PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

THREE CASE STUDIES FOR UNDERSTANDING, MEASURING AND USING A COMPOUND NOTION OF DATA QUALITY WITH EMPHASIS ON THE DATA STALENESS DIMENSION

Oleksiy Chayka

Advisor:
Prof. Paolo Bouquet
Università degli Studi di Trento

Abstract

By its nature, the term "data quality" with its generic meaning "fitness for use" has both subjective and objective aspects. There are numerous methodologies and techniques to evaluate its subjective parts and to measure its objective parts. However, none of them are uniform enough for exploitation in diverse real-world applications. None of those, in fact, can be created as such, since data quality penetrates too deep into business operations to prevent from finding "a silver bullet" for all of them: it normally goes from representation of real world entities or their properties with data in an information system, to data processing and delivering to consumers. In this work, we considered three real world use cases which entirely or partially cover those areas of data quality scope.

In particular, we study the following problems: 1) how quality of data can be defined and propagated to customers in a business intelligence application for quality-aware decision making; 2) how data quality can be defined, measured and used in a web-based system operating with semi-structured data from and designated to both humans and machines; 3) how a data-driven (vs. system-driven) time-related data quality notion of staleness can be defined, efficiently measured and monitored in a generic information system. Thus, we expand the corresponding state of the art with Application, System and Dimension aspects of data quality.

In the Application context, we propose a quality-aware architecture for a typical business intelligence application in a healthcare environment. We demonstrate potential quality issues implications, including intra- and inter-dimensional quality dependencies, prone to data from early processing stages up to the reporting level.

In the part dedicated to the System, we demonstrate an approach to understand, measure and disseminate data quality measurement results in a context of a web based system called Entity Name System (ENS).

On the Dimension side, we propose a definition of data staleness in accordance with key time-related quality metrics requirements, relying on the corresponding similar notions elaborated by the researchers before. We demonstrate an approach to measure data staleness by different statistical methods, including exponential smoothing. In our experiments, we compare their space efficiency and data update instants predictive accuracy using history of updates of sample representative articles from Wikipedia.

Keywords

data quality definition, data quality assessment, data staleness, quality-aware business intelligence

Acknowledgements

I would like to express my gratitude to all those people who made a significant contribution, direct or indirect, to this work, and to my life.

In particular, I would like to thank:

Paolo Bouquet, my advisor, for elucidating the research path.

Themis Palpanas, **Fabio Casati**, **Florian Daniel** and **Cinzia Cappiello** with whom I started my research on the own "Journey to Data Quality", and who gave me valuable feedbacks on this way, directing my efforts, together with **Bernd Heinrich** and **Mathias Klier** who helped me to connect the dots along data freshness and staleness dimensions.

Yevgeniy Bodyanskiy, **Anatoliy Gvozdinskiy**, **Vadim Gubin** and **Aleksandr Vechur** for their guidance and help to go beyond.

John Mylopoulos and **Gerhard Lakemeyer** for steering towards and in research.

Dan Defend and **Aparna Vani** for their great directions, and not only. I learnt a lot from you. **Heiko Stoermer** and other Okkam team members for their cooperative work on data quality research and development part in the ENS.

Marina and **Andrey**, **Ruslan**, **Lena, Sasha**, **Kolya** and **Alexandra**, together with **Andrey L.**, **Dima P.** and **Vova K.** for their supportive and pervasive friendship.

Natasha Bielova and **Marco Biazzini**, a special couple taking special place in our life.

Laura Maino and **Giuliano Zendri** for showing me a side of Trentino not everybody can experience.

Juan, **Nghi**, **Ali**, **Ravi** and all the guys who kept my mind open off the work and study, especially while being on the field.

Y.V. Rybalko and **A.A. Popov** who dedicated part of their life to support aspiration of young people for knowledge.

Valentina Petrovna for inspiring bio-molecular associations.

Special gratitude I would like to express to **my parents** who gave me life, to **my siblings** with whom I spent most of it, not forgetting **Vitalik** and my beautiful **niece**.

And finally, and most importantly, I would like to express my gratitude to **my beloved wife**. For your understanding and continuous support, for all those months we spent living apart on the way to this dissertation – I dedicate it to you.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1 MOTIVATION AND PROBLEMS

"Business intelligence (BI) systems combine operational data with analytical tools to present complex and competitive information to planners and decision makers" [1] is the generic description of a BI concept. For the decades that BI studies took place, there was a lot of work done and research challenges solved for components of an entire scope of a simplified BI architecture in Web era. As such an architecture, BI can be seen as getting data in a data warehouse (ETL, integrations, matching, deduplication, cleansing, etc.) and getting data out of it (reporting, OLAP, analytics, etc.) [2].

