : \mathfrak{D}_\Omega \to \mathbf{F}$. We require cn to be bijective, i.e., 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 $\mathbf{e} = \{\mathsf{Importance} := \mathsf{FlightCritical}, \mathsf{Location} := \mathsf{LOWW}, \mathsf{Time} := \text{12-02-2020}, \mathsf{Aircraft} := \text{A380}\}$, we have $\mathsf{cn}(\mathbf{e}) = \mathsf{c}_5$. We denote with cn^- the inverse function of cn.

Context coverage derives from the order of the dimensional vectors associated with the contexts, which in turn derives from the individual dimension orders. 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 $\mathsf{A} \in \mathbf{D}$. Similarly, given $\mathsf{c}_1, \mathsf{c}_2 \in \mathbf{F}$, 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 $\mathsf{A} \in \mathbf{D}$, $d_\mathsf{A} \preceq e_\mathsf{A}$. For example, given the dimension hierarchies from Fig. 4, for the dimensional vectors $\mathbf{d} = \{\mathsf{Importance} := \mathsf{All\text{-}importance}, \mathsf{Location} := \mathsf{LOVV}, \mathsf{Time} := \text{2020}, \mathsf{Aircraft} := \mathsf{All\text{-}aircraft}\}$ and $\mathbf{e} = \{\mathsf{Importance} := \mathsf{FlightCritical}, \mathsf{Location} := \mathsf{LOWW}, \mathsf{Time} := \text{12-02-2020}, \mathsf{Aircraft} := \text{A380}\}$, 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 the *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 the meta and object levels employ the same logic. Furthermore, we stress that the definitions are agnostic to the DL language chosen to express the knowledge inside modules, i.e. the expressiveness of operators used to combine atomic elements into complex concepts and roles.

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 of the CKR as presented by Bozzato and Serafini [18] – which has the advantage over the original CKR formulation [16] that it can be implemented as forward rules – to express propagation of knowledge modules along the cover-

age relations, and to allow contexts with empty knowledge contents. Further extending the definition of CKR in [18], we adopt the notion of the contextual structure being defined by contextual dimensions from the original CKR formulation [16], and we define knowledge propagation as inheritance of knowledge modules. We provide only basic definitions of the CKR core and refer to previous work [16, 18] 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 which contains *meta-knowledge*, i.e., the structure and properties of contexts, and *global (context-independent) knowledge*, i.e., knowledge that applies to every context; a CKR also consists of (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 then a 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 CKR is defined (cf. [18]) 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 the following we call $\mathfrak{K}$ a *KG-OLAP cube* (or simply *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 [45]. We then extend as follows the original definition of CKR model [18] 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* $\mathsf{A} \in \mathbf{D}$ *with* $d \in \mathsf{D}_\mathsf{A}$, $\mathcal{M} \models \mathsf{A}(\mathsf{c}_1, d)$ *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* $\mathsf{A} \in \mathbf{D}$ *with* $d_\mathsf{A} \in \mathsf{D}_\mathsf{A}$*;*

(vi). *if* $\mathsf{c}_1, \mathsf{c}_2 \in \mathbf{F}$*, if* $\mathcal{M} \models \mathsf{c}_1 \preceq \mathsf{c}_2$ *and* $\mathcal{M} \models \mathsf{mod}(\mathsf{c}_2, \mathsf{m})$ *with* $\mathsf{m} \in \mathbf{M}$*, 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 KG-OLAP Cubes

While the definitions for KG-OLAP cube and CKR are independent of any specific DL language used at the meta and object levels, we formalize instance-level reasoning inside a KG-OLAP cube using a *materialization calculus* (see [46]) for cubes that employ the $\mathcal{SROIQ}$-RL language, which corresponds to the OWL 2 RL profile [47]. We refer to Sect. 2 and Sect. 3 of the Appendix [26] for definitions of the $\mathcal{SROIQ}$-RL language and an extension to the KG-OLAP cube semantics of the materialization calculus for CKR [18], respectively.

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*). 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 (through *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 $\mathcal{SROIQ}$-RL CKRs (see [18]) it is necessary to introduce additional rules and translation steps in order to express computation of coverage relations and propagation of object knowledge. In particular, regarding translation rules, we introduce global input rules that encode coverage in the level and dimension hierarchies. The following global deduction rule encodes propagation of knowledge – corresponding to condition (vi) in Definition 3 – where gm denotes the context name of the global meta-knowledge.

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

Then, the translation procedure is extended from the one presented for CKR [18] by introducing new steps in which the cell coverage relation is computed from the dimensional coverage in the global program (see Sect. 3.3 of the Appendix [26]).

The rules and the translation process constitute a sound and complete calculus for instance checking in KG-OLAP cubes using $\mathcal{SROIQ}$-RL (Theorem 1). For the proof of Theorem 1 we refer to Sect. 3.4 of the Appendix [26].

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 $\mathfrak{K} \models \mathsf{c} : \alpha$. □

3.3. Extensions

The presented KG-OLAP model can be extended in several directions in order to increase flexibility. In the following, we briefly identify possible extensions.

As in the underlying model of CKR, the KG-OLAP model assumes a centralized view on the available data: The structure of the cube and the knowledge associated to each cell is assumed to be locally available in a single knowledge base for the application of reasoning and operations on the cube. Thus, a possible extension, supported by the modular nature of the cube model, concerns the distribution of the cube structure over separate knowledge bases. This would require to revise the formalization so that both the meta and object information are distributed across more local knowledge bases and knowledge is propagated also considering the topology of the distributed knowledge bases (e.g., by introducing an additional "distribution" dimension). The reasoning procedures and OLAP operations on such distributed system have to be consequently adapted to the distributed structure and the intended knowledge propagation across local knowledge bases. On the other hand, the modularization of knowledge bases can represent an advantage to limit the load of reasoning tasks and to restrict application of operations to a limited subset of cells.

In the proposed KG-OLAP model, we assume that all the dimensions and dimensional values are known a priori and thus no context is undefined (but may have an empty module). The KG-OLAP model could be extended to consider updates of dimensions and dimensional values, which necessitate the dynamic reorganization of knowledge in the contexts that are identified by the newly introduced dimensional values. Updates of the dimensions and their dimensional values will typically be limited to extending the hierarchy with new branches rather than modifying the existing di-

mensional values. In this regard, however, the notion of "slowly-changing dimensions" from traditional data warehousing (see [17, pp. 139-145] for further information) may be adopted in KG-OLAP. Furthermore, in some cases, it may not be possible to determine the right dimensional value for the statements. The KG-OLAP model allows for the introduction of "unknown" or "not applicable" members in the dimensions. For example, there could be a cell containing knowledge of unknown importance, a cell containing knowledge of unknown temporal or spatial applicability. Future work may investigate the different notions and implications of such default members in the dimension hierarchies.

We note that, due to the modular structure of the model, concepts in different cells can assume different interpretations: the particular situation identified by a cell can determine the local meaning of a concept, reflecting the contextual view of the model. In this regard, we refer to the *eval* operator for the local propagation of knowledge [18].

The concepts that are propagated from the higher to the lower cells are assumed to be refined by the knowledge in the lower cells. Such refinement is necessarily monotonic, as the knowledge in the lower-level cells cannot truly "override" the concept definitions from the higher-level cells. In this direction, in the CKR model, a notion of *defeasible axiom* has been recently introduced: Defeasible axioms hold in general but can be "overridden" by the instances in the lower contexts of a contextual hierarchy [44, 48, 49]. Likewise, defeasible axioms could be introduced in KG-OLAP.

For the presented definition of the KG-OLAP model we consider a simple hierarchical organization of dimensional values, which can be divided into levels, i.e., *ranked hierarchies* [48]. While this provides an intuitive organization of the cube structure, such organization might be relaxed to allow for multiple relations and more general definitions of hierarchies. In this regard, an extensive body of research in data warehousing and OLAP is dedicated to investigate different kinds of hierarchies (see [17, pp. 94-106]), which could be adapted for the KG-OLAP framework. Regarding the CKR model, propagation of knowledge (with defeasible inheritance) in general contextual hierarchies has been studied by Bozzato et al. [50].

4. Query Operations

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

4.1. Contextual Operations

The contextual operations select and combine cells of a KG-OLAP cube, thus satisfying Requirement 6, 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

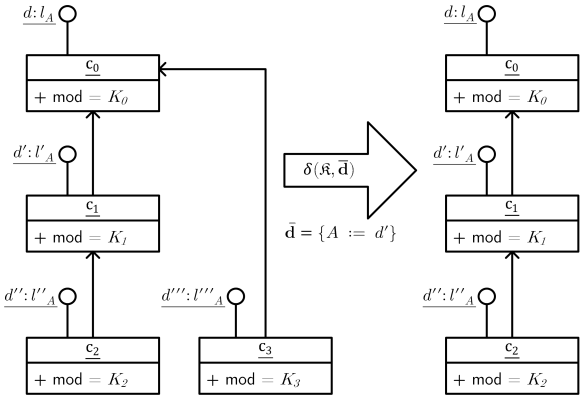
The *slice-and-dice* operation restricts a cube to a set of cells with a specific subset of dimension attribute values; the operation selects a subcube of an input KG-OLAP cube. The slice-and-dice operation selects a partition of the cube for subsequent manipulation. Note that slice-and-dice operations in data warehousing literature and practice come in various forms. 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 operation $\delta(\mathfrak{K}, \overline{\mathbf{d}})$ *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:*[2]

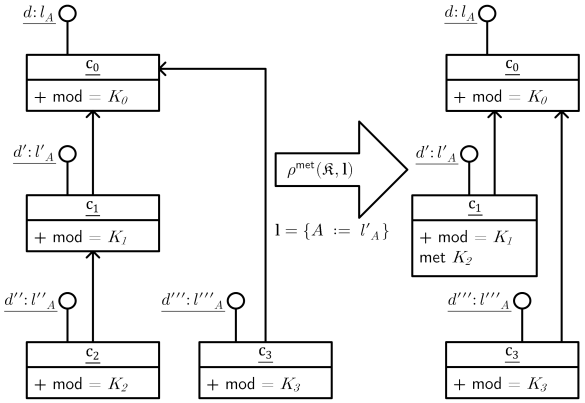
- $\mathbf{M}' = \mathbf{M}$, $\mathbf{D}' = \mathbf{D}$, *and for each* $\mathsf{A} \in \mathbf{D}$, $\mathsf{L}'_{\mathsf{A}} = \mathsf{L}_{\mathsf{A}}$*;*
- *For each* $\mathsf{A} \in \mathbf{D}$,
  $\mathsf{D}'_{\mathsf{A}} = \{\mathsf{d}'_{\mathsf{A}} \in \mathsf{D}_{\mathsf{A}} \mid \mathsf{d}'_{\mathsf{A}} \preceq \mathsf{d}_{\mathsf{A}}$ *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 under-

[2] 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.



