Context Based Completeness Assessment For Data Warehouse Hierarchies

Camila Sanz[1,*,†], Adriana Marotta[1,†]

[1]Facultad de Ingeniería, Universidad de la República

Abstract

Data quality is an unavoidable issue in Data Warehouse systems due to their nature. Quality of data is compromised throughout the Data Warehouse (DW) lifecycle. This work addresses the completeness data quality dimension applied to DW hierarchies, which refers to the existence of a parent for each data value in a hierarchy. An incomplete hierarchy can result in data errors when applying aggregations and navigating cubes. The detection of incomplete hierarchies makes the user aware of the problem, avoiding biased decisions. As data quality is context-dependent, proper metrics should be defined for each domain and use case. We propose a generic data quality rule to assess hierarchy completeness, using OWL domain ontologies as context. With this approach, we obtain a data quality metric that adapts itself to any application domain.

Keywords

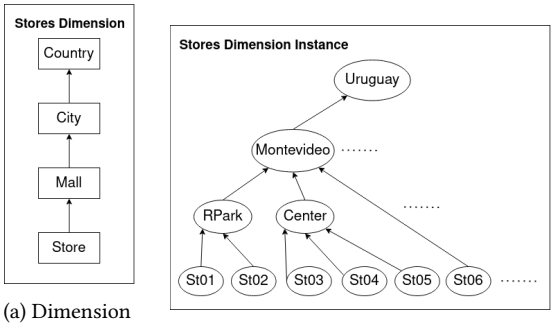
Completeness, Context, Data Warehouse, Data Quality

1. Introduction

Data Warehouses (DW) are populated with heterogeneous sources, which are re-structured to be queried from a multidimensional point of view. Multidimensional operations allow navigating the DW dimensions hierarchies, through the aggregation of data. As the quality of external sources is unknown, the user may obtain dirty data as the result of multidimensional operations.

Many aspects can be taken into account when trying to define and measure the quality of data. These aspects are called DQ dimensions, while the mechanisms for measuring DQ dimensions are called DQ metrics [1].

This work is part of an ongoing project that proposes a mechanism for detecting DQ problems in DW systems, considering context [2]. The general scenario consists of a final user who wants to measure DQ over her existing DW. Our approach is to use one or many domain ontologies as context to define possible DQ metrics. One of the main challenges is that the DW and the domain ontologies do not necessarily have the same structure, and this mismatch must be solved in order to use the ontology as context for the DW. Well known knowledge graphs such as Yago [3], DBPedia [4], etc. can be used as domain ontologies to get available contextual data. In our approach DQ metrics are defined through DQ rules.



(a) Dimension schema

(b) Dimension instance

Figure 1: Incomplete hierarchy

This paper addresses the DQ dimension *completeness* applied to hierarchies, a DQ problem that is referred to as incomplete hierarchies in [5]. The detection of incomplete hierarchies makes the user aware of data errors that can be generated when navigating cubes and applying aggregations. Figure 1 shows an example of hierarchy incompleteness, where the mall of the store "st06" is absent. If in the real world store "st06" is located in a mall, then the data absence represents a DQ problem. If not, this should not be a completeness problem.

In our work, we propose to use data context for deciding whether there is a completeness problem or not. Observe that for our approach the context is used as ground truth or DQ referential.

DQ is often described in the literature as "fitness for use" [1], in other words, DQ is context dependent [6, 7, 8, 9, 10]. DQ can be perceived differently according to the user, the location, or the application domain to which the data correspond. The latter is the charac-

DOLAP 2024: 26th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT 2024, March 25, 2024, Paestum, Italy

*Corresponding author.

