

Three Metric-Based Method for Data Compatibility Calculation

Daniel Vodňanský 

Faculty of Informatics and Statistics, Prague University of Economics and Business, W. Churchill Sq. 1938/4, 130 67 Prague 3, Czech Republic

Corresponding author: Daniel Vodňanský (xvodd00@vse.cz)

Abstract

This article analyzes ways of calculating characteristics of data and most common data structure types that allow comparison between them or on a time axis. To achieve this, it studies the key aspects of relational databases, XML, JSON and RDF structure types. These data structure types are compared to multiple isolated approaches to data quality and other data characteristics measurements. The goals of the article are the calculation method itself and a storage structure for calculated values. The article presents a method of characterization of data and data structure types based on the calculation of three metrics: the amount of structuredness, the amount of hierarchicallity and the amount of information. This triad of metrics allows comparison between various data sets (objects), for example evaluating the complexity of the transformation of data from one data object to another, as well as with data structure types (as mentioned above). Based on the vector of three metrics, the calculation method of the compatibility between data and data structure type is proposed. This method can help select the most compatible data format for existing data. The calculated values of metrics can also detect non-optimal storage design and classify data transformations. The method was evaluated on an example case study, which showed its usability on an example demonstration data set. It can be used in the process of data modelling to help select optimal data structure type, to design a data transformation process and to optimize existing data storages.

Keywords

Data metrics, Amount of information, Metadata, Relational database, XML, JSON, RDF, Ontology, Transformation, Structuredness, Hierarchicallity, Normalization, Visualization.

Citation: Vodňanský, D. (2021). Three Metric-Based Method for Data Compatibility Calculation. *Acta Informatica Pragensia*, 10(1), 38–60.
<https://doi.org/10.18267/j.aip.145>

Academic Editor: Stanislava Mildeova, University of Finance and Administration, Czech Republic

Copyright: © 2021 by the author(s). Licensee Prague University of Economics and Business, Czech Republic.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution License (CC BY 4.0).

1 Introduction

The motivation for this article is to design a method for characterization of various data (independently on its structure type, format, or data model) by three calculable metrics. This calculation allows comparison between data sets (data with another data) and the calculation of their compatibility with any data structure type (data with relational database, XML etc.). Compatibility calculation helps to design the transformation of any data set to the most compatible data structure type. The calculated compatibility can be compared with the complexity of data transformation represented by its transformation length (vector of metrics values changes).

To demonstrate the problem addressed by our approach, we shall consider complex enterprise data stored in a relational database. Multiple questions can be asked:

1. Can the data be more easily converted and stored by XML or JSON (or any other format or data model)?
2. Which parts of the data can most easily be converted and stored by XML?
3. Which parts of the data are unsuitable for storing in a relational database?
4. Which part of the data changes the most in time?
5. Which parts of the data most increase its size and what is the probable reason?
6. Which parts of the data are the most redundant?

This article builds on the idea that to answer such questions, a single three metric-based method can be applied. These metrics are the amount of structuredness, hierarchicallity and information.

2 Background

There have been multiple isolated researches comparing single or two data structure types, their usage advantages and disadvantages and specifics for their most compatible data. Typically, relational database and XML (Krishnamurthy et al., 2001; Ma et al., 2020; Shanmugasundaram et al., 2001; Song & Haw, 2020) or XML and JSON (Helland, 2017). But current research lacks a **unified systematic approach** that allows comparing various pairs of data structure types and their corresponding data, which is the perspective of this article.

The problem of the amount of **data structuredness** is usually limited to focusing on data at a specific structuredness level. Paper (Florescu, 2005) represents a general consideration of the problem of less structured information and data as one of the main limitations of relational databases. Article (Šperková, 2014) mentions that unstructured data is generated mostly by social networks nowadays. This paper extends this structuredness approach into a continuous calculable metric. Article (Wellenzohn et al., 2020) propose an indexation method for semi-structured (and also hierarchical) data.

This paper also addresses the problem of measuring **how hierarchical the data is** (the structure of a tree). Despite multiple definitions of a tree, no work approaching a measurable continuous value has been found. For example, (Musca et al., 2011) only say that a structure is hierarchical if "lower level units are part of higher level units". This article uses this definition to identify non-hierarchical parts of data and to calculate the amount of hierarchicallity based on their share.

The amount of information has been well known since the approach of Shannon's entropy. However, its main philosophy of the amount of information as a "non-redundancy" expressing the "unpredictability" of data is extended by Algorithmic information theory (AIT) based on Kolmogorov complexity, which deals with "the relationship between the computational method, information and randomness" and thus measures in particular the complexity of the relevant data (Hutter, 2007). This paper further applies AIT to measure the amount of information.

There are two main objectives of this article. The first objective is the calculation method, that takes a data object (any bounded set of data) and computes a triad of values – its amount of structuredness, hierarchicallity and information, belonging to the interval $\langle 0; 1 \rangle$. The second objective is the storage method that allows saving computed values referenced to specific data and time into a structured database called “metadata warehouse”. Both objectives are strongly interconnected – each method requires using the other (the calculation method uses previously stored values). Their usability is evaluated on experimental data examples.

The calculated values that characterize various data can help to demonstrate their characteristics in simple tables or graphically and to facilitate communication between data experts, software engineers as well as statisticians and content managers.

Many universal or specifically targeted metrics of data have been designed and used. A large number of them refers to the concepts of “Big Data” and “data quality”. Article (Morton, 2014, p. 4) presents the following characteristics of Big Data – volume, variety, velocity, value and validity. Regarding data quality, (Floridi, 2013) presents “accessibility, accuracy, availability, completeness, currency, integrity, redundancy, reliability, timeliness, trustworthiness and usability”. Article (Närman et al., 2011) present the *main dimensions of data quality*, which they study in the context of Enterprise architecture: *completeness, consistency, timeliness, relevance and accuracy*, which is, however, a concept of metric rather than a dimension. This approach is also related to the “data governance” topic, which (Begg & Caira, 2012) define as a framework – a logical structure for classifying, organizing and communicating complex activities involving making decisions about them and taking action on business data.

Multiple papers have studied topics related to the specific three metrics used in this article. The structuredness of data is widely understood as a discrete rather than continuous variable, having three levels – structured, semi-structured and unstructured, described e.g. by (Bartmann et al., 2011). Similarly, (Pokorný, 2010) extends the concept of classic and traditional database, because the goal is to *manage a rich collection of structured, semi-structured, and unstructured data, spread in various enterprise repositories and on the Web*. No relevant publications about data hierarchicallity have been found, just general definitions of hierarchies and the author’s previous poster (Vodňanský & Zamazal, 2016) measuring hierarchicallity on OWL ontologies.