(a) Slice/Dice



(b) Merge

Fig. 7. Illustration of contextual operations definitions

neath 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 $\{$Importance := Essential, Location := LOWW, Date := All-date, Aircraft := FixedWing$\}$. Only cells that are underneath the point identified by the dice coordinates, i.e., c_4, c_5, and c_6, or cells that are in a coverage relationship with c_4, c_5, and c_6, i.e., c_0, c_1, and c_3 are kept in the result cube $\mathfrak{K}'_{\mathsf{ATM}}$; the disregarded cells are shown in gray color. ◇

In data warehousing, conceptual multidimensional models typically distinguish between dimensional attributes and non-dimensional attributes (see [43]). While the dimensional attributes identify the cell, non-dimensional attributes provide additional information that can be used for selection. Similarly, KG-OLAP cubes could be extended with non-dimensional attributes to allow for additional variants of the slice-and-dice operation.

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* $A \in \mathbf{D}$, $\mathsf{D}'_\mathsf{A} = \{\mathsf{d}'_A \in \mathsf{D}_\mathsf{A} \mid \mathsf{lev}(\mathsf{d}'_A, \mathsf{l}_A), \mathsf{l}_A \in \mathsf{L}'_\mathsf{A}\}$*;*
- $\mathfrak{G}' = \mathfrak{G}_\Sigma \cup \mathfrak{G}_{\Gamma'} \cup \{\mathsf{mod}(\mathsf{c}, \mathsf{mg}(\mathsf{c})) \mid \mathsf{c} \in \mathbf{F}'$ *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:* $\mathrm{K}_{\mathsf{mg}(\mathsf{c})} = \bigcap_{\mathsf{m} \in \mu(\mathsf{c})} \mathrm{K}_\mathsf{m}$

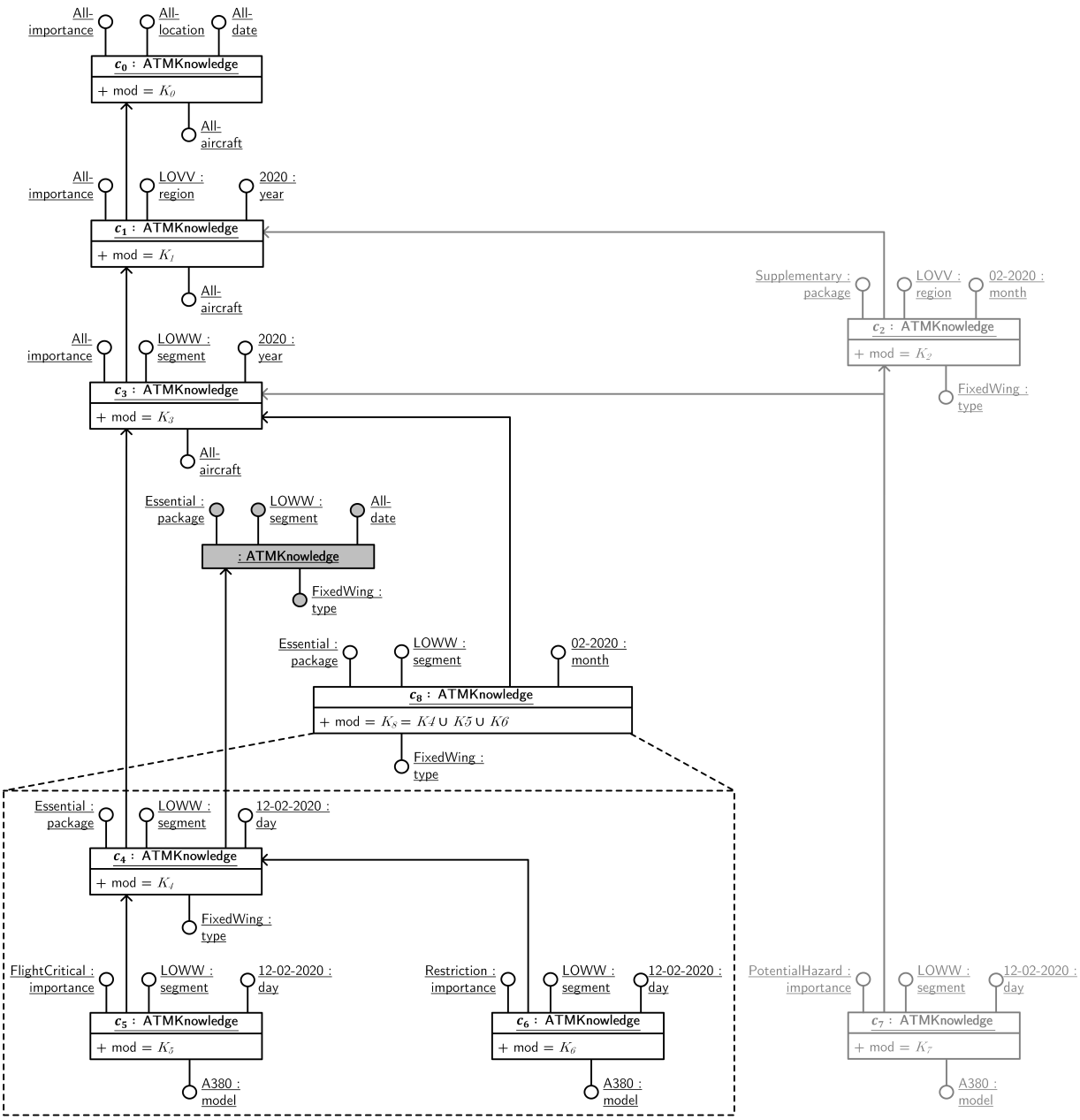


Fig. 8. Applying slice-and-dice and merge operations on the KG-OLAP cube instance from Fig. 5. Gray lines denote contexts that are disregarded by the slice-and-dice operation $\delta(\mathfrak{K}_{\mathsf{ATM}}, \{\mathsf{Importance} := \mathsf{Essential},\ \mathsf{Location} := \mathsf{LOWW},\ \mathsf{Date} := \mathsf{All\text{-}date},\ \mathsf{Aircraft} := \mathsf{FixedWing}\})$, 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.

Figure 7b illustrates the definition of the merge operation. 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 merge operation employs a specific combination method $\mathsf{met} \in \{\cup, \cap\}$, which specifies the kind of combination of knowledge inside the merged cells. Note that the merge operation changes the modules of the contexts at the specified roll-up granularity rather than creating new contexts, since a context is uniquely identified by its dimensional vector. Consequently, in Fig. 7b, the result of the merge operation shows c_1 with the combination of K_1 and K_2 as a single module.

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

In the case of general context coverage hierarchies that do not follow a clear notion of granularity level, the merge operation, which is now defined in terms of levels, requires reformulation. Moreover, the combination method in the merge operation – which in its current form simply considers either the union or intersection of the knowledge from the contexts in lower levels – can be refined to consider different methods for selection of knowledge in the newly generated module, e.g., by considering ontology merging methods that suit the use case at hand. We note that in the merge operation we do not explicitly consider the management of RDF blank nodes since we abstract from the serialization of the knowledge. We assume that blank nodes have been skolemized before the application of the merge operation; another option is to consider methods for merging RDF blank nodes in the resulting graph [51, #shared-blank-nodes-unions-and-merges].

4.2. Graph Operations

Graph operations – abstraction, pivoting, and reification – alter the structure of the KGs inside the knowledge modules of a cell, thus satisfying Requirement 7. 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:*

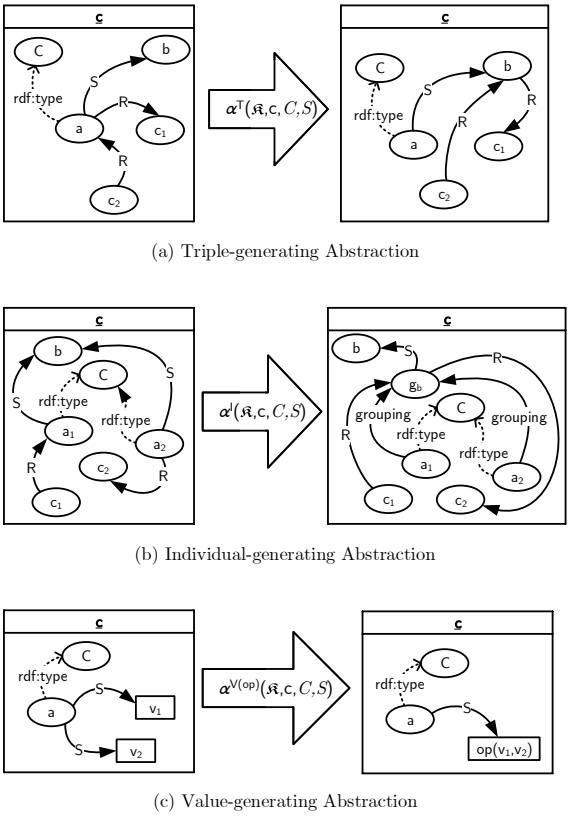
– *for every* $b \in S(a)$*, add* $\mathsf{grouping}(a, g_b)$ *to* $\mathrm{K}_{\mathsf{mg(c)}}$ *with* $g_b \in \mathrm{NI}_{\Sigma'}$ *a new individual name (associated with the grouping individual* b*);*
– *for every role assertion* $R(a,c) \in \mathrm{K}_{\mathsf{mg(c)}}$*, for every* $b \in S(a)$*, add* $R(g_b, c)$ *to* $\mathrm{K}_{\mathsf{mg(c)}}$ *and remove* $R(a,c)$ *from* $\mathrm{K}_{\mathsf{mg(c)}}$*;*
– *resp. for every* $R(c,a)$ *and* $C(a) \in \mathrm{K}_{\mathsf{mg(c)}}$*.*