Given the role of output data and its influence on business decisions [3], it is undisputable that such data must be of a proper quality level for data stakeholders. Given that, there must be certain measurement approaches and control mechanisms enforcing required level of data quality. In fact, there are number of ready-to-go solutions from the key BI software providers that allow to measure and control quality of data: MS SQL Server 2012 'Knowledge bases' for data quality services from Microsoft, 'Information server data quality module' from IBM, 'Business objects' from SAP and others. All of these data quality components primarily serve for data deduplication, error correction and standardization with help of natural language processing (NLP) and ranking algorithms. Due to the wide scope of notion of 'data quality', none of those components aim to get objective measures of quality level of data. In fact, whether data is of high quality, depends on its intended or potential use. Hence, it is the data consumers who decide if the data they have is of high quality, and this decision is usually based on a number of measurable data characteristics (quality dimensions), either objective or subjective. Hence, the ultimate questions for researchers in data quality might be 'Who is *the decision maker or data consumer* and how they are going to *consume* the data?'. Without answering this question, the research in most cases is limited to a scope of a particular system, as it was noted in [1].

However, definite pervasive research challenges do exist in data quality. One of those is defining and objectively measuring data quality, which is composed of a set of quality dimensions. Hence, among the key challenges in this area would be: 1) understanding of a compound notion of data quality and 2) elaboration of methods for objective measurement of quality dimensions.

In our work, we focused on one of those dimensions that play an important role in representing time-related aspect of data quality, but it is underexplored in the research community. 'How stale data do we have at hand, and hence, how much can we trust it is still valid?' – is the primary question we want to answer studying quality dimension of staleness. By staleness, as we will show in more details, we denote a linearly growing measure of time from an instant when a data element came out of synchronization with its corresponding real world instance.

A great value of such a time-related measure of quality cannot be underestimated in business intelligence, especially of a real-time nature [1]. In spite of the fact that there are companies already having real-time BI systems, both engineers from industry and researches lack objective data-driven[1] quality measures for the time-related quality dimensions. None of the key vendors of BI software for management of data quality (Microsoft, SAP, IBM) deal with measurement of a quality dimension of freshness, timeliness or another time-related one. However, in some cases, such time-related quality measure in BI can be success-critical and simply vital for the entire company, like Continental Airlines [4], distributing its resources in accordance with current state of flights (on time/delayed/cancelled) for each of the company's traveler or supplier.

[1] this is opposed to system-driven works, where the task is mainly to research various optimization techniques based on information provided by source systems, instead of data originators from the real world

Of course, large size companies like Continental Airlines or Yahoo!, have a number of qualified professionals that implement ad-hoc solutions for specific data quality challenges their companies confront. Nevertheless, lack of widely applicable methods to objectively measure data quality dimensions comes from inability to have a single goal of data usage that the methods should work for. For example, *consistency* usually denotes correspondence of data elements to required schemas or values. Still, schemas or business logic, as well as notion of acceptable degree of correspondence to them which ultimately defines consistency level of data, may vary from one application to another. Another example is *validity*, which primarily denotes an extent or correspondence of data to its original entity in the real world (stored address in a database of a person vs. real address of that person at current time). Due to inability to verify data at hand against most of entities from the real world, this quality dimension's scope is normally limited to semantic validity for reference data (sources of truth) or known rules (valid intervals and values for specific types of data, like date, month, year, etc.).

From the other side, data quality scope is much wider than measurement of certain quality dimensions and their exploitation in context of an information system. Essence of a compound notion of data quality can be best demonstrated while end customer of a system is involved, and the entire process from getting and processing data out of the real world (or an origin source system) to its delivery to the end customer, is demonstrated via a prism of data quality. For these purposes, beyond the context of a particular quality dimension, we study a system environment. It is essential to provide such a scope in order to reduce a gap between scientific approach to measurement of a compound system-wide characteristic of data quality, and ultimate data consumers.

Demonstration of a sample real life business intelligence use case scenario, that involves user interaction, completes a logical data quality assessment triad used in this work by closing a conventional data life cycle – from data origination, to processing, assessment and consumption. This ultimate abstraction level provides necessary contextual aggregation of the other two levels in order to enable judgment by the end customer whether assessed data fits for ones use. At this level, it is also important to understand how data quality dimensions will affect each other, as well as how they will influence perception of the compound DQ concept at the consumer level.