†These authors contributed equally.

csanz@fing.edu.uy (C. Sanz); amarotta@fing.edu.uy (A. Marotta)

0000-0002-9366-7669 (C. Sanz); 0000-0001-6547-466X (A. Marotta)

teristic of the context we use in our approach. In the literature review presented in [11] about context dependant DQ, authors find a lack of formalization for context in DQ, pointing out that many of the proposals do not even define the context. [12]. Hurtado-Mendelzon's multidimensional model [13] is the foundation for many other existing specifications. It comprehends the main DW concepts. One particularity of this model is that enables the representation of heterogeneous dimensions. As a consequence of heterogeneous dimensions, using this model generates the need to handle the problem of incomplete hierarchies [14, 15]. Our approach tries to detect whether a hierarchy incompleteness is due to a DQ problem whereas these works focus on completing the hierarchies without considering the DQ point of view.

The main contributions of this work are: (i) a solution for detecting the problem of incompleteness in DW hierarchies, based on the context, and (ii) the specification of a DQ rule for hierarchy incompleteness detection (completeness rule), together with the specifications of the needed system components and relationships.

The rest of the document is structured as follows: in Section 2 we present some definitions that are necessary to introduce our completeness rule in Section 3, in Section 4 we present an example and in Section 5 its implementation. Finally, in Section 6 we conclude.

2. Previous Definitions

We introduce the DW and context formal specifications and the correspondences between their elements.

DW Formal Specification The specification of the DW is based on Hurtado-Meldenzon's multidimensional model [13]. We made some minor adaptations to their model for it to meet our necessities: in this paper, we introduce the use of tuples instead of atomic elements as members of categories. We only introduce the notation needed for understanding this work, details about the formal model are available in [13, 16].

A *dimension schema* is a tuple $\mathcal{S} = \langle \mathcal{C}, \nearrow \rangle$ where:

- $\mathcal{C} = \{c_1, \dots c_n\}$ is a finite set of categories.
- Each $c_j = \langle att_{j_1}, \dots, att_{j_p} \rangle$ is a list of attributes, representing the characteristics of the category.
- $\nearrow$: $\mathcal{C} \times \mathcal{C}$ s a binary relation between elements of $\mathcal{C}$, which represents the hierarchies of dimension $\mathcal{S}$, and $\nearrow^*$ its transitive closure.

A *dimension instance* for $\mathcal{S}$, is a tuple $\mathcal{D} = \langle \mathcal{U}, <, mem \rangle$, where:

- $\mathcal{U}$ a non empty finite set of tuples called *members*.
- $t = \langle val_1, \dots val_p \rangle$ is a tuple of the universe, each val_i is its value for attribute i and p is the length of the list of attributes of $mem(t)$.
- $mem : \mathcal{U} \to \mathcal{C}$ is a total function that for each value of $\mathcal{U}$ returns its category.
- $<$: $\mathcal{U} \times \mathcal{U}$ is a binary relation between elements of $\mathcal{U}$ that represents hierarchy instances of $\mathcal{S}$, and is consistent with $\nearrow$. $<^*$ is its transitive closure.

In this work it is not necessary for relation $<$ to be homogeneous nor strict (every member of a category has exactly one parent in each of the categories above). Both conditions are required in [13] but not in [16].

Context We use OWL domain ontologies to represent context. Given an OWL ontology O, we consider its classes CL, its object properties OP, and its data properties DP.

- $CL = \{CL_1, \dots CL_c\}$
- $OP = \{OP_1 \dots OP_{op}\}$, where $dom(OP_j)$ and $range(OP_j)$ are its domain and range.
- $DP = \{DP_1, \dots DP_{dp}\}$, where $dom(DP_j)$ is a class and $range(DP_j)$ is a data type.

An instance o of O is composed of its classes, data properties, and object properties instances.

- Class instance. An instance of CL_i is a set of individuals $cl_i = \{cl_{i_1}, \dots, cl_{i_t}\}$. Therefore, $cl = cl_1 \cup \dots \cup cl_c$ is an instance of CL.
- Data Property instance. An instance of DP_i is a relation between an instance of $dom(DP_i) = CL_i$ and a value of type $range(DP_i)$, $dp_i \subseteq cl_i \times range(DP_i)$. Therefore, $dp \subseteq dp_1 \cup \dots \cup dp_{dp}$ is an instance of DP.
- Object Property instance. An instance of OP_i is a relation between an instance of $dom(OP_i) = CL_j$ and at least one instance of $range(OP_i) = CL_k$, $op_i \subseteq cl_j \times 2^{cl_k}$. Therefore, $op \subseteq op_1 \cup \dots \cup op_{op}$ is an instance of OP.