– ***value generation*** $V(op)$***:*** *for* $a \in \mathrm{NI}_\Sigma$ *with* $\mathrm{K}_{\mathsf{mod(c)}} \models C(a)$*, considering the operation* op *on values in the range of* S*, let* $S(a) = \{v \in \mathrm{NI}_\Sigma \mid S(a,v) \in \mathrm{K}_{\mathsf{mg(c)}}\}$*, then:*
  – *add to* $\mathrm{K}_{\mathsf{mg(c)}}$ *the assertion* $S(a, op(v_1, \ldots, v_m))$ *with* $\{v_1, \ldots, v_m\} = S(a)$*;*
  – *remove every* $S(a,v) \in \mathrm{K}_{\mathsf{mg(c)}}$ *with* $v \in S(a)$*.*

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 a 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 [52]). 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 for definitions of the abstraction operations. Intuitively, the abstraction operation takes as input the single cell, on the knowledge module of which the operation is applied, a 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 $C(a)$, if there is some relation of the kind $S(a,b)$, i.e., a is "grouped" into 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 $S(a,b)$ then a new grouping individual g_b and assertion $\mathsf{grouping}(a, g_b)$ are added, and 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.

The definition of individual-generating abstraction can be easily extended to allow for multiple grouping



(a) Triple-generating Abstraction

(b) Individual-generating Abstraction

(c) Value-generating Abstraction

Fig. 9. Illustration of abstraction operations definitions

properties. For example, in individual-generating abstraction, given grouping relations S_1 and S_2, for every instance $C(a)$, if there is some relation of the kind $S_1(a,m)$ and some relation of the kind $S_2(a,n)$, a new grouping individual g_{mn} and assertion $\mathsf{grouping}(a, g_{mn})$ are added, and all of the ABox assertions of the kind $R(a,c), R(c,a)$ and $A(a)$ are redirected to g_{mn}. The definition of individual-generating abstraction could be defined as follows: instead of a single grouping property S, the individual-generating abstraction would have roles $S_1, \ldots, S_k$ as grouping properties. Then, for every $a \in \mathrm{NI}_\Sigma$ with $\mathrm{K}_{\mathsf{mod(c)}} \models C(a)$, and for every $j \in \{1, \ldots, k\}$, let $S_j(a) = \{b \in \mathrm{NI}_\Sigma \mid \mathrm{K}_{\mathsf{mod(c)}} \models S_j(a,b)\}$. For every $b \in S_1(a) \times \ldots \times S_k(a)$ add $\mathsf{grouping}(a, g_b)$ to $\mathrm{K}_{\mathsf{mg(c)}}$ with $g_b \in \mathrm{NI}_{\Sigma'}$ a new individual name (associated with the grouping tuple b). For every role assertion $R(a,c) \in \mathrm{K}_{\mathsf{mg(c)}}$, for every $b \in S_1(a) \times \ldots \times S_k(a)$ add $R(g_b, c)$ to $\mathrm{K}_{\mathsf{mg(c)}}$ and remove $R(a,c)$ from $\mathrm{K}_{\mathsf{mg(c)}}$; resp. for every $R(c,a)$ and $C(a) \in \mathrm{K}_{\mathsf{mg(c)}}$.

Example 8 (Abstraction)**.** Figure 10 illustrates the different variants of abstraction on the running example of

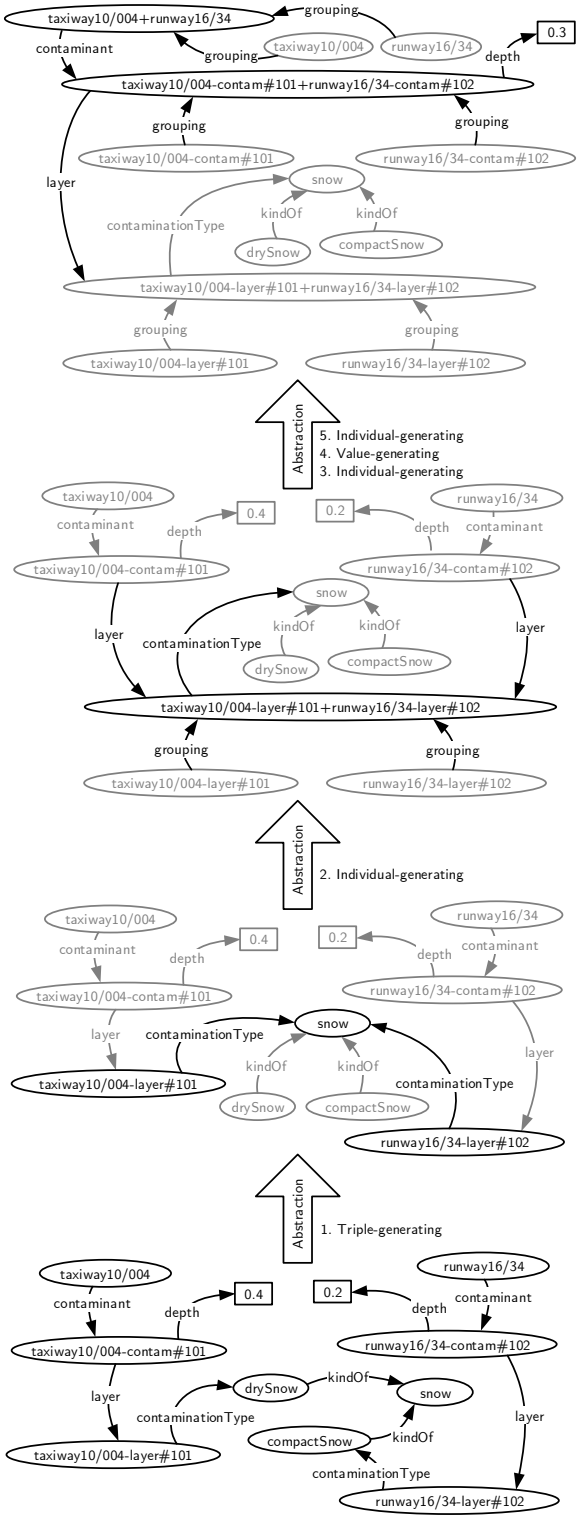


Fig. 10. Examples of triple-, individual-, and value-generating abstraction

ATM knowledge graphs. Take the K_4 module of the c_4 context of $\mathfrak{K}_{\text{ATM}}$ from Fig. 6, which contains an example of taxiway contamination. Assume that the K_4 module also contains a runway contamination with compactSnow, which is a kind of snow. The RDF graph at the bottom of Fig. 10 then illustrates those contents. First, a triple-generating abstraction $\alpha^T(\mathfrak{K}_{\text{ATM}}, c_4, \top, \mathsf{kindOf})$ leads to the replacement of individuals drySnow and compactSnow with snow, using kindOf as grouping property. Let $\mathfrak{K}'_{\text{ATM}}$ denote the cube resulting from the previous triple-generating abstraction. Second, over that cube, an individual-generating abstraction $\alpha^I(\mathfrak{K}'_{\text{ATM}}, c_4, \mathsf{SurfaceContaminationLayer}, \mathsf{contaminationType})$ then groups all SurfaceContaminationLayer 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. Let $\mathfrak{K}''_{\text{ATM}}$ denote the cube resulting from the individual-generating abstraction. Third, another individual-generating abstraction $\alpha^I(\mathfrak{K}''_{\text{ATM}}, c_4, \mathsf{SurfaceContamination}, \mathsf{layer})$ groups all SurfaceContamination individuals with the same layer property value. Let $\mathfrak{K}'''_{\text{ATM}}$ denote the cube resulting from that individual-generating abstraction. Fourth, a value-generating abstraction $\alpha^{V(avg)}(\mathfrak{K}'''_{\text{ATM}}, c_4, \mathsf{SurfaceContamination}, \mathsf{depth})$ replaces multiple depth property values (0.2 and 0.4) by the average depth (0.3). Let $\mathfrak{K}''''_{\text{ATM}}$ denote the cube resulting from that value-generating abstraction. A final individual-generating abstraction $\alpha^I(\mathfrak{K}''''_{\text{ATM}}, c_4, \mathsf{Runway} \sqcup \mathsf{Taxiway}, \mathsf{contaminant})$ then groups all Runway and Taxiway individuals with the same contaminant property value, which after the previous individual-generating abstraction means taxiway10/004 and runway16/34. In this case, a complex concept Runway $\sqcup$ Taxiway determines which individuals are candidates for abstraction. The result graph indicates, at an abstract level, the presence of snow contamination at runways and taxiways along with the average depth of contamination. $\diamond$

The grouping role S can be an inverse role as well as a composite (complex) role. For example, in order to perform an individual-generating abstraction that groups SurfaceContaminationLayer individuals by the infrastructure element they belong to, the composite role $\mathsf{layer}^- \circ \mathsf{contaminant}^-$ may serve as a grouping property. Similarly, a triple-generating abstraction could replace entities by another entity reached via a property path, e.g., using composite role $\mathsf{kindOf} \circ \mathsf{kindOf}$ to replace one entity by another two steps up the kindOf

hierarchy. SPARQL-like property paths with wildcard operators could also be introduced.