The amount of information is a popular topic since Shannon’s concept of information entropy (Shannon, 2001), which is limited to a linear sequence of messages and their probability in an information channel. This concept is usable on single-dimension structured data only, but it is not flexible enough for usage on the multiple data structure types in this paper. According to (Hutter, 2007), the algorithmic amount of information is the “length of the shortest description” of a string or a data set which can be created by a compression (e.g. by Lempel-Ziv, bzip2 and PPMZ algorithms). Authors in article (Grünwald & Vitányi, 2008) mention a “minimum number of bits from which a particular message can effectively be reconstructed”. Paper (Meinsma, n.d.) discusses the relation to lossy compression and proves that “there is no lossless compression that strictly reduces every file” and (Grünwald & Vitányi, 2008) recognize the extension of the AIT concept to lossy algorithms, which “allows formalizing more sophisticated concepts such as ‘meaningful information’ and ‘useful information’”.

3 Applied methods

This article uses a synthesis of sorted information sources of current theoretical knowledge (summarized in chapter 2). Data structure types are gradually qualitatively analyzed in part 4.1 and the result of their analysis is formalized by the three metrics described in part 4.2. By summarizing the possible combinations of metrics value changes, the data transformation types are derived in part 4.3.

The evaluation method is an experimental verification of the applicability of the metrics calculation method on an example case study (chapter 5), which uses three sets of demonstration data and also analyses their transformations. The output of this paper can be used as a quantitative analytical method based on deduction, which is practically verified on demonstration and real data.

4 Designed calculation method

This paper designs a method which allows describing various data by a triad of metrics.

4.1 Considered data structure types

This paper studies the following common data structure types.

Relational databases (RDBs) are undoubtedly one of the dominant data storage technologies, popular on a theoretical basis from (Codd, 1990), strongly popular in the enterprise sector (Ramel, 2015). RDBs are oriented primarily on structured data, despite the support of XML and JSON storing typical for most RDBS systems and the NoSQL concept. An important aspect of RDBS is normalization – their main intention is to save structured data with a minimum redundancy.

XML is a format complementary with RDBs especially in data interchange and semi-structured data storage. Compared with RDBs, it is self-describing and more flexible (Helland, 2017). It can store data from unstructured to fully structured level. The storage is hierarchical (if we ignore the non-uniform mechanism of elements referencing each other). XML documents can contain any level of redundancy – data normalization can be impossible in this format when the information represented by the XML document is not hierarchical.

JSON represents a brackets-based notation of objects originally from the JavaScript language. However, the following description corresponds to multiple formats with similar structures including Redis and other “key-value” storages. JSON has almost same characteristics as XML, but it is focused primarily on structured data and probably the record can be shorter due to using brackets instead of elements, which decreases data redundancy.

RDF is a data format used for publishing Linked Data together with schema defined by RDBs and OWL. Its specific form depends on a serialization of RDF, but the main aspect is its graph orientation (even its XML serialization does not use the classic tree relation). Its decentralized web-based concept allows and assumes redundancy of data published by multiple subjects.

4.2 Three data metrics

The metrics designed in this paper are marked as follows:

- amount of structuredness – $s(D)$,
- amount of hierarchicallity – $h(D)$,
- amount of information – $i(D)$,

while $s(D), h(D), i(D) \in \langle 0; 1 \rangle$. D is a **data object** – a bounded set of data that can consist of partial **child data objects**, or it can be **elementary** (the smallest possible object that still allows calculating the values of metrics). The computational formulas also use $v(D)$ as a bit size of the data object D and $c_1 \dots c_n$ (or a string length, if all data is textual), which represent the elementary child objects of data object D .

The triad of data metrics can be visualized in a “**three-dimensional data metrics space**” – a cube with edge length 1 (all values $\in \langle 0; 1 \rangle$), where one corner is the point $[0; 0; 0]$ and the three adjacent edges correspond to these three data metrics (see further Figure 3).

4.2.1 Amount of structuredness

As mentioned in the Background section, this paper, compared to the three-level structuredness concept (structured, semi-structured and unstructured), considers structuredness as a **continuous value**.

The requirement for this metric is its ability to respond to two aspects of the examined data – its ability to break down into smaller parts and the degree of structure of individual maximally distributed parts of data (child elementary data objects). For elementary data objects, a significantly simpler way of determining the value of this metric can be expected. Both aspects should increase the resulting value of the degree of structuring of the examined data. This is also the reason why a simple weighted average calculation cannot be used – it ignores the ability of data to distribute.

Calculating the amount of structuredness of a large data object with multiple levels of child objects is based on splitting the objects into elementary child objects, while all middle level (non-elementary) child objects are ignored.

$$s(D) = \frac{\sum_{j=1}^n \left(1 - \frac{v(c_j)}{v(D)} (1 - s(c_j)) \right) \cdot v(c_j)}{v(D)}$$

Formula 1. Amount of structuredness.

This amount is calculated by Formula 1, where $s(D)$ is the result amount of structuredness, v is a bit or text size function, D is the data object and $c_1 \dots c_n$ are its the elementary child objects. The weight of a data elementary child object is $\frac{v(c_j)}{v(D)}$.

Such a setting of the formula means that the amount of structuredness of, for example, a data object D_2 consisting of two equally large child objects (one structured and one unstructured, both having a weight of $\frac{1}{2}$) is not $\frac{1}{2}$, as would be the case with a simple weighted average calculation, but the following:

$$s(D_2) = \left(1 - \frac{1}{2} (1 - 1) \right) \cdot \frac{1}{2} + \left(1 - \frac{1}{2} (1 - 0) \right) \cdot \frac{1}{2} = 0.5 + 0.25 = 0.75.$$

The reason is that the object is divisible into two child objects; therefore, the unstructured child data object reduced the value by only 0.25.

This method is based on the assumption that the structuredness $s(c_j)$ of an elementary data object c_j is obvious and its determination can be automated – an elementary object is usually a small single value (e.g. a number or a single-word string in a database etc.) or a larger structure that cannot be split without losing the information it represents (e.g. a sentence, any longer text as well as media files). The principle of this method is that the unstructured elementary object c_j decreases the structuredness of object D proportionally to its share of the object.

For example, a single long text is an unstructured elementary object. By organizing thousands of such texts into a list, database etc., we get a much more structured superior object. Therefore, the limit of $s(D)$, as the number of data objects increases, is one; the organization of objects next to each other and the ability to split their superior data object makes it more structural.

4.2.2 Amount of hierarchicallity

The amount of hierarchicallity expresses how a data object fulfills the definition of hierarchical structure – a structure that can be expressed as a tree with a root node and its children on multiple levels (e.g. a web page or a XML file) without creating a redundancy. Formula 2 shows the definition of entity a having a hierarchical property H is the following (based on the definition from (Musca et al., 2011)).

$$\forall a, b: (H(a) \wedge R(a, b)) \Rightarrow (\neg \exists c: c \neq b \wedge R(a, c)).$$

Formula 2. Hierarchical property definition.