DW - Context Mappings The correspondences between DW and ontology elements are binary relations, called mappings (only the mappings needed for the completeness rule are presented). The first argument is the DW element and the second one is the ontology element.

Consider a DW with n dimension schemas, its corresponding dimension instances, an ontology O and its instance o, as presented in 2.

A value of an attribute of a DW tuple is mapped to an instance of a data property.

$$MapVal \subseteq (val_{1_1} \cup \dots \cup val_{n_{|\mathcal{U}_n|}}) \times dp$$

Where $val_{i_j} = val_{i_j}[1] \cup \dots \cup val_{i_j}[k]$ and k is the number of attributes of category $mem_i(t_{i_j})$. Precondition: there must exist a mapping at schema level between the corresponding attribute and the data property schema.

A pair of categories of a DW hierarchy is mapped to an object property.

$$MapHier \subseteq ((\mathcal{C}_1 \times \mathcal{C}_1) \ldots \cup (\mathcal{C}_n \times \mathcal{C}_n)) \times OP$$

Precondition: there must exist a mapping between each of the categories and $dom(OP_j)$ and $range(OP_j)$ respectively. Furthermore, mapped categories must be related through $\nearrow^*$.

An element of the DW can be mapped to more than one element of the ontologies.

3. Assessing Hierarchy Completeness Using Context

Our formal DQ rule for hierarchy completeness is based on the definitions presented in section 2.

According to [5] an incomplete hierarchy is one in which some of its levels have missing values in one or more instances. These values can be nonexistent or unknown. The proposed completeness rule aims to detect values in a category that do not have a corresponding value in the immediate superior category and to check whether or not this situation is consistent with the context. To do so, we detect the tuples that do not have a parent in the immediate superior category and check their mapped values in the ontologies, used as referential, to see if there exists a corresponding value there.

Considering two categories c_{i_k} and $c_{i_{k'}}$ of dimension $\mathcal{S}_i$ such that $c_{i_k} \nearrow c_{i_{k'}}$, we define the set of tuples of category c_{i_k} that do not have a corresponding tuple in category $c_{i_{k'}}$ as:

$$\Delta_{c_{i_k},c_{i_{k'}}} = \{t \in \mathcal{U}_i | mem_i(t) = c_{i_k} \wedge \neg(\exists t')(mem_i(t') = c_{i_{k'}} \wedge t < t')\} \quad (1)$$

To define a completeness rule based on the context, two preconditions must hold:

- There must exist a mapping between two categories of a hierarchy and an object property: $MapHier((c_{i_k}, c_{i_{k'}}), OP_j)$.
- There must exist a mapping between an attribute value of a tuple t of category c_{i_k} and a data property instance dp_r, whose domain must be an instance of class $dom(OP_j)$: $MapVal(t[m], dp_r)$.

These preconditions are shown in Figure 2. The Figure shows a mapping between the pair of categories $(Category\ 1\ (c_{i_k}),\ Category\ 2\ (c_{i_{k'}}))$ and the object property OP_j : $MapHier((c_{i_k}, c_{i_{k'}}), OP_j)$. In addition, attribute value $val4$ of tuple $t = \langle val4, val5, val6 \rangle$, which belongs to category c_{i_k}, is mapped to instance dp_r of data property DP, whose domain is the class $CL_1 = Dom(OP_j)$: $MapVal(t[1], dp_r)$. If all of the previous conditions hold, then the **completeness rule**