Additional variants of the abstraction operation can be defined in order to adapt the concept of abstraction to the structure of the object knowledge. For example, in individual-generating abstraction, we consider an explicit *grouping* relation to be introduced between grouped individuals and the newly introduced group individual. The grouping may also be realized with the introduction of a new class representing the abstraction, with class membership assuming the role of the grouping relation.

We note that KG-OLAP, in general, aims at being independent from the specific ontology language chosen for the representation of local axioms and assertions. If an ontology language were employed that could express the semantics of the abstraction operations (or some variant), that ontology language, in conjunction with an automated reasoner, could serve to implement the abstraction operations. Regardless, in the case of OWL, reasoning alone cannot serve to implement the abstraction operations. For example, in case of individual-generating abstraction, since OWL does not support grouping of individuals by property value when that value is a variable, extralogical steps must be followed in addition to reasoning. In Example 8, membership reasoning for the complex concept $\exists$contaminationType.{snow} could serve to obtain all the individuals which have the same value snow for the contaminationType property. In OWL, snow could not be replaced by a variable, though. In order to perform individual-generating abstraction, membership reasoning would have to be performed for multiple complex concepts, replacing the constant snow with other possible types of contamination. Suppose slush and ice are the other types of contamination. Then, in order to group layers by contamination type, membership reasoning for $\exists$contaminationType.{slush} and $\exists$contaminationType.{ice} would have to be performed as well. For each of those concepts, an individual would have to be created that represents the group of members that belong to the respective concept; alternatively, the concepts could be named and used as group individuals (OWL punning). Finally, each individual representing a group would replace the individuals belonging to that group in the KG.

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

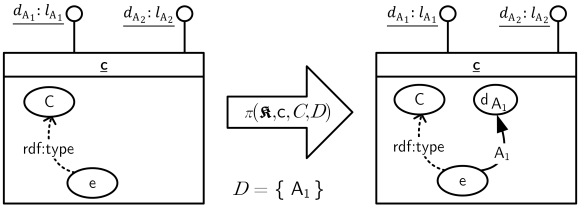


Fig. 11. Illustration of the pivoting operation definition

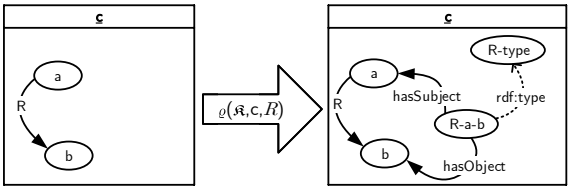


Fig. 12. Illustration of the reification operation definition

Definition 7 (Pivoting)**.** *Given a cube* $\mathfrak{K} = \langle \mathfrak{G}, \mathrm{K}_{\mathbf{M}} \rangle$*, a cell name* $c \in \mathbf{F}$*, 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 $\pi(\mathfrak{K}, c, C, D)$ *as a new cube* $\mathfrak{K}' = \langle \mathfrak{G}', \mathrm{K}_{\mathbf{M}'} \rangle$ *over* $\langle \Gamma', \Sigma' \rangle$ *s.t.*

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

Note that we have to admit that $\Sigma \cap \Gamma \neq \emptyset$ in order to use metaknowledge symbols in the local object knowledge. Figure 11 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 well as a set of dimensions D. The operation associates with c an additional knowledge module $\mathsf{mg}(c)$ 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 9 (Pivoting)**.** Consider the K_4 module of the c_4 context from Fig. 6. 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 $\mathsf{Importance}(\mathsf{taxiway10/004\text{-}contam\#101}, \mathsf{Essential})$ and $\mathsf{Date}(\mathsf{taxiway10/004\text{-}contam\#101}, \mathsf{12\text{-}02\text{-}2020})$. ◇

4.2.3. Reification

The *reification* operation takes "triples" (instance-level assertions) 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, which would otherwise be lost.

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* $R\text{-}\mathsf{type} \in \mathrm{NC}'_\Sigma$ *(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* $R\text{-}a\text{-}b$ *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):*

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

Figure 12 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 10 (Reification)**.** Consider the K_4 knowledge module of the c_4 context from Fig. 6. 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 $\mathsf{hasSubject}(\mathsf{depth\text{-}taxiway10/004\text{-}contam\#101\text{-}0.2}, \mathsf{taxiway10/004\text{-}contam\#101})$, $\mathsf{hasObject}(\mathsf{depth\text{-}taxiway10/004\text{-}contam\#101\text{-}0.2}, \mathsf{0.2})$, and $\mathsf{depth\text{-}type}(\mathsf{depth\text{-}taxiway10/004\text{-}contam\#101\text{-}0.2})$. ◇

5. Proof-of-Concept Prototype

In this section we sketch the foundations of a proof-of-concept prototype of a KG-OLAP system implemented on top of off-the-shelf quad stores. We evaluate the prototype's performance in order to grasp the complexity of KG-OLAP cube maintenance and query operations as basis for future optimization efforts; performance optimization is not a goal of this paper. We refer to Sect. 4 and Sect. 5 of the Appendix [26] for additional information on implementation and performance evaluation, respectively. Source code and logs from the experiments are available in an online repository[3].

5.1. Design and Implementation

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. Details about logical model, reasoning and queries are provided in Sect. 4 of the Appendix [26].

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.

Logical 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. The two-layered structure of the KG-OLAP system with a global context and multiple local contexts – as in the CKR core [18] – 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.

Reasoning. The reasoning procedure presented in Section 3.2.3, analogously to the CKR core [18], can be implemented using SPARQL-based forward rules that materialize the inferences, including coverage relationships between contexts. The KG-OLAP implementa-

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

tion employs the *RDFpro* framework [53] for rule execution. RDFpro supports the specification of SPARQL-based rules over multiple graphs, a feature required for reasoning inside individual cells as well as across different cells.

Queries. The query operations introduced in Section 4 can be implemented using SPARQL queries. In particular, the query operations translate into 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.

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. 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.

The performance experiments employed synthetic datasets based on the ATM scenario used throughout the paper, which allowed us to vary the number of dimensions, contexts, and statements while keeping the graphs similar. The generated contexts contain knowledge about airports, runways and taxiways, warnings, temporary closures of runways and taxiways for different aircraft characteristics based on weight or wingspan, layers of surface contamination with the depth of each layer, and navigation aids with their frequencies. Query operations ran on three-dimensional and four-dimensional datasets with 1 365, 2 501, and 3 906 contexts, respectively. The contexts were not all on the same granularity but distributed over five different granularity levels (not including the root context), where the number of contexts per granularity level gradually increased with the depth; the majority of the contexts were on the most specific granularity level (see Sect. 5.2 of the Appendix [26] for details). Some entities were shared between contexts at different granularities to accurately study the effect of knowledge propagation. Context size was varied by adding/removing branches in the context hierarchy. In turn, for each dimensionality and context size there were three different repository sizes. In order to make for more realistic datasets, repository size was varied by increasing the number of airports, runways and taxiways, warnings, etc. per context, dependent on the granularity level, not by randomly distributing statements across contexts, which due to the structure of the context hierarchy results in differences in the repository sizes between context sizes. Additional "baseline" datasets were used in the experiments involving abstraction operations, which consisted of a single context, in order to investigate the impact of contextualization on query processing.

We remark that designing performance experiments after existing benchmarks for traditional OLAP, e.g., TCP-DS[4], would not give an accurate idea of the inherent complexity of KG-OLAP cube maintenance and query operations due to the different nature of the involved data and the queries. Traditional OLAP operates on numeric data, and does not fulfill Requirements 1-3 as well as Requirement 5 described in Sect. 2.5. While it would be possible to model numeric data using RDF and analyze those data using SPARQL, it would be beside the point to measure performance of atypical uses of KG-OLAP. Furthermore, related work on OLAP and information networks that aims at performance optimization, e.g., [54–56], does not (wholly) fulfill the requirements described in Sect. 2.5 (see also Sect. 6.3), rendering those approaches incomparable to KG-OLAP in terms of performance. Related work on ATM KGs conducting performance evaluation [38] runs general SPARQL queries over large RDF datasets but does not feature modularization (Requirement 4). Those SPARQL queries cannot be reasonably adapted to queries over multiple modules, and running SPARQL queries over a single context would only serve to replicate the results of related work, which depend on the experimental setting and require the same datasets to be used, which are not public. Finally, in this paper, we do not aim at performance optimization. Rather, we investigate complexity of KG-OLAP cube maintenance and query operations to inform future optimization efforts.

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[5] 8.9 instance. The Java Virtual Machine (JVM) of the GraphDB instance ran with 100 GB heap