Relation $R(a, b)$ means that b is a 's parent – for example in a database, a row in a table refers by a foreign key to another “parent” row, which creates 1: N relation, in XML a child element has a parent element etc. An entity referring to multiple others (having for example $M:N$ relation) breaks the definition of hierarchical property – such structure cannot be represented by a tree without repeating information from another entity. More examples of this problem are given in (Vodňanský, 2016).

The amount of hierarchicallity is calculated by Formula 3.

$$h(D) = \frac{\sum_{j=1}^n v(H(c_j))}{v(D)}$$

Formula 3. Amount of hierarchicallity.

This method is also based on splitting a data object into elementary child objects, while $v(H(c_j))$ is the size of an elementary child data object that **is not a part of any data object breaking** the definition of hierarchical property and is not a child object of any object breaking the definition.

4.2.3 Amount of information

As the background stated, Shannon's concept of information entropy is not usable for complex data structures. This paper applies AIT by measuring the compression rate of the whole data object with a given compression algorithm. This method can be computationally demanding on large objects and no algorithm can achieve the lowest possible size of compressed data, but using AIT is possible on data of any type and structure with a clear result.

The amount of information is calculated by Formula 4.

$$i(D) = \frac{v(k(D))}{v(D)}$$

Formula 4. Amount of information.

Function k is the compression function – $v(k(D))$ is the size of the compressed data object and $v(D)$ is the size of the original object.

The characteristics of data structure types listed in the previous section are summarized in Table 1, which includes the intervals of data metrics values of an optimal data storage – a storage filled by data with the most suitable characteristics.

Table 1. Optimal ranges of metrics values.

	amount of structuredness	amount of hierarchicallity	amount of information
RDBS	{1}	$\langle 0; 1 \rangle$	{1}
XML	$\langle 0; 1 \rangle$	{1}	$\langle 0; 1 \rangle$
JSON	{1}	{1}	$\langle 0; 1 \rangle$
RDF	{1}	$\langle 0; 1 \rangle$	$\langle 0; 1 \rangle$

4.3 Data transformations classification

The triad of data metrics allows classifying a transformation of data, based on the changes of metrics values. Data transformation is a process in which data object D_0 is transformed into data object D_1 , while both objects represent the same information.

Before the process of transformation, the system of three metrics helps to calculate the compatibility of data object D_1 and data structure type S_1 . Formula 5 shows the compatibility calculation (function k).

$$k_T(D) = 1 - \frac{1}{\sqrt{3}} \cdot \min_{S \in T} \sqrt{(s(S) - s(D))^2 + (i(S) - i(D))^2 + (h(S) - h(D))^2}$$

Formula 5. Amount of compatibility of data object D_1 and data structure type S_1 .

Data object D is defined by a triad of data metric values – amount of structuredness $s(D_1)$, amount of hierarchicallity $h(D_1)$ and amount of information $i(D_1)$. Data structure S of type T is defined by three intervals or values of these metrics, as shown in Table 1. The number $\sqrt{3}$ is the longest possible distance (geometrically, it is the inner diagonal of the cube). Geometrically, compatibility is based on the distance between a point (D_1) and a block (S_1) inside a three-dimensional data metrics space (estimated transformation length) – the shorter this distance is, the higher is the compatibility.

The real data transformation length is the distance of two points (scaled on the interval $\langle 0; 1 \rangle$) representing the first object D_0 and the transformed object D_1 . This number has the opposite function than compatibility – the higher the compatibility, the lower the transformation length should be; both numbers can be compared with each other. The calculation is shown in Formula 6.

$$t(D_0, D_1) = \frac{1}{\sqrt{3}} \sqrt{(s(D_1) - s(D_0))^2 + (h(D_1) - h(D_0))^2 + (i(D_1) - i(D_0))^2}.$$

Formula 6. Data transformation length between data objects D_0 and D_1 .

The transformation can be classified into 6 basic types (one data metric value is changing) of transformations and multiple other combined types (more metrics values are changing). The basic types are described by the following parts.

4.3.1 Structuralization

The process in which the amount of structuredness increases is typical to many job roles and disciplines, including data modelling and filling a database, accounting, semantic annotation (explained in more detail by (Oren et al., 2006)). This process is difficult to automate; one of such methods is FRED (Gangemi et al., 2017), a tool that transforms natural language text to RDF/OWL format, OCR (optical text recognition from images) and other natural language processing methods. In this context, (Šperková, 2014) mentions the process of “sorting, cleaning and transformation of unstructured data into matrixes”. Structuralization is neutral to the other transformation types – it allows increasing and decreasing both other data metrics.

4.3.2 Destructuralization

This inverse transformation type is typical for most interpersonal communication based on using structured data, e.g. a simple presentation. This process can only keep or increase the amount of hierarchicallity – by using less structured data, the definition of hierarchical property cannot be broken. The amount of information typically decreases but keeping this amount or even normalizing the data while being destructuralized cannot be excluded.

4.3.3 Hierarchization

This process, in which violations of hierarchical property definition are being removed, is typical for using data with a strict hierarchical structure – creating a webpage, XML document or even this paper itself (if inner cross-references are excluded). This process is described in more detail in (Vodňanský, 2016).

Hierarchization is neutral with respect to structuredness. It can very probably increase the level of redundancy when non-hierarchical structures are transformed and decrease the amount of information, while it cannot normalize the data and increase it in any way.

4.3.4 Dehierarchization

This process creates breakings of the definition of hierarchical property. It can be assumed that this creation of double reference from a single entity is intentional (a reference cannot occur randomly). It is typical for processes creating graphs and diagrams – data models, ontologies, mind maps etc.

This inverse transformation can increase the amount of structuredness. This metric cannot be decreased just by organizing data non-hierarchically. It also very typically occurs together with the normalization process, while denormalization is practically excluded – a non-hierarchical reference does not create redundancy, only if it is redundant itself.

4.3.5 Normalization

The process, in which redundancies are removed and the amount of information increases, is very popular in RDBs and its principles are described in multiple papers, e.g. (Codd, 1990; Halpin, 2001). A more detailed description and examples are superfluous for this paper. This paper understands normalization as a process applied on existing data, not just its schema and not just RDBs.

It is typical for the same activities as the strongly related dehierarchization and it excludes hierarchization due to reasons set out in that section. This process is typical for structured data and Structuralization, but it is admissible even for unstructured data, which can be less redundant, therefore it is neutral to the amount of structuredness.

4.3.6 Denormalization

This inverse process, which creates redundancy, occurs when data is being prepared for easier or faster access for a machine (a data cache storage) or a human.

It can probably decrease structuredness but allows its increase (for which there is probably no practical example) and it also probably increases hierarchicallity; dehierarchization cannot denormalize data.