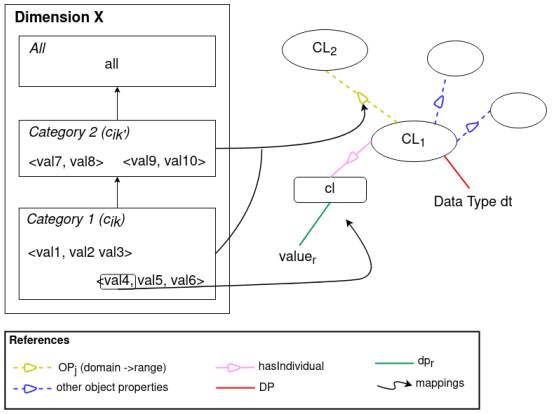


Figure 2: Completeness rule preconditions

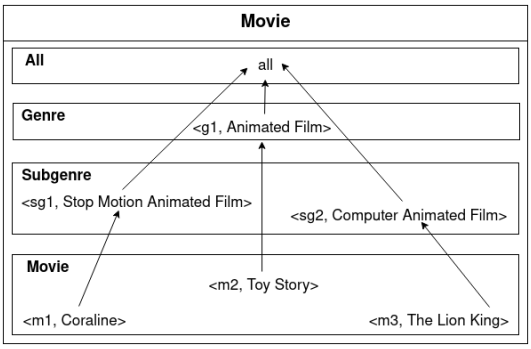


Figure 3: Movie Dimension

between category c_{i_k} and $c_{i_{k'}}$ can be defined as shown in equation 2.

$$\begin{aligned} t \in \Delta_{c_{i_k},c_{i_{k'}}} \wedge (cl, \_) \in op_j &\rightarrow Comp_{c_{i_k},c_{i_{k'}}}(t, 0) \\ t \notin \Delta_{c_{i_k},c_{i_{k'}}} \vee (cl, \_) \notin op_j &\rightarrow Comp_{c_{i_k},c_{i_{k'}}}(t, 1) \end{aligned} \quad (2)$$

The presented rule may be aggregated to measure completeness for a complete hierarchy and not only between a pair of levels.

4. Example

We consider a Cinema DW that stores the quantity of tickets sold and contains a dimension called Movie. The hierarchy goes as `Movie` → `Subgenre` → `Genre` → `All`. Figure 3 shows an instance of Movie dimension. An ontology populated with data from Yago knowledge graph for movies, shown in Figure 4, gives context to Movie dimension. The ontology contains information about Movies, Subgenres, and Genres and their relations.

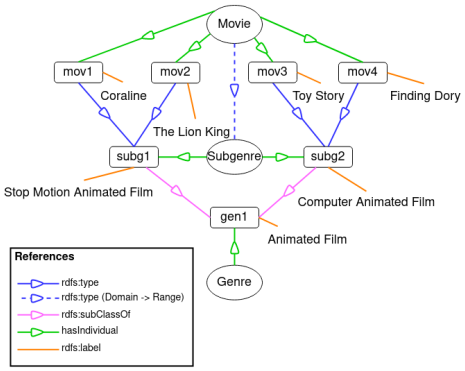


Figure 4: Movies ontology

We want to apply the completeness rule shown in equation 2 for categories *movie* and *subgenre*. To do so, we define the set $\Delta_{mov,subg}$ as shown in equation 1.

$$\begin{aligned}\Delta_{mov,subg} = \{t \in \mathcal{U}_{Movies} | mem_i(t) = movie \wedge \\ \neg(\exists t')(mem_i(t') = subgenre \wedge t < t')\} \\ = \{\langle m2, \textit{Toy Story}\rangle\}\end{aligned}$$

The necessary mappings are presented in equation 3. Note that preconditions also hold.

$$\begin{aligned}&MapVal(Coraline,\ (\texttt{mov1}, Coraline)) \\ &MapVal(The\ Lion\ King, (\texttt{mov2}, The\ Lion\ King)) \\ &MapVal(Toy\ Story,\ (\texttt{mov3}, Toy\ Story)) \\ &MapHier((movie, subgenre), \texttt{rdfs:type})\end{aligned} \tag{3}$$