[4] http://www.tpc.org/tpcds/
[5] http://graphdb.ontotext.com/

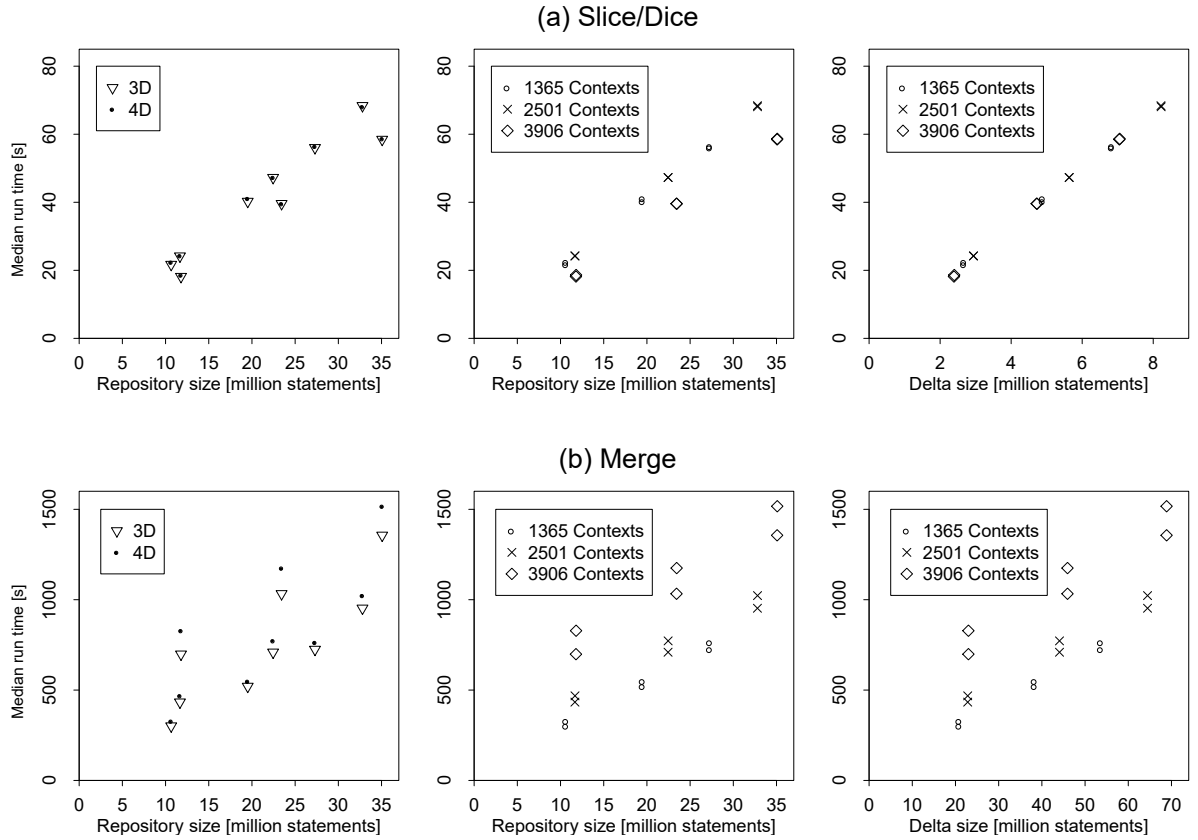


Fig. 13. Performance of contextual operations

space. The JVM running the KG-OLAP system, which conducts rule evaluation, prepares queries, and caches query results, ran with 20 GB heap space.

The GraphDB instance comprised two repositories – base and temporary – with the following configuration (see [57] 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 13a 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 because of the distribution of the statements across the contexts. Dimensionality does not play a role here. In summary, slice and dice does not involve any complex computations, consisting only of the selection of relevant contexts and their contents.

In case of the merge operation, in order to study worst-case performance, we deliberately chose dataset and formulation of the merge operation such that the result would be an almost complete reorganization of the contexts in the KG-OLAP cube. The lower-level cells contain the majority of statements, which are all affected by the merge operation. The complexity of the chosen operation is evidenced by the delta size, which is about twice the repository size: the merge operation first removes the statements from the merged cells only to insert those statements again into higher-level cells.

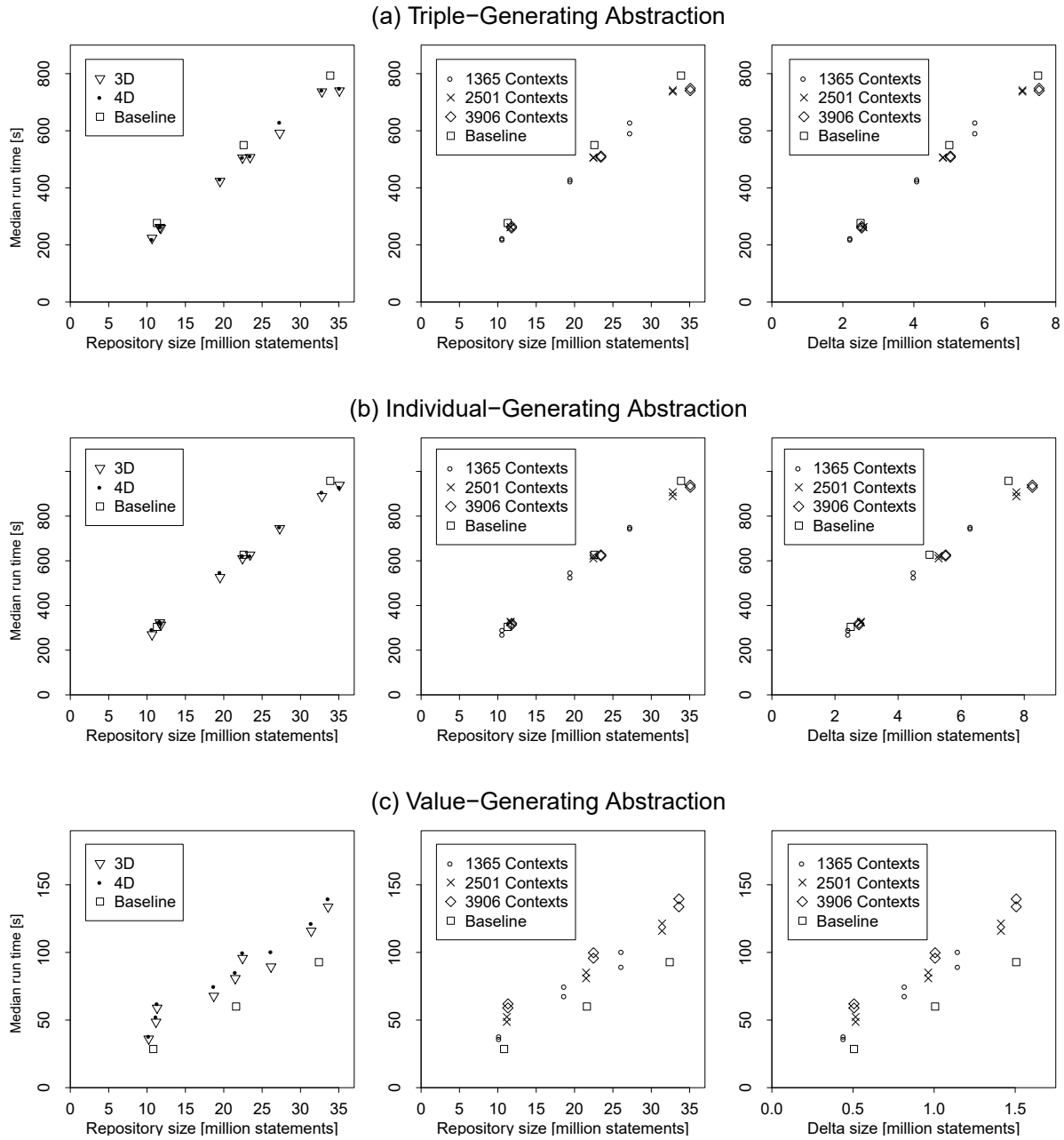


Fig. 14. Performance of abstraction operations

The run time of the merge union operation (Figure 13b) 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 performance evaluation of the abstraction operations, we employ, on the one hand, 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 of contextualization. Such abstraction operations on a single context correspond to the formalization. On the other hand, we perform abstraction operations in a variant that performs the abstraction 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 14a 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 14b). 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 14c 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 Sect. 4.5.6 and Sect. 4.5.7, respectively, of the Appendix [26]. 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.

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 15 shows run times for the evaluation of different rulesets relative to the repository sizes before and after reasoning – input and output repository size, respectively – as well as number of contexts for three-dimensional and four-dimensional KG-OLAP cubes. Reasoning in the performance experiments was conducted with the entire repository loaded into main memory first. Performance of two different rulesets was evaluated. The first ruleset, at the local level, i.e., within each context, performs membership reasoning under consideration of subclass relationships. For example, if Runway is a subclass of RunwayTaxiway and the fact Runway(runway16/34) is known, then RunwayTaxiway(runway16/34) is inferred. The second ruleset considers subclass relationships as well as domain and range of the properties. For example, if AirportHeliport is the range of the isSituatedAt property, and isSituatedAt(runway16/34, airportLOWW) is known, then AirportHeliport(airportLOWW) can be inferred. Figure 15a shows the results for the first ruleset, Fig. 15b for the second. Note that the datasets were different regarding class membership assertions: most membership assertions were omitted in the input dataset for domain/range reasoning and left to be inferred during the reasoning process. Domain/range reasoning had significant implications on the runtime of rule evaluation. In both cases, however, the main factor influencing run time turned out to be the number of contexts. The reasoning task was complicated by the fact that the relevant knowledge for reasoning was distributed over multiple modules, evidenced by the fact that rule evaluation without any form of local inferencing takes only a couple of seconds. We leave the investigation of complexity and efficient implementation of reasoning in a contextualized setting under different ontology languages to future work.

5.3. Discussion

The aim of the implementation was to get an idea of the complexity of KG-OLAP cube maintenance and query operations. Even though optimization was not a main goal of this paper, performance is already more than satisfactory for certain use cases. In order to handle big KGs, however, future work will consider a distributed, parallelized implementation. Nevertheless, optimization requires a thorough definition of the conceptual fundamentals, which is the aim of this paper.

In the following, we briefly discuss the implications of the results of the performance evaluation regarding the use cases presented in Sect. 2.