4.3.7 Combined transformation

The discussion in the paragraphs above described the basic transformation types and their relation to each other. These relations can be combined into more complex types, in which two or three metrics change. By excluding the combinations described above and inverse pairs, 18 types exist and can be illustrated by the following Venn diagram (Figure 1), where a letter represents the initial letter of the transformation type name and an added letter D represents the inverse type.

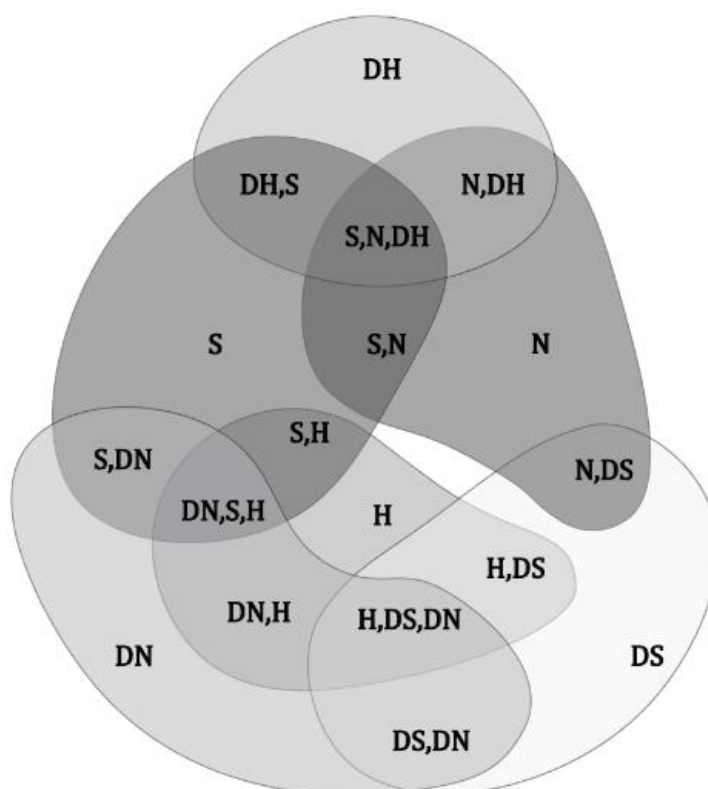


Figure 1. Venn diagram of permissible combined data transformation types.

This diagram in Figure 1 allows placing each classified data transformation, regardless of the size of the amount change. Simple transformations are represented by letters S – structuralization, H – hierarchization and N – normalization, the inverse transformations add the letter D (e.g. DN as denormalization). Combined transformations have multiple abbreviations separated by a comma.

4.4 Metadata warehouse

The metadata warehouse principle is based on the popular OLAP (online analytical processing) concept used in BIT (Business intelligence technologies). The metadata warehouse does not only serve as a repository, it is also an instrument of calculation itself – the values for parent objects are calculated from already stored values for lower-level objects.

There are 3 main differences from the OLAP concept based on tables of facts and tables of dimension. The first difference is that it uses the three metrics together, which can serve as dimensions as well – hierarchicallity can be studied on multiple levels of structuredness and vice versa. The second difference is that values cannot be aggregated – they must be computed separately for an object on all levels (multiple data objects may contain no redundancy but can be redundant to each other inside the parent data object, they can break the hierarchy together or be organized more structurally). The third difference is that the structure of the warehouse is universally set for various data, while the OLAP cube is usually designed for a specific usage.

The metadata warehouse can be built once (to study a specific situation ad-hoc) or continuously (to monitor data changes in time). The main principle of storing metadata in a warehouse is its referencing to the unique data it represents (in a format depending on data structure type). Metadata warehouse is a relational database with structure shown in Figure 2.

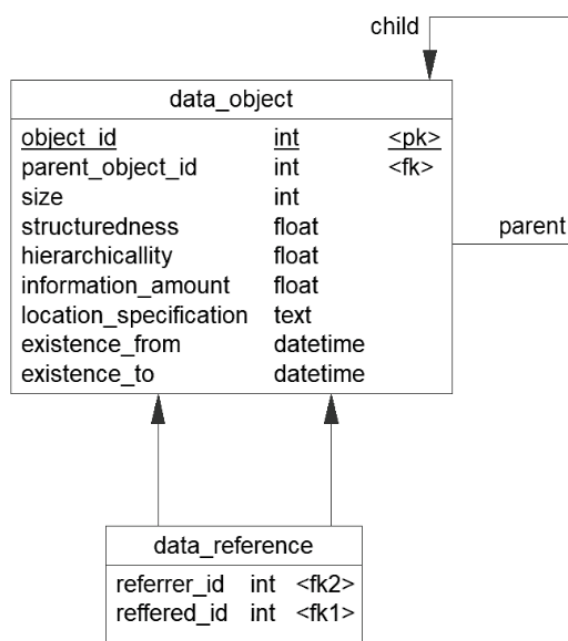


Figure 2. Metadata warehouse physical model.

The first table shows an object that has its primary key, foreign key referring to a parent data object inside which this object is contained (the main, “root” object has null value and there must be no cycle), bit size (or string length for fully textual data), three amounts of metrics, reference to the represented data explained above and two datetimes defining the time range when the data object is changeless – each change in any parameter creates a new data object inside the warehouse with a new time range which can express a data transformation.

The second table shows the $M:N$ relation, storing the information about the data mutual referencing of data (foreign keys in database, links in HTML, predicates in RDF data), when the referrer and rereferred objects are distinguished (they usually are in relation with $1:N$ cardinality). This table is used for hierarchicallity calculation.

4.5 Metric-based data visualization

The triad of metrics allows three-dimensional visualization of most situations described in this paper. This visualization method uses a cube with the amount of hierarchicallity on axis x , the amount of information on axis y and the amount of structuredness on axis z . While all metrics belong to the interval $(0; 1)$, the three-dimensional space is represented by a cube shown in Figure 3. The axis from top to bottom is the amount of information, the axis from left to right is the amount of hierarchicallity and the oblique axis from bottom left to the middle is the amount of structuredness.

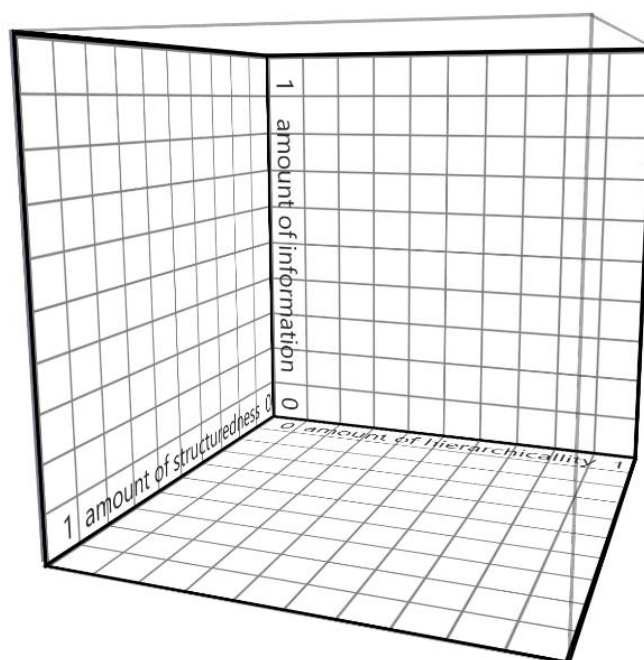


Figure 3. Three-dimensional data metrics space.