The application of the completeness rule in equation 2 for the movie titled *Toy Story* is shown in 4. As $\langle m2, \textit{Toy Story}\rangle \in \Delta_{mov,\ subg}$ and $(\texttt{mov3}, \texttt{subg2}) \in op_{rdfs:type}$ hold, hierarchy completeness between *movie* and *subgenre* categories for this movie is 0.

$$\begin{aligned}\langle m3, \textit{Toy Story}\rangle \in \Delta_{mov,\ subg} \wedge \\ (\texttt{mov3}, \texttt{subg2}) \in op_{rdfs:type} \rightarrow \\ Comp_{mov,\ subg}(\langle m3, \textit{Toy Story}\rangle, 0)\end{aligned} \tag{4}$$

For movies titled *Coraline* and *The Lion King* the result of the application of the completeness rule is 1, because $\langle m1, Coraline\rangle \notin \Delta_{mov,\ subgen}$ and $\langle m2, \textit{The Lion King}\rangle \notin \Delta_{mov,\ subgen}$.

5. Implementation and Validation

We use an extended Datalog language with aggregation functions in order to allow the expressivity of different DW elements. Even though we used pyDatalog in our implementation [17], we present our solution using standard Datalog syntax in the sake of clarity.

```
MovieId                                    | MovieName     | RuleResult
-------------------------------------------|---------------|-----------
<__main__.movies object at 0x7fce0e948370> | Toy Story     | 0
<__main__.movies object at 0x7fce0e948640> | The Lion King | 1
<__main__.movies object at 0x7fce0e9488e0> | Coraline      | 1
```

Figure 5: Completeness rule result

Data Warehouse Implementation. We follow Datalog definition of a DW's Dimensions and Hierarchies presented in [18].
Dimensions: For dimension *movie* we introduce a predicate $Dim_{Movie}(X_{movie},\ X_{subgenre},\ X_{genre})$.
Hierarchies: We introduce a Datalog rule for each element in relation $\nearrow^*_{movie}$ to represent the hierarchy.
Completeness Assessment Implementation. The ontologies are managed using a Python library, owlready2 [19] and the DW is implemented using PyDatalog. Mappings between Python and Datalog are implemented using simple Python data structures. Datalog completeness rule is implemented in pyDatalog. Rules implementation and datasets are available in [20].

A first validation of the completeness rule was made by comparing the results obtained by the implementation against human observation of a small data set. To this end, hierarchy completeness problems were detected manually and compared with the automatic completeness assessment. We used the instance of Movie Dimension shown in Figure 3 and the movie ontology of Figure 4.

Manually detected values for completeness rule are shown in section 4 and the result of the execution of the completeness rule is shown in Figure 5. For the movie titled *"Toy Story"*, the manually detected value for completeness rule is 0, as shown in equation 4. In the excecution (Figure 5), the same value is obtained for that movie. For the other two movies the manually detected value and the obtained in the excecution are also the same.

6. Conclusions

In this work we proposed a general mechanism for DW hierarchy completeness assessment, considering context as domain ontologies. We presented the specification of a completeness rule, as well as the specifications of the concepts needed for supporting the rule, which are the DW, the domain ontology and the mappings between them. We also presented an example and an implementation that show how the proposed mechanism is applied.

We believe that our proposal has a level of abstraction in the formalizations that enables evaluating hierarchy completeness for any DW in any context.

This work is being extended building solutions for other DQ dimensions. Future work includes applying our proposal to a real world case.

References

[1] C. Batini, M. Scannapieco, Data and Information Quality, Data-Centric Systems and Applications, Springer International Publishing, Cham, 2016. URL: http://link.springer.com/10.1007/978-3-319-24106-7. doi:10.1007/978-3-319-24106-7.

[2] C. Sanz, Context based data quality rules for multidimensional data, in: Z. Bao, T. K. Sellis (Eds.), Proceedings of the VLDB 2022 PhD Workshop colocated with the 48th International Conference on Very Large Databases (VLDB 2022), Sydney, Australia, September 5, 2022, volume 3186 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: http://ceur-ws.org/Vol-3186/paper_3.pdf.