Use Case 1 (Pilot briefings)**.** Performance of the proof-of-concept prototype is more than satisfactory for use in pilot briefings. In that case, a KG-OLAP cube would cover the pilot briefing for a particular flight. Knowledge would consist, on the one hand, of relevant baseline information, e.g., available runways and taxiways,

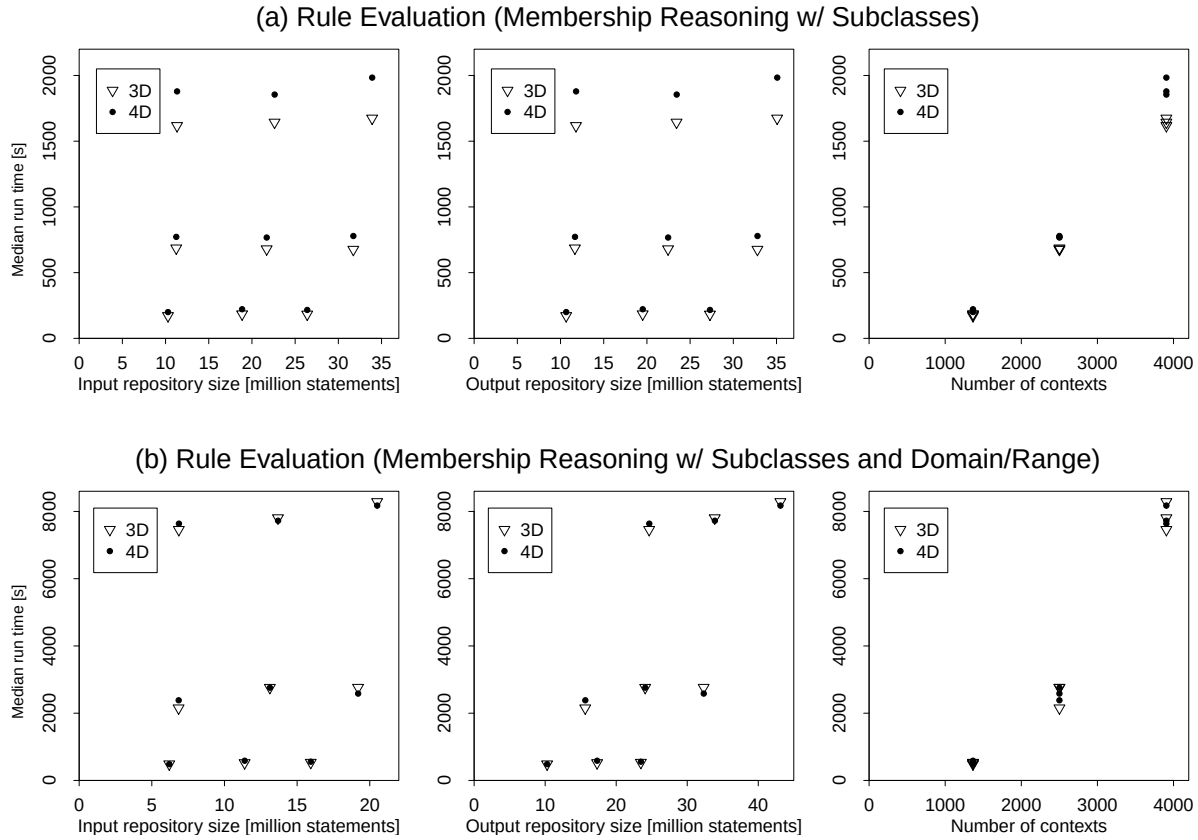


Fig. 15. Performance of rule evaluation

route of the flight, and navigation aids. On the other hand, temporary knowledge would be extracted from DNOTAMs and METARs. A typical short-distance flight has fewer than 200 relevant DNOTAMs [20, p. 6B2-10]. Assuming that it takes 10 triples to represent a single DNOTAM, the knowledge extracted from DNOTAMs amounts to 2 000 triples for a short-distance flight. Even when accounting for other potentially relevant knowledge, and assuming a long-distance flight, the KG for a pilot briefing for a single flight will be in the range of at most 100 000 triples, which is well within the capabilities of the proof-of-concept prototype and would also allow for more sophisticated reasoning. The expected number of contexts for a single flight, even with hourly or half-hourly granularity of contextualization, assuming tens of geographic segments, is within the capabilities of the prototype implementation.

Use Case 2 (Post-operational analysis)**.** Maintaining a large KG-OLAP cube for post-operational analysis in Europe or the United States, or even in a single region within one of those areas, with possibly tens of thousands of contexts and billions of triples, would require a different implementation strategy. Taking the figures regarding ATMGRAPH's size as reference, the volume of knowledge for only the New York area generated within a single month amounts to 260 million triples [38]. An alternative to maintaining a single KG-OLAP cube would be the adoption of a federated approach, maintaining multiple cubes within a *metacube* [25], i.e., a cube of cubes. A drill-across operation would then serve to combine knowledge from different cubes. Processing would have to be distributed and parallelized on multiple worker nodes.

6. Related Work

In this section we review related work on contextualization in KGs, on OLAP and semantic web technologies, on OLAP and information networks, and on (knowledge) graph summarization.

6.1. Contextualized Knowledge Graphs

The choice of CKR as base formalism for the definition of the KG-OLAP cube model was motivated by the explicit representation of dimensions and the organization of dimensions into levels which, as discussed in the previous sections, is compatible with the multidimensional perspective of OLAP. The idea of associating contexts with dimensions traces back to the theoretical works of Lenat [58] and the "context as a box" paradigm from Benerecetti et al. [59] for representation of contexts in knowledge representation.

The problem of annotation and contextualization of knowledge in KGs has been present since the early years of semantic web technologies. Different ways of introducing a notion of modularization in the RDF data model or KG implementations have been proposed [60–62]. Such approaches, however, often do not provide a formal (logic-based) definition of the resulting contextual model, which goes against Requirement 2 (ontological knowledge).

The need for formalization of the notion of context in KGs led to the proposal of several (description) logic-based approaches for the contextualization of knowledge bases: apart from the CKR [16, 18], we can cite [63–66], for example. While these models explicitly introduce contexts as a primitive in knowledge bases, these proposals do not always take into account an explicit representation of the notion of dimensions and/or level hierarchies, which is important – as of Requirement 4 for modularization and Requirement 5 for hierarchical decomposition of KGs into more general and more specific knowledge along with knowledge propagation – to qualify the different situations represented by contexts, and to have a clear separation of knowledge bases associated with each context.

Krötzsch et al. [67] propose an approach for adding annotations in a logic-based reading of KGs. While that model formally introduces annotations to local description logic axioms and assertions, it does not provide definite criteria for knowledge propagation across contextual structures as in the KG-OLAP coverage hierarchy (cf. Requirement 5, general and specific knowledge).

6.2. OLAP and Semantic Web Technologies

Semantic technologies have been used for a variety of tasks in the context of OLAP (see [68] for an overview). Related to KG-OLAP are techniques for data analysis over RDF data. The RDF data cube vocabulary (QB) [69] and its extension, QB4OLAP [70], 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 [71] and QB4OLAP [72, 73]. Such statistical linked data are just different serialization and publication formats of traditional OLAP cubes. Other work has suggested "lenses" over RDF data [74] 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 [75] 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 [76] 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 [77] 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 Semantic Cockpit project [78] 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 have some similarities to structured data lakes (see [79] for more information on data lakes), which stores the data of interest in a semantically richer format than plain numeric measures, but unlike conventional data lakes, which store raw data, provides a certain degree of integration and cleaning as well as dedicated query operations.

6.3. OLAP and Information Networks

KG-OLAP is related to applications of OLAP technology for analysis of information networks. Among the first, and arguably the most prominent, of these approaches was Graph OLAP (also known as InfoNet-OLAP) [80, 81], 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, i.e., the contents of the cube cells are graphs.

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. The focus of Graph OLAP are weighted directed graphs with highly structured and homogeneous data. Hence, Graph OLAP does not consider heterogeneity (Requirement 1) and ontological knowledge (Requirement 2). Schema information, e.g., about roll-up hierarchies, are external to the data, i.e., graphs in Graph OLAP are not self-describing (Requirement 3), resulting in rigid, inflexible graph schema and queries. Graph OLAP also does not consider the systematic propagation of knowledge from more general to more specific contexts (Requirement 5); graph data are only associated with the most specific granularity in the cube.

Since the original proposal of Graph OLAP, other approaches that apply OLAP to information networks have been proposed, which can be compared along different criteria, most notably the type of network, i.e., homogeneous or heterogeneous [82]. KGs have been likened to "schema-rich" heterogeneous information networks [83] and, therefore, only approaches applying OLAP to heterogeneous networks can be accurately compared to KG-OLAP. Yet, even approaches applying OLAP technology to heterogeneous information networks, e.g., [54–56, 84, 85], do not explicitly consider schema variability (Requirement 1c) or ontological knowledge (Requirement 2). Furthermore, unlike KG-OLAP, approaches to OLAP on information networks also lack a systematic way of structuring large bodies of knowledge, which comes with the decomposition of large knowledge graphs into hierarchically structured modules, from more general to more specific, in conjunction with knowledge propagation (Requirement 5). Schema and instance data are also firmly separated, i.e., the data are not self-describing (Requirement 3), reducing flexibility of the query operations. Another approach [86] aims to reconcile statistical/-multidimensional linked data in QB and QB4OLAP with the informational/topological view of OLAP on information networks. With respect to the requirements, that approach does not consider schema variability (Requirement 1c), ontological knowledge (Requirement 2), distinction between more general and more specific knowledge in conjunction with knowledge propagation (Requirement 5), and knowledge abstraction (Requirement 7). Some approaches, e.g., [56], offer abstraction operations that are similar to the KG-OLAP operations, but less flexible due to lack of self-describing data (Requirement 3), which allow for a more flexible formulation of roll-up paths within the graph structure. Furthermore, KG-OLAP allows for the use of complex concepts and roles in abstraction (in line with Requirement 2, ontological knowledge), which allows for flexible query formulation. For example, individual-generating abstraction with complex concepts such as $\mathsf{Runway} \sqcup \mathsf{Taxiway}$ as selection criteria and complex roles such as $\mathsf{layer}^- \circ \mathsf{contaminant}^-$ as the grouping properties would not be possible in OLAP on information networks.

Techniques for optimization of OLAP on information networks may inform the optimization of KG-OLAP. Precomputation and materialization of aggregate views [55, 56] is the most prevalent optimization technique for OLAP on information networks, which may not be applicable directly to KG-OLAP due to the heterogeneous data and highly flexible query formulation in KG-OLAP. Adopting a MapReduce-based implementation like Pagrol's [55] or Process OLAP's [87] is an interesting perspective for KG-OLAP as well.