Inside this space, a data object is shown as a point with three metric values as coordinates. Data structure types, based on Table 1, with intervals or single values are shown as blocks, rectangles, lines or points – their intersections show areas compatible with multiple structure types. Data transformations are shown as arrows, representing a vector from one data object (point) to another. Webpage (Vodňanský, 2020) proposes an open-source application for 3D visualization of data object metrics and data structure types.

5 Method evaluation by example case study

This part shows three practical examples of using the principles on two purposefully created demonstration data sets with minimal size to illustrate the calculation. Both data sets represent the same information. Both are used to show the principle of data metrics and compatibility calculation. They are also used to show an example of data transformation, expressed by transforming the first data set into the second one.

5.1 Relational database

The first data set consist of a RDB with 3 tables, based on the topic of organizing sport courses. The database table are shown in Table 2, Table 3 and Table 4.

Table 2. RDB table Area of the first data set.

id_area	name	description
1	swimming pool	This is an example-long description of a swimming pool
2	football pit	<i>null</i>
3	tennis court	Tennis

Table 3. RDB table Course of the first data set.

id_course	day	time	id_area	id_period
1	1	15:00:00	1	3
2	2	15:00:00	3	3
3	5	14:40:00	3	3
4	2	14:00:00	1	4
5	3	15:30:00	2	4

Table 4. RDB table Period of the first data set.

id_period	from	to
3	2019-03-01	2019-06-30
4	2018-09-01	2019-02-26

The metadata warehouse is built in six steps.

Step 1: This step creates the “data_object” table, while id, creation (“existence_from”) and location specification is set (in the format “/table/id/column”). This process starts from the main “root” object to fill out the reference to the parent object.

Step 2: This step fills the “data_reference” table, while in RDBs, only objects at RDB row level are practically referring using their foreign keys.

Step 3: This step fills the size of the objects. Due to the text format of the data, a simple string length can be used instead of bit size.

Step 4: This step calculates the amount of hierarchicallity. The object has value 1 if it does not contain any child objects both referencing each other in data_reference table non-hierarchically (breaking the definition of hierarchical property). This practically happens for the root object only, which contains all three tables, while all rows of Course table refer to the rows of both other tables. The value is calculated by Formula 3, dividing the size of elementary data objects that are children of the object breaking the criterion by the size of the whole object. For the root object, it is $h(D_1) = \frac{\sum_{j=1}^n v(H(c_j))}{v(D_1)} = \frac{95}{180} = 0.5278$.

Step 5: This step calculates the amount of structuredness. All elementary objects are structured except the description of Area 1 (data object 7), which contains a sentence. This data object decreases the metric value for all parent objects in proportion to its share, therefore the amount for object 3 is $\frac{13\left(1-\frac{13}{68}(1-1)\right)+54\left(1-\frac{54}{68}(1-0)\right)}{68} = 0.3546$, where 13 is the size of child object 6, 54 is the size of child object 7, and 68 is the size of the entire data object 3. Other data object values (especially 2 and 1, which are the parent of this one) are computed analogically.

Step 6: This step calculates the amount of information. This applies the AIT principle and uses bzip2 algorithm to compress the data (in this case, a simple text representation of the data objects, in

which the child objects are split by a single space). The value of root data object 1 is $\frac{204}{238} = 0.8571$, while 204 is the length of the compressed text and 238 is the original length. For very small data objects, the compressed text can be longer than the original; then the value is 1.

The final metadata warehouse is shown in the Appendix A. The compatibility of this (root) data object can be calculated as

$$k_{RDBS}(D) = 1 - \frac{1}{\sqrt{3}} \cdot \min_{S \in T} \sqrt{(0.8233 - 1)^2 + (0)^2 + (0.8571 - 1)^2} = 0.8688$$

Analogically, the compatibility with the other data type structures can be calculated, which results in Table 5. It shows that even though this data set is a RDB, its compatibility with RDBs is decreased by an unstructured child object, but it is still the most appropriate data structure type together with RDF. Compatibility with XML and JSON is decreased mostly by the data set's hierarchicallity, which is only 0.5278.

Table 5. Compatibility of data object D_1 and all data structure types.

Data structure type	Compatibility
RDBS	0.8688
XML	0.7152
JSON	0.6974
RDF	0.8688

5.2 JSON

The second data set consists of the same information as the first one, but it is organized hierarchically in accordance with the JSON format:

```
{
  "coursesByArea":{
    "swimming_pool":{
      "description":"This is an example long description of a swimming
pool",
      "courses":[
        {
          "day":1,
          "time":"15:00",
          "period":"2019-03-01 - 2019-06-30"
        },
        {
          "day":2,
          "time":"14:00",
          "period":"2018-09-01 - 2019-02-26"
        }
      ]
    },
    "football_pit":{
      "description":"",
      "courses":[
        {
          "day":3,
          "time":"15:30",
          "period":"2018-09-01 - 2019-02-26"
        }
      ]
    }
  },
}
```

```

    "tenis_court":{
      "description":"Tenis",
      "courses":[
        {
          "day":2,
          "time":"15:00",
          "period":"2019-03-01 - 2019-06-30"
        },
        {
          "day":5,
          "time":"14:40",
          "period":"2019-03-01 - 2019-06-30"
        }
      ]
    },
    "coursesByPeriod":[
      {
        "range":"2018-09-01 - 2019-02-26",
        "courses":[
          {
            "day":1,
            "time":"15:00",
            "area":"swimming_pool"
          },
          {
            "day":2,
            "time":"15:00",
            "area":"tenis_court"
          },
          {
            "day":5,
            "time":"14:40",
            "area":"tenis_court"
          }
        ]
      },
      {
        "range":"2019-03-01 - 2019-06-30",
        "courses":[
          {
            "day":2,
            "time":"14:00",
            "area":"swimming_pool"
          },
          {
            "day":3,
            "time":"15:30",
            "area":"football_pit"
          }
        ]
      }
    ]
  }

```

The metadata warehouse is built in six steps again. The following steps list only the differences from the first data set.

Step 1: The location specification is based on sub-object names and numbers in square brackets (in case of array items), separated by a slash.

Step 2: The JSON data objects have no explicit mechanism of interval referencing, therefore the "data_reference" table is empty and omitted.