Hence, to tackle the key problems of understanding and assessing a compound notion of data quality, in our work we will consider use cases of three different abstraction levels – Application, System and Dimension, which we explain below.

2 OUR PROPOSITION AND OUTLINE OF THE THESIS

Our ultimate goal in this work is to demonstrate in context of data quality a connection of originators of data in the real world (which are usually humans) with the corresponding data consumers (which are usually humans as well) in the scope of BI applications in the Web era and heterogeneous information systems, providing objective automated measures of a compound notion of data quality with emphasis on one of key time-related and data-driven quality dimensions that is underexplored among researchers and practitioners – data staleness. For that, we explored three different use cases with context of Application, System and (quality) Dimension.

In the **Application** use case, we wanted to understand how the notion of data quality can be understood, measured, and how the measurement results can influence a decision maker which is the main consumer of such measurements in BI. For that, we considered a health care scenario[2], where we highlighted key areas in which we provided data quality analysis and elaboration of quality-aware reports consumable by decision makers. Without such a real world scenario, there is little sense in providing concrete measures for abstract goals. In fact, the biggest limiting factor for the researchers in BI area is its application-oriented nature. In order to make sense of data quality research in context of BI, an instantiation of the above mentioned notions of "operational data" and "decision makers" should be fixed, i.e., which kind of data is used by whom and, ultimately, for which purposes? Nobody can simply answer the latter question, since it is up to 'decision makers' to consider relevant information to drive a concrete company according to their fixed goals, whether making more profit, or increasing their presence in a market, or invading a new one, for instance. Hence, in **Chapter 3 (Application)**, we propose an architecture for a typical BI application in a data warehouse environment supporting execution of a health

[2] original BI scenario is taken from an FP7 research project "Master"

care scenario. In this architecture, we focus on its reporting part which comprehends methods for extraction, processing, aggregation and presentation of data from various sources to data stakeholders such as decision makers. We describe possible reasons for data at the report level to suffer from quality issues, considering human/computer aspects relevant to data processing phases from source systems to aggregation level. In particular, we focus on quality dimensions of consistency, completeness and confidence which we believe are among the most important quality characteristics in such a BI application. For these dimensions, we demonstrate possible contextual quality issues and consequences of their aggregation and propagation to reporting level, where decision makers eventually consume quality-aware reports.

In the **System** context, continuing exploration and aiming at getting objective measurement of a compound notion of data quality in a user-centric environment from the previous chapter, we were looking for data management system that possessed the following properties. It should take both structured and semi-structured data of different granularity from various sources on the Web, integrate it and serve data consumers (users) who are the ultimate assessors of data quality in such systems. Moreover, we wanted to thoroughly explore certain types of data in context of their quality, since not all data is suffered to the same extent from quality problems, as SAP research revealed [5]. In fact, in web-based systems it is the customer data that suffers the most. All those properties we found instantiated in a system called Entity Name System or ENS [6], which manages descriptions of unique entities that may represent potentially anything in the real world.

In **Chapter 4 (System)** we show how one can resolve data quality issues by breaking them down and exploring proper quality dimensions in a system like ENS. In particular, we elaborate and formalize dimensions of consistency, staleness, popularity, distinctiveness and representational importance. We show how one can instantiate the dimensions for measurement at necessary granularity levels (data value / entity / database); we demonstrate how the measured values can be used in the ENS to fulfill one of its ultimate goals – delivery of a high quality data to an end user.

Chapter 5 (Dimension) explores deeper one of the most important but currently underexplored data quality dimension, which we study in the system context of the previous chapter – data staleness. It follows a corresponding chapter of state of the art where we present an overview of existing definitions and measurement methods of this dimension, seen from both data and system perspectives. While proposing our notion of data-driven staleness dimension, we confront it with a set of key requirements that each objectively measurable data quality dimension should possess (e.g., feasibility of its measurement, interpretability of measures, etc.). After setting the notion of staleness, we propose base methods to objectively measure it: naïve averaging, shifting window and exponential smoothing. For their accuracy evaluation, we used history of updates of articles from an online encyclopedia – Wikipedia. We also explore possible options for delivery of staleness measurement results to a subject consuming them (which can be a user or a software component, for instance). We provide parametrical analysis of cost of system resources for usage of one or another delivery method.

Generalizing approaches for measuring of a time-related data quality aspect, we propose metrics for evaluation of overall performance of the algorithms measuring this dimension.