Work on the analysis (or mining) of heterogeneous information networks [88] and Semantic Web databases [89] is orthogonal to KG-OLAP. Such approaches calculate, for example, degree distribution or PageRank, but may also serve to predict links and as the basis for recommender systems. Just like data mining may use OLAP cubes, information network mining may use KG-OLAP cubes.

6.4. (Knowledge) Graph Summarization

The KG-OLAP query operations also invite comparison with *(knowledge) graph summarization* techniques [90, 91], 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 [91]. 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 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* [91]. 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 modularization and contextuality in KGs (Requirement 4). 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 the analysis of knowledge represented in KGs. We extended the multidimensional modeling paradigm from OLAP to the representation of contextualized KGs. Each cell/context in a KG-OLAP cube contains knowledge triples. We introduced specific query operations for KG-OLAP cubes. On the one hand, contextual operations allow for selecting and merging contexts. On the other hand, graph operations allow for summarizing knowledge triples within individual contexts. We illustrated KG-OLAP with use cases in air traffic management [24, 25]. A proof-of-concept prototype using off-the-shelf quad stores and SPARQL queries demonstrates feasibility. Using the prototype, we conducted experimental performance evaluation, which may inform future optimization efforts. Even though optimization was not a goal, we conclude that there is a class of use cases for which the proof-of-concept implementation's performance is more than satisfactory.

Different directions can be investigated for future work. We aim at extending the formalism for a distributed approach with federated KG-OLAP cubes. We sketch a first version of a distributed, parallelized KG-OLAP framework in [92]. Regarding implementation, we aim at realizing a distributed, parallelized implementation of a KG-OLAP system including the concept of metacube and the drill-across operation [25], in order to support big KGs. We are interested in extending the KG-OLAP model with defeasible axioms similar to previous work on CKR [44, 48]. Regarding the KG-OLAP operations, different variants can be studied to meet the needs of the use cases at hand and the extensions of the underlying model (e.g. by distribution). Moreover, we may refine the current definitions of operators with common RDF summarization techniques.

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, Cham, 2017, pp. 1–14. doi:10.1007/978-3-319-45654-6_1.

[3] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic Web* **8**(3) (2017), 489–508. doi:10.3233/SW-160218.

[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)*, C. Sierra, ed., 2017, pp. 2–10. doi:10.24963/ijcai.2017/1.

[5] R. Cyganiak, J.J. Carroll and B. McBride, RDF 1.1 Concepts and Abstract Syntax – W3C Recommendation 25 February 2014, Technical Report, W3C, 2014. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[6] 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. http://sites.computer.org/debull/A16sept/p5.pdf.

[7] 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.

[8] 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. ESWC 2011*, G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. De Leenheer and J. Pan, eds, Lecture Notes in Computer Science, Vol. 9981, Springer, Berlin, Heidelberg, 2011, pp. 154–168. doi:10.1007/978-3-642-21064-8_11.

[9] T. Ruan, L. Xue, H. Wang, F. Hu, L. Zhao and J. Ding, Building and Exploring an Enterprise Knowledge Graph for Investment Analysis, in: *The Semantic Web – ISWC 2016. ISWC 2016*, P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck and Y. Gil, eds, Lecture Notes in Computer Science, Vol. 9982, Springer, Cham, 2016, pp. 418–436. doi:10.1007/978-3-319-46547-0_35.

[10] Google, Introducing the Knowledge Graph: things, not strings, 2012, https://search.googleblog.com/2012/05/introducing-knowledge-graph-things-not.html (Accessed: 20 October 2020).

[11] R. Qian, Understand your world with Bing, 2013, https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/ (Accessed: 20 October 2020).

[12] 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. doi:10.3233/SW-140134.

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

[14] 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, Cham, 2017, pp. 147–180. doi:10.1007/978-3-319-45654-6_6.

[15] 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: *Joint Proceedings of the International Workshops on Contextualized Knowledge Graphs, and Semantic Statistics*, S. Capadisli, F. Cotton, J.M. Giménez-García, A. Haller, E. Kalampokis, V. Nguyen, A.P. Sheth and R. Troncy, eds, CEUR Workshop Proceedings, Vol. 2317, CEUR-WS.org, 2018. http://ceur-ws.org/Vol-2317/article-10.pdf.

[16] L. Serafini and M. Homola, Contextualized Knowledge Repositories for the Semantic Web, *Journal of Web Semantics* **12** (2012), 64–87. doi:10.1016/j.websem.2011.12.003.

[17] A. Vaisman and E. Zimányi, *Data Warehouse Systems – Design and Implementation*, Springer, Berlin Heidelberg, 2014. doi:10.1007/978-3-642-54655-6.

[18] L. Bozzato and L. Serafini, Materialization calculus for contexts in the Semantic Web, in: *DL 2013*, T. Eiter, B. Glimm, Y. Kazakov and M. Krötzsch, eds, CEUR Workshop Proceedings, Vol. 1014, CEUR-WS.org, 2013. http://ceur-ws.org/Vol-1014/paper_51.pdf.

[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] 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.

[21] 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.

[22] 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.

[23] 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, Lecture Notes in Computer Science, Vol. 7908, Springer, 2013, pp. 514–529. doi:10.1007/978-3-642-38709-8.

[24] 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.

[25] 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* **123**(1268) (2019), 1639–1665. doi:10.1017/aer.2019.74.

[26] C.G. Schuetz, L. Bozzato, B. Neumayr, M. Schrefl and L. Serafini, Knowledge Graph OLAP: Appendix, Technical Report, 2020. http://kg-olap.dke.uni-linz.ac.at/appendix.

[27] Skybrary, Situational Awareness, https://www.skybrary.aero/index.php/Situational_Awareness (Accessed: 20 October 2020).

[28] International Civil Aviation Organization, *Annex 15 to the Convention on International Civil Aviation: aeronautical information services*, 14 edn, 2013, Accessed: 29 October 2020. https://www.icao.int/NACC/Documents/Meetings/2014/ECARAIM/REF05-Annex15.pdf.

[29] Federal Aviation Administration, *Federal NOTAM system airport operations scenarios*, 2010, Accessed: 20 October 2020. https://notams.aim.faa.gov/FNSAirportOpsScenarios.pdf.

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

[31] R.M. Keller, Ontologies for aviation data management, in: *Proceedings of the IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 2016. doi:10.1109/DASC.2016.7777971.

[32] E. Gringinger, R.M. Keller, A. Vennesland, C.G. Schuetz and B. Neumayr, A Comparative Study of Two Complex Ontologies in Air Traffic Management, in: *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019. doi:10.1109/DASC43569.2019.9081790.

[33] A. Vennesland, R.M. Keller, C.G. Schuetz, E. Gringinger and B. Neumayr, Matching Ontologies for Air Traffic Management: a Comparison and Reference Alignment of the AIRM and NASA ATM Ontologies, in: *Proceedings of the 14th International Workshop on Ontology Matching*, P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh and C. Trojahn, eds, CEUR Workshop Proceedings, Vol. 2536, CEUR-WS.org, 2019. http://ceur-ws.org/Vol-2536/om2019_LTpaper1.pdf.

[34] R.M. Keller, The NASA Air Traffic Management Ontology (atmonto) – Release dated March 2018, Technical Report, National Aeronautics and Space Administration, 2018, Accessed: 20 October 2020. https://data.nasa.gov/ontologies/atmonto/.

[35] 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.

[36] R.M. Keller, The NASA Air Traffic Management Ontology (atmontoPlus) – Release dated March 2018, Technical Report, National Aeronautics and Space Administration, 2018,

Accessed: 20 October 2020. https://data.nasa.gov/ontologies/atmontoPlus/.

[37] M.M. Eshow, M. Lui and S. Ranjan, Architecture and capabilities of a data warehouse for ATM research, in: *2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*, 2014. doi:10.1109/DASC.2014.6979418.

[38] R.M. Keller, S. Ranjan, M.Y. Wei and M.M. Eshow, Semantic representation and scale-up of integrated air traffic management data, in: *Proceedings of the International Workshop on Semantic Big Data*, S. Groppe and L. Gruenwald, eds, 2016. doi:10.1145/2928294.2928296.

[39] AIXM 5.1.1 - Data Model (UML), Accessed: 20 October 2020. http://aixm.aero/document/aixm-511-data-model-uml.

[40] F. Burgstaller, D. Steiner, B. Neumayr, M. Schrefl and E. Gringinger, Using a model-driven, knowledge-based approach to cope with complexity in filtering of notices to airmen, in: *Proceedings of the Australasian Computer Science Week Multiconference*, 2016, p. 46. doi:10.1145/2843043.2843044.

[41] F. Burgstaller, D. Steiner, M. Schrefl, E. Gringinger, S. Wilson and S. van der Stricht, AIRM-based, fine-grained semantic filtering of notices to airmen, in: *Proccedings of the 15th Integrated Communication, Navigation and Surveillance Conference (ICNS) Conference*, 2015. doi:10.1109/ICNSURV.2015.7121222.

[42] S. Niarchakou and J. Simón Selva, *ATFCM operations manual – network operations handbook*, 21.0 edn, 2017, Accessed: 20 October 2020. http://www.eurocontrol.int/sites/default/files/content/documents/nm/network-operations/HANDBOOK/ATFCM-Operations-Manual-next.pdf.

[43] 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. doi:10.1142/S0218843098000118.

[44] L. Bozzato, T. Eiter and L. Serafini, Enhancing context knowledge repositories with justifiable exceptions, *Artificial Intelligence* **257** (2018), 72–126. doi:10.1016/j.artint.2017.12.005.

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

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

[47] 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/.

[48] 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.

[49] L. Bozzato, T. Eiter and L. Serafini, Reasoning with Justifiable Exceptions in $\mathcal{EL}_\bot$ Contextualized Knowledge Repositories, in: *Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, C. Lutz, U. Sattler, C. Tinelli, A. Turhan and F. Wolter, eds, Lecture Notes in Computer Science, Vol. 11560, Springer, Cham, 2019, pp. 110–134. doi:10.1007/978-3-030-22102-7_5.