- Step 3:** The size of the data object is calculated from the text length without delimiting parentheses or brackets but including all whitespace inside. Therefore, the size is not always equal to the sum of all children object sizes.
- Step 4:** As mentioned in step 2, the amount of hierarchicallity in the JSON data structure type is always 1.
- Step 5:** The amount of structuredness is calculated analogically to the previous dataset. The JSON format includes a lot of whitespace and delimiting characters. A sum of children data object sizes instead of the size of the data object itself is used.
- Step 6:** This step applies AIT the same way, including inner whitespaces and delimiters.

The compatibility is shown in Table 6. The metadata warehouse in the Appendix B. It shows, that even though this data set is JSON, it is more compatible with XML (due to unstructured child object occurrence) and therefore the selected data structure type is not optimal. Also, the very low amount of information 0.2426 (high redundancy) of the data makes it very incompatible with RDBS data structure type.

Table 6. Compatibility of object D_1 and data structure types.

Data structure type	Compatibility
RDBS	0.5614
XML	1
JSON	0.9661
RDF	0.9661

5.3 Data transformation

The transformation study uses results from the first and second data set as the initial and final state of transformation. The data from the relational database is transformed into a JSON document, which is a common process, e.g. in web-based technologies.

Table 7. Part of metadata warehouse containing initial and transformed objects.

object_id	parent_object_id	size	structuredness	hierarchicallity	information _amount	location_specification
1	NULL	201	0.8233	0.8967	0.8571	rdbs/
2	NULL	1764	0.9412	1	0.2426	json/

The changes of metrics values can classify this transformation as a slight structuralization (number of structured data objects and their levels has increased due to the created redundancy), hierarchization and denormalization (S, H, DN transformation type area in Figure 1). This can be written as the following vector:

$$\begin{aligned} \rightarrow_t(D) &= [\Delta s(D), \Delta h(D), \Delta i(D)] = [0.9412 - 0.8233; 1 - 0.8967; 0.2426 - 0.8971] \\ &= [0.1179; 0.1033; -0.6145] \end{aligned}$$

From this, the transformation length can be calculated (D_0 is the first row and D_1 is the second row in Table 7):

$$t(D_0, D_1) = \frac{1}{\sqrt{3}} \sqrt{0.1179^2 + 0.1033^2 + (-0.6145)^2} = 0.3661.$$

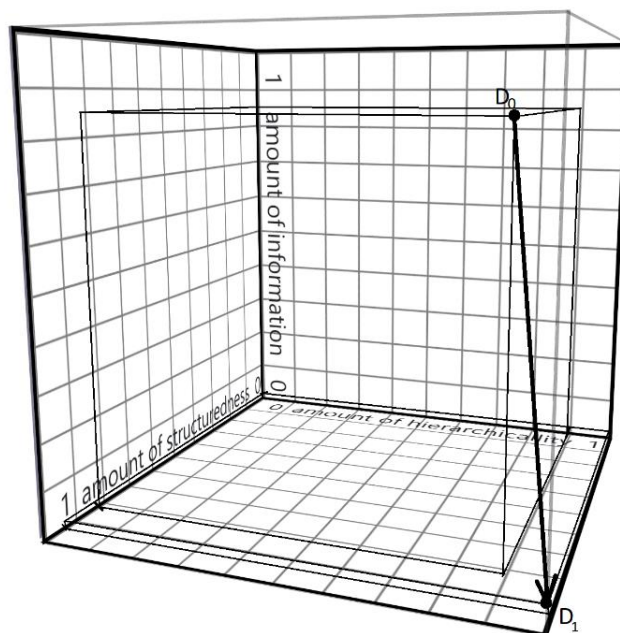


Figure 4. Data transformation in a three-dimensional space.

The length is approximately **one third**, which means a lower level of complexity and therefore a less expensive (demanding) data transformation process. This corresponds to the compatibility of the first data set and JSON type, which is 0.6974 (approximately **two thirds**), as shown in Table 5, which is inverse to the transformation length. Also, guidelines for points D_0 and D_1 have been added to help to imagine the position of the points (this paper does not allow inserting an animation of a rotating space, which would be more illustrative).

6 Discussion

This paper opened the topic of data structure type and its corresponding data heterogeneity and calculation possibilities. It proposed a basic theoretic method of characterizing data and their structure types by a **triad of metrics**. These metrics values can be used to compare data with each other and to analyze data transformation process complexity.

Transformation process complexity can also be predicted by calculating the compatibility between data and data structure type. The calculated values can be stored in a metadata warehouse and allow a further application of statistic methods. The transformation process can be classified into six basic types and multiple combined types.

The triad of values and their changes have shown the ability to visually represent the state of a data object, the optimal area of data structure type and how the values change direction in three-dimensional space. The calculated compatibility can detect a non-optimal storage design.

Using this approach can help to answer the question asked in the Introduction:

1. Data can be stored the more easily by XML or JSON the more hierarchical it is. The higher the amount of structuredness is, the more suitable it is for the JSON format. The example case study has shown the hierarchicallity difference.
2. Data objects with the highest amount of hierarchicallity are most compatible with the XML format. When the example case study hierarchized the data from the relational database, the amount of information strongly decreased – a redundancy was created.
3. Data unsuitable for relational databases are data with a high redundancy (low amount of information) and a low amount of structuredness. The process of data normalization and structuralization can require the input of an expert.
4. The most dynamic parts (child objects) of a data object can be detected by simple size volatility. The triad of metrics can explain the reason for size change (redundancy, low structuredness etc.).
5. This is partially explained in answer 4 – size is measured alongside the three metrics. The source of the size increase can be traced inside the metadata warehouse from root to smaller child objects.
6. The most redundant data parts can be detected by combinations of child data objects whose parent data object has a low amount of information despite the fact that the amount of information of the child objects is significantly higher – such objects might be mutually redundant. This can be seen in parent object 5 and child objects 6 and 10 in Appendix B.

The example case study demonstrated usage on simple data sets, which experimentally evaluated the usability of the described methods, which was the objective of the paper. It showed a very typical situation, when non-hierarchical data (RDBs) are hierarchized into JSON, which creates redundancy.

The main limitation of these methods is their dependency on an expert, who needs to understand the data and to design the calculation and data warehouse storage process – no full or partial automatization method has been found, but it probably can be further designed for typical data structure types. Also, using algorithmic information theory on very large data can be computationally demanding. The metrics also showed that there can exist some mutual dependency between them (especially the amount of information and hierarchicallity), but a high level of hierarchicallity is not the only reason for redundancy and normalized data can also be hierarchical.

The designed methods can be used in **data modelling** as well as in part of the **data mining process** – it can help to select an optimal data structure type or evaluate the optimality of an existing data storage and provide feedback. It can help to **reveal problematic areas** where data volume increases due to redundancies caused by inappropriate hierarchization or unstructured storage of structured information.