2.1 OUR CONTRIBUTIONS

The main contributions provided in this thesis falls into data quality definition and assessment areas. In particular:

- we define the notion *of data quality* and discuss key *quality dimensions* for one of typical quality-aware BI applications; we discuss *intra-dimensional effects* of sample quality dimensions of completeness, consistency, confidence for aggregated and non-aggregated reports;

- we propose a *report quality aware data warehouse architecture*, together with a notion of *quality-aware reports* in a BI environment, that aims at computation and propagation of quality metadata to the report level, considering user quality feedback; we discuss related aspects of *propagation and aggregation of potential quality issues* from source systems up to the report level in a data warehouse.

- we propose for a web-oriented information system an *approach for definition, measurement and usage of data quality dimensions* dedicated to characterize specific quality aspects of incoming

data (dimensions of consistency and distinctiveness) and outgoing data (dimensions of popularity, representational importance and staleness), covering entire data lifecycle from data originators to consumers;

- following current state of the art and in accordance to key requirements to objectively measureable data quality notions, we define a *data-driven notion of staleness*;

- we propose different *space-efficient methods for measurement of data staleness*, comparing their update prediction accuracy we examined on articles update metadata from Wikipedia;

- we explore possible *options for delivery of staleness measurement results*, providing a comparative parametrical analysis; for possible types of staleness measurement algorithms, we propose *efficiency evaluation metrics*.

Chapter 2
State of the Art

A classical concept of "product quality" established in the era of industrial revolution [7]*, less than a century after served as one of origins for concept of "data quality". Some approaches for understanding of quality were common for both of them. However, certain aspects of data use in a virtual world coupled with its ubiquity in a wide range of applications, led to such problems that quality assurance industry normally did not have. One of those is certification and standardization of real world product vs. data product. In the real world, an extensive corpus of various standards and certification methods has been developed since the beginning of industrial era. In spite of few attempts to come up with data quality standards (like ISO 8000), there is little one can done in standardization in this area. For example, can someone set a pervasive standard for data freshness or data completeness? Even if treated separately for different types of applications (like safety-critical, operational, etc.), it is nearly impossible to have those standards universal.*

Given that, researchers mainly focused on data quality in its context: e.g., data duplication elimination, uncertain data management or data synchronization between various types of systems (caching, replicating systems, etc.). Numerous approaches and techniques were proposed to deal with those and other problems in data management community. However, data quality per se can be seen as a separate field of study, research in which is driven by user and data needs instead of system requirements, like in the database community. This made us provide user and data-centric research and experiments we will demonstrate in this thesis. In the following section, we give the relevant state of the art served as basis for our work.

1 DATA QUALITY AND QUALITY DIMENSIONS

Researches of various fields looked at problems of data quality (DQ) differently. Statisticians began addressing those problems in the late 60's, trying to resolve duplicates in statistical datasets. In the 80's management-oriented research started investigation of methods for detection and quality problems elimination in data manufacturing systems. In the beginning of 90's computer scientists began research of definition, assessment and improvement of data quality in databases and data warehouses. As for now, they concentrate in the following main research streams if the DQ field:

1. Definitions of DQ dimensions;
2. DQ analysis (root causes analysis of DQ);
3. DQ assessment (current DQ level assessment and definition of a gap between current and required DQ level);
4. DQ improvement (solution for a DQ problem; monitoring of its effectiveness).

An application of data mining algorithms for the DQ assessment and improvement (data auditing) was introduced by [8]. Apart from these key challenging directions, specificity of DQ application raised such issues as:

5. DQ mining (application of data mining algorithms to derive "quality data");
6. DQ visualization (representation of the data quality to a user).

From the methodological point of view, DQ dimensions and methodologies form a basis for the following major research issues in the DQ:

1. *Models* to represent data, data schemas and business processes of an organization;
2. *Techniques* to provide algorithms, heuristics to solve a given DQ problem;
3. *Tools* and *frameworks* to solve DQ problems and to provide DQ services.

Given available DQ tools and techniques, DQ methodologies support the selection of the most efficient approaches to DQ assessment and improvement while data quality dimensions are the components by means on which a compound and context-dependent notion of data quality can be better evaluated for fitness to a certain system or business process.

Since the middle of 90's computer scientists started active research in the DQ field. Many dimensions to capture various quality characteristics of the data have been identified by them (for example, some of about 180 dimensions proposed by pioneers in the DQ of computer science [9] see in a list in Figure 1).

Ability to be joined with	Ability to download	Accessibility	Ability to upload
Acceptability	Access by competition	Adequate volume	Accuracy
Age	Adequate detail	Alterability	Aestheticism
Auditable	Aggregatability	Availability	Believability
Breadth of data	Authority	Certified data	Clarity
Clarity of Origin	Clear data	Compactness