[50] L. Bozzato, T. Eiter and L. Serafini, Justifiable Exceptions in General Contextual Hierarchies, in: *Modeling and Using Context. CONTEXT 2019*, G. Bella and P. Bouquet, eds, Lecture Notes in Computer Science, Vol. 11939, Springer, Cham, 2019, pp. 26–39. doi:10.1007/978-3-030-34974-5_3.

[51] P.J. Hayes and P.F. Patel-Schneider, RDF 1.1 Semantics – W3C Recommendation 25 February 2014, Technical Report, W3C, 2014. https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.

[52] 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. doi:10.1016/j.websem.2008.05.001.

[53] 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 (SAC 2015)*, R.L. Wainwright, J.M. Corchado, A. Bechini and J. Hong, eds, 2015, pp. 368–375. doi:10.1145/2695664.2695720.

[54] P. Zhao, X. Li, D. Xin and J. Han, Graph Cube: On Warehousing and OLAP Multidimensional Networks, in: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, T.K. Sellis, R.J. Miller, A. Kementsietsidis and Y. Velegrakis, eds, 2011, pp. 853–864. doi:10.1145/1989323.1989413.

[55] Z. Wang, Q. Fan, H. Wang, K. Tan, D. Agrawal and A. El Abbadi, Pagrol: Parallel graph OLAP over large-scale attributed graphs, in: *Proceedings of the IEEE 30th International Conference on Data Engineering*, I.F. Cruz, E. Ferrari, Y. Tao, E. Bertino and G. Trajcevski, eds, 2014, pp. 496–507. doi:10.1109/ICDE.2014.6816676.

[56] P. Wang, B. Wu and B. Wang, TSMH Graph Cube: A novel framework for large scale multi-dimensional network analysis, in: *Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA 2015)*, 2015. doi:10.1109/DSAA.2015.7344826.

[57] Ontotext, Configuring a repository, http://graphdb.ontotext.com/documentation/8.9/free/configuring-a-repository.html (Accessed: 20 October 2020).

[58] D. Lenat, The Dimensions of Context-Space, Technical Report, CYCorp, 1998, Accessed: 28 October 2020. https://www.researchgate.net/publication/243530490_The_dimensions_of_context-space.

[59] M. Benerecetti, P. Bouquet and C. Ghidini, On the Dimensions of Context Dependence: Partiality, Approximation, and Perspective, in: *Modeling and Using Context. CONTEXT 2001*, V. Akman, P. Bouquet, R.H. Thomason and R.A. Young, eds, Lecture Notes in Computer Science, Vol. 2116, Springer, Berlin, Heidelberg, 2001, pp. 59–72. doi:10.1007/3-540-44607-9_5.

[60] O. Khriyenko and V. Terziyan, A framework for context sensitive metadata description, *International Journal on Metadata, Semantics and Ontologies* **1**(2) (2006), 154–164, ISSN 1744-2621. doi:10.1504/IJMSO.2006.011011.

[61] J.J. Carroll, C. Bizer, P.J. Hayes and P. Stickler, Named graphs, provenance and trust, in: *Proceedings of the 14th international conference on World Wide Web (WWW 2005)*, A. Ellis and T. Hagino, eds, ACM, 2005, pp. 613–622. doi:10.1145/1060745.1060835.

[62] J. Hoffart, F.M. Suchanek, K. Berberich and G. Weikum, YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia, *Artificial Intelligence* **194** (2013), 28–61. doi:10.1016/j.artint.2012.06.001.

[63] S. Klarman, Reasoning with Contexts in Description Logics, PhD thesis, Free University of Amsterdam, 2013.

[64] U. Straccia, N. Lopes, G. Lukácsy and A. Polleres, A General Framework for Representing and Reasoning with Annotated Semantic Web Data, in: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, M. Fox and D. Poole, eds, 2010. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1590.

[65] O. Udrea, D. Recupero and V.S. Subrahmanian, Annotated RDF, *ACM Transactions on Computational Logic* **11**(2) (2010), 10–11041, ISSN 1529-3785. doi:10.1145/1656242.1656245.

[66] A. Zimmermann and J.M. Giménez-García, Contextualizing DL Axioms: Formalization, a New Approach, and Its Properties, in: *Joint Proceedings of the Web Stream Processing workshop (WSP 2017) and the 2nd International Workshop on Ontology Modularity, Contextuality, and Evolution (WOMoCoE 2017)*, D. Dell'Aglio, D. Anicic, P.M. Barnaghi, E.D. Valle, D.L. McGuinness, L. Bozzato, T. Eiter, M. Homola and D. Porello, eds, CEUR Workshop Proceedings, Vol. 1936, CEUR-WS.org, 2017, pp. 74–85. http://ceur-ws.org/Vol-1936/paper-07.pdf.

[67] M. Krötzsch, M. Marx, A. Ozaki and V. Thost, Attributed Description Logics: Reasoning on Knowledge Graphs, in: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, J. Lang, ed., 2018, pp. 5309–5313. doi:10.24963/ijcai.2018/743.

[68] 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. doi:10.1109/TKDE.2014.2330822.

[69] 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/.

[70] 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. http://ceur-ws.org/Vol-905/EtcheverryAndVaisman_COLD2012.pdf.

[71] 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)*, E. Pitoura, S. Maabout, G. Koutrika, A. Marian, L. Tanca, I. Manolescu and K. Stefanidis, eds, 2016, pp. 281–292. doi:10.5441/002/edbt.2016.27.

[72] L. Etcheverry, A.A. Vaisman and E. Zimányi, Modeling and Querying Data Warehouses on the Semantic Web Using QB4OLAP, in: *Data Warehousing and Knowledge Discovery. DaWaK 2014*, L. Bellatreche and M.K. Mohania, eds, Lecture Notes in Computer Science, Vol. 8646, Springer, Cham, 2014, pp. 45–56. doi:10.1007/978-3-319-10160-6_5.

[73] 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.

[74] 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*, C. Chung, A.Z. Broder, K. Shim and T. Suel, eds, 2014, pp. 467–478. doi:10.1145/2566486.2567982.

[75] 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.

[76] 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. doi:10.4018/jdwm.2013040104.

[77] V. Nebot and R. Berlanga Llavori, Building data warehouses with semantic web data, *Decision Support Systems* **52**(4) (2012), 853–868. doi:10.1016/j.dss.2011.11.009.

[78] T. Neuböck, B. Neumayr, M. Schrefl and C. Schütz, Ontology-Driven Business Intelligence for Comparative Data Analysis, in: *Business Intelligence. eBISS 2013*, E. Zimányi, ed., Lecture Notes in Business Information Processing, Vol. 172, Springer, Cham, 2014, pp. 77–120. doi:10.1007/978-3-319-05461-2_3.

[79] P. Russom, Data Lakes: Purposes, Practices, Patterns, and Platforms, 2017, Accessed: 20 October 2020. https://tdwi.org/research/2017/03/best-practices-report-data-lakes.

[80] 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. doi:10.1007/s10115-009-0228-9.

[81] 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, New York, 2010, pp. 411–438. doi:10.1007/978-1-4419-6515-8_16.

[82] P.O. Queiroz-Sousa and A.C. Salgado, A Review on OLAP Technologies Applied to Information Networks, *ACM Transactions on Knowledge Discovery from Data* **14**(1) (2019). doi:10.1145/3370912.

[83] C. Shi and P.S. Yu, Schema-Rich Heterogeneous Network Mining, in: *Heterogeneous Information Network Analysis and Applications*, Springer, Cham, 2017, pp. 181–199. doi:10.1007/978-3-319-56212-4_7.

[84] S. Loudcher, W. Jakawat, E.P.S. Morales and C. Favre, Combining OLAP and information networks for bibliographic data analysis: a survey, *Scientometrics* **103**(2) (2015), 471–487. doi:10.1007/s11192-015-1539-0.

[85] A. Ghrab, O. Romero, S. Skhiri and E. Zimányi, TopoGraph: an End-To-End Framework to Build and Analyze Graph Cubes, *Information Systems Frontiers* (2020), 1–24. doi:10.1007/s10796-020-10000-z.

[86] A. Matei, K.-M. Chao and N. Godwin, OLAP for Multidimensional Semantic Web Databases, in: *Enabling Real-Time Business Intelligence. BIRTE 2014, BIRTE 2013*, M. Castellanos, U. Dayal, T.B. Pedersen and N. Tatbul, eds, Lecture Notes in Business Information Processing, Vol. 206, Springer, Berlin, Heidelberg, 2015, pp. 81–96. doi:10.1007/978-3-662-46839-5_6.

[87] B. Benatallah, H.R. Motahari-Nezhad et al., Scalable graph-based OLAP analytics over process execution data, *Distributed and Parallel Databases* **34**(3) (2016), 379–423. doi:10.1007/s10619-014-7171-9.

[88] C. Shi, Y. Li, J. Zhang, Y. Sun and P.S. Yu, A survey of heterogeneous information network analysis, *IEEE Transactions*

on Knowledge and Data Engineering **29**(1) (2017), 17–37. doi:10.1109/TKDE.2016.2598561.

[89] S. Lee, S.R. Sukumar, S. Hong and S.-H. Lim, Enabling graph mining in RDF triplestores using SPARQL for holistic in-situ graph analysis, *Expert Systems with Applications* **48** (2016), 9–25. doi:10.1016/j.eswa.2015.11.010.

[90] 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.

[91] 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.

[92] L. Bozzato and C.G. Schuetz, Towards Distributed Contextualized Knowledge Repositories for Analysis of Large-Scale Knowledge Graphs, in: *Proceedings of the 35th Italian Conference on Computational Logic (CILC 2020)*, F. Calimeri, S. Perri and E. Zumpano, eds, CEUR Workshop Proceedings, Vol. 2710, CEUR-WS.org, 2020. http://ceur-ws.org/Vol-2710/short1.pdf.