A continuous and automated data analysis applying the described method can be used to **monitor any base of data** (not only a database system itself) for a longer time. This process has a similar principle to OLAP storage, but it focused mostly on database engineers, developers, and data curators rather than business.

Most processes in software development perform some kind of **data transformation**. Calculated compatibility can help to select the optimal structure type. Transformation length indicates the real complexity of the transformation. This can help to select the direction of transformation and to decrease the complexity and the overall cost of the transformation process. Despite possible external influences on the selection of data structure type, this method can bring calculation-based arguments to the discussion about structure selection.

Additional Information and Declarations

Funding: The research has been partially supported by project no. 18-23964S administered through the Czech Science Foundation (GACR).

Conflict of Interests: The author declares no conflict of interest.

Data Availability: The data that support the findings of this study are available in Appendix A and B.

Appendix A

Metadata warehouse of the first data set.

object_id	parent_object_id	size	amount of structuredness	amount of hierarchicality	information_amount	location_specification
1	NULL	201	0.8233	0.5278	0.8571	/
2	1	99	0.6868	1	1	/area
3	2	68	0.3546	1	1	/area/1
4	2	13	1	1	1	/area /2
5	2	17	1	1	1	/area/3
6	3	13	1	1	1	/area/1/name
7	3	54	0	1	1	/area/1/description
8	4	12	1	1	1	/area/2/name
9	4	0	1	1	1	/area/2/description
10	5	11	1	1	1	/area/3/name
11	5	5	1	1	1	/area/3/description
12	1	42	1	1	1	/period
13	12	21	1	1	1	/period/3
14	12	21	1	1	1	/period/4
15	13	10	1	1	1	/period/3/from
16	13	10	1	1	1	/period/3/to
17	14	10	1	1	1	/period/4/from
18	14	10	1	1	1	/period/4/to
19	1	60	1	1	1	/course
20	19	12	1	1	1	/course/1
21	19	12	1	1	1	/course/2
22	19	12	1	1	1	/course/3
23	19	12	1	1	1	/course/4
24	19	12	1	1	1	/course/5
25	20	1	1	1	1	/course/1/day
26	20	8	1	1	1	/course/1/time
27	21	1	1	1	1	/course/2/day
28	21	8	1	1	1	/course/2/time
29	22	1	1	1	1	/course/3/day
30	22	8	1	1	1	/course/3/time
31	23	1	1	1	1	/course/4/day
32	23	8	1	1	1	/course/4/time
33	24	1	1	1	1	/course/5/day
34	24	8	1	1	1	/course/5/time

Appendix B

Metadata warehouse of the second data set. Columns with “existence” are omitted due to space requirements and same values as first data set.

object_id	parent_object_id	size	structuredness	hierarchicallity	information_amount	location_specification
1	NULL	1764	0.9412	1	0.2426	/
2	1	961	0.8498	1	0.3330	coursesByArea
3	2	934	0.6348	1	0.3298	coursesByArea/swimming_pool
4	3	54	0	1	1	coursesByArea/swimming_pool/description
5	3	827	0.4898	1	0.3047	coursesByArea/swimming_pool/courses
6	5	114	0.4898	1	0.8070	coursesByArea/swimming_pool/courses[1]
7	6	1	1	1	1	coursesByArea/swimming_pool/courses[1]/day
8	6	5	1	1	1	coursesByArea/swimming_pool/courses[1]/time
9	6	15	0	1	1	coursesByArea/swimming_pool/courses[1]/period
10	5	114	0.4898	1	0.8421	coursesByArea/swimming_pool/courses[2]
11	10	1	1	1	1	coursesByArea/swimming_pool/courses[2]/day
12	10	5	1	1	1	coursesByArea/swimming_pool/courses[2]/time
13	10	15	0	1	1	coursesByArea/swimming_pool/courses[2]/period
14	2	190	0.4898	1	0.4246	coursesByArea/football_pit
15	14	0	1	1	1	coursesByArea/football_pit/description
16	14	137	0.4898	1	0.7591	coursesByArea/football_pit/courses
17	16	114	0.4898	1	0.8421	coursesByArea/football_pit/courses[1]
18	17	1	1	1	1	coursesByArea/football_pit/courses[1]/day
19	17	5	1	1	1	coursesByArea/football_pit/courses[1]/time
20	17	15	0	1	1	coursesByArea/football_pit/courses[1]/period
21	2	537	0.7963	1	0.4246	coursesByArea/tenis_court


object_id	parent_object_id	size	structuredness	hierarchicallity	information_amount	location_specification
22	21	5	1	1	1	coursesByArea/tenis_court/ description
23	21	266	0.7449	1	0.4511	coursesByArea/tenis_court/ courses
24	23	114	0.4898	1	0.8070	coursesByArea/tenis_court/ courses[1]
25	24	1	1	1	1	coursesByArea/tenis_court/ courses[1]/day
26	24	5	1	1	1	coursesByArea/tenis_court/ courses[1]/time
27	24	15	0	1	1	coursesByArea/tenis_court/ courses[1]/period
28	23	114	0.4898	1	0.8070	coursesByArea/tenis_court/ courses[2]
29	28	1	1	1	1	coursesByArea/tenis_court/ courses[2]/day
30	28	5	1	1	1	coursesByArea/tenis_court/ courses[2]/time
31	28	15	0	1	1	coursesByArea/tenis_court/ courses[2]/period
32	1	756	0.9449	1	0.3386	coursesByPeriod
33	32	425	0.9513	1	0.4518	coursesByPeriod[1]
34	33	15	0	1	1	coursesByPeriod[1]/range
35	33	355	1	1	0.3718	coursesByPeriod[1]/courses
36	35	102	1	1	0.7843	coursesByPeriod[1]/courses [1]
37	36	1	1	1	1	coursesByPeriod[1]/courses [1]/day
38	36	5	1	1	1	coursesByPeriod[1]/courses [1]/time
39	36	13	1	1	1	coursesByPeriod[1]/courses [1]/area
40	35	100	1	1	0.8000	coursesByPeriod[1]/courses [2]
41	40	1	1	1	1	coursesByPeriod[1]/courses [2]/day
42	40	5	1	1	1	coursesByPeriod[1]/courses [2]/time
43	40	11	1	1	1	coursesByPeriod[1]/courses [2]/area
44	35	100	1	1	0.8000	coursesByPeriod[1]/courses [3]
45	44	1	1	1	1	coursesByPeriod[1]/courses [3]/day

object_id	parent_object_id	size	structuredness	hierarchicallity	information_amount	location_specification
46	44	5	1	1	1	coursesByPeriod[1]/courses[3]/time
47	44	11	1	1	1	coursesByPeriod[1]/courses[3]/area
48	32	311	0.9168	1	0.6045	coursesByPeriod[2]
49	48	15	0	1	1	coursesByPeriod[2]/range
50	48	241	1	1	0.5311	coursesByPeriod[2]/courses
51	50	102	1	1	0.7843	coursesByPeriod[2]/courses[1]
52	51	1	1	1	1	coursesByPeriod[2]/courses[1]/day
53	51	5	1	1	1	coursesByPeriod[2]/courses[1]/time
54	51	13	1	1	1	coursesByPeriod[2]/courses[1]/area
55	50	101	1	1	0.7921	coursesByPeriod[2]/courses[2]
56	55	1	1	1	1	coursesByPeriod[2]/courses[2]/day
57	55	5	1	1	1	coursesByPeriod[2]/courses[2]/time
58	55	12	1	1	1	coursesByPeriod[2]/courses[2]/area