[3] Yago, Yago knowledge graph, https://yago-knowledge.org/, 2023.

[4] DBPedia, Dbpedia knowledge graph, https://dbpedia.org, 2023.

[5] M. Golfarelli, S. Rizzi, Data Warehouse Design: Modern Principles and Methodologies, McGraw Hill LLC, 2009. URL: https://books.google.com.uy/books?id=cRMkPa3TpPYC.

[6] L. Bertossi, F. Rizzolo, L. Jiang, Data Quality Is Context Dependent, in: Enabling Real-Time Business Intelligence, Lecture Notes in Business Information Processing, Springer, Berlin, Heidelberg, 2010, pp. 52–67. URL: https://link.springer.com/chapter/10.1007/978-3-642-22970-1_5. doi:10.1007/978-3-642-22970-1_5.

[7] M. Helfert, O. Foley, A Context Aware Information Quality Framework, in: 2009 Fourth International Conference on Cooperation and Promotion of Information Resources in Science and Technology, 2009, pp. 187–193. doi:10.1109/COINFO.2009.65.

[8] A. L. McNab, D. A. Ladd, Information Quality: The Importance of Context and Trade-Offs, in: 2014 47th Hawaii International Conference on System Sciences, 2014, pp. 3525–3532. doi:10.1109/HICSS.2014.439.

[9] G. Rogova, M. Hadzagic, M. O. St-Hilaire, M. C. Florea, P. Valin, Context-based information quality for sequential decision making, in: 2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2013, pp. 16–21. doi:10.1109/CogSIMA.2013.6523818.

[10] D. M. Strong, Y. W. Lee, R. Y. Wang, Data Quality in Context, Commun. ACM 40 (1997) 103–110. URL: http://doi.acm.org/10.1145/253769.253804. doi:10.1145/253769.253804.

[11] F. Serra, V. Peralta, A. Marotta, P. Marcel, Use of context in data quality management: a systematic literature review, 2022. arXiv:2204.10655.

[12] A. Ranganathan, R. H. Campbell, An infrastructure for context-awareness based on first order logic, Personal and Ubiquitous Computing 7 (2003) 353–364. URL: https://link-springer-com.proxy.timbo.org.uy:88/article/10.1007/s00779-003-0251-x. doi:10.1007/s00779-003-0251-x.

[13] C. A. Hurtado, A. O. Mendelzon, OLAP Dimension Constraints, in: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '02, ACM, 2002, pp. 169–179. URL: http://doi.acm.org/10.1145/543613.543636. doi:10.1145/543613.543636.

[14] M. Caniupán, L. Bravo, C. A. Hurtado, Repairing inconsistent dimensions in data warehouses, Data & Knowledge Engineering 79-80 (2012) 17–39. URL: https://www.sciencedirect.com/science/article/pii/S0169023X12000596. doi:https://doi.org/10.1016/j.datak.2012.04.002.

[15] C. Dyreson, T. Pedersen, C. Jensen, Incomplete information in multidimensional databases, Multidimensional databases (2003). doi:10.4018/978-1-59140-053-0.ch010.

[16] L. Bertossi, M. Milani, Ontological Multidimensional Data Models and Contextual Data Quality, J. Data and Information Quality 9 (2018) 14:1–14:36. URL: http://doi.acm.org/10.1145/3148239. doi:10.1145/3148239.

[17] pyDatalog library, https://pypi.org/project/pyDatalog/, 2023.

[18] C. Sanz, A. Marotta, Using ontologies as context for data warehouse quality assessment, in: Big Data Analytics and Knowledge Discovery: 25th International Conference, DaWaK 2023, Penang, Malaysia, August 28–30, 2023, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2023, p. 3–17. URL: https://doi.org/10.1007/978-3-031-39831-5_1. doi:10.1007/978-3-031-39831-5_1.

[19] owlready2, owlready2 library, https://owlready2.readthedocs.io/en/v0.37/, 2023.

[20] C. Sanz, Movies example, 2023. URL: https://github.com/camila-sanz/Example.