References

- Bartmann, D., Bodendorf, F., Sinz, E. J., & Ferstl, O. K. (2011). *Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse*. University of Bamberg Press.
- Begg, C., & Caira, T. (2012). Exploring the SME Quandary: Data Governance in Practise in the Small to Medium-Sized Enterprise Sector. *Electronic Journal of Information Systems Evaluation*, 15(1), 3–13.
- Codd, E. F. (1990). *The relational model for database management: Version 2*. Addison-Wesley.
- Florescu, D. (2005). Managing Semi-Structured Data. *Queue*, 3(8), 18–24. <https://doi.org/10.1145/1103822.1103832>
- Floridi, L. (2013). Information Quality. *Philosophy & Technology*, 26(1), 1–6. <https://doi.org/10.1007/s13347-013-0101-3>
- Gangemi, A., Presutti, V., Reforgiato Recupero, D., Nuzzolese, A. G., Draicchio, F., & Mongiovì, M. (2017). Semantic web machine reading with FRED. *Semantic Web*, 8(6), 873–893. <https://doi.org/10.3233/SW-160240>
- Grünwald, P. D., & Vitányi, P. M. (2008). *Algorithmic information theory*. <https://arxiv.org/abs/0809.2754>
- Halpin, T. (2001). *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann Publishers Inc.
- Helland, P. (2017). XML and JSON Are Like Cardboard. *Communications of the ACM*, 60(12), 46–47. <https://doi.org/10.1145/3132269>
- Hutter, M. (2007). Algorithmic information theory. *Scholarpedia*, 2(3), 2519. <https://doi.org/10.4249/scholarpedia.2519>
- Krishnamurthy, R., Naughton, J. F., Shanmugasundaram, J., & Shekita, E. (2001). Dealing with (un) structuredness in XML Data and Queries Using Relational Databases. *DB Seminar at Wise University*. <https://pdfs.semanticscholar.org/acb6/72e6feea4893192c74fc4cf3dcce31b3ad65.pdf>

- Ma, Z., Bai, L., & Yan, L.** (2020). Transformation of Fuzzy Spatiotemporal Data Between Relational Databases and XML. In Z. Ma, L. Bai, & L. Yan (Eds.), *Modeling Fuzzy Spatiotemporal Data with XML* (pp. 123–145). Springer International Publishing. https://doi.org/10.1007/978-3-030-41999-8_6
- Meinsma, G.** (n.d.). *Data compression & Information theory*. 2014. <https://www.yumpu.com/en/document/view/27882302/data-compression-information-theory>
- Morton, J.** (Ed.). (2014). *Big data: Opportunities and challenges*. BCS, The Chartered Institute for IT.
- Musca, S. C., Kamiejski, R., Nugier, A., Méot, A., Er-Rafiy, A., & Brauer, M.** (2011). Data with Hierarchical Structure: Impact of Intraclass Correlation and Sample Size on Type-I Error. *Frontiers in Psychology*, 2, 74. <https://doi.org/10.3389/fpsyg.2011.00074>
- Närman, P., Holm, H., Johnson, P., König, J., Chenine, M., & Ekstedt, M.** (2011). Data accuracy assessment using enterprise architecture. *Enterprise Information Systems*, 5(1), 37–58. <https://doi.org/10.1080/17517575.2010.507878>
- Oren, E., Möller, K., Scerri, S., Handschuh, S., & Sintek, M.** (2006). What are semantic annotations. *Relatório Técnico*. 9, 62. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.7985&rep=rep1&type=pdf>
- Pokorný, J.** (2010). Databases in the 3rd Millennium: Trends and Research Directions. *Journal of Systems Integration*, 1(1–2), 3–15. <https://doi.org/10.20470/jsi.v1i1-2.25>
- Ramel, D.** (2015). Relational Databases Still Reign in Enterprises, Survey Says. *Enterprise Systems Journal*. <https://esj.com/articles/2015/04/23/database-survey.aspx>
- Shanmugasundaram, J., Shekita, E., Kiernan, J., Krishnamurthy, R., Viglas, E., Naughton, J., & Tatarinov, I.** (2001). A general technique for querying XML documents using a relational database system. *ACM SIGMOD Record*, 30(3), 20–26. <https://doi.org/10.1145/603867.603871>
- Shannon, C. E.** (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1), 3. <https://doi.org/10.1145/584091.584093>
- Song, E., & Haw, S.-C.** (2020). XML-REG: Transforming XML Into Relational Using Hybrid-Based Mapping Approach. *IEEE Access*, 8, 177623–177639. <https://doi.org/10.1109/ACCESS.2020.3026006>
- Šperková, L.** (2014). Unstructured Data Analysis from Facebook Banking Sites. *Acta Informatica Pragensia*, 3(2), 154–167. <https://doi.org/10.18267/j.aip.44>
- Vodňanský, D.** (2016). Entropy-based hierarchization of relational data structures. *Journal of Systems Integration*, 7(4), 25–34. <https://doi.org/10.20470/jsi.v7i4.275>
- Vodňanský, D.** (2020). *3D data metrics visualizer*. <https://danielvodnansky.github.io/3d-data-histogram/>
- Vodňanský, D., & Zamazal, O.** (2016). Study on Graph Metrics over Linked Open Vocabularies and OntoFarm Collections. In *Proceedings of the 7th International Conference of Knowledge Engineering and Semantic Web, KESW 2016* (pp. 1–2). Prague University of Economics and Business.
- Wellenzohn, K., Böhlen, M. H., & Helmer, S.** (2020). Dynamic Interleaving of Content and Structure for Robust Indexing of Semi-Structured Hierarchical Data (Extended Version). <https://doi.org/10.14778/3401960.3401963>

Editorial record: The article has been peer-reviewed. First submission received on 6 March 2021. Revisions received on 15 April 2021 and 2 May 2021. Accepted for publication on 2 May 2021. The editor in charge of coordinating the peer-review of this manuscript and approving it for publication was Stanislava Mildeova .

Acta Informatica Pragensia is published by Prague University of Economics and Business, Czech Republic.

ISSN: 1805-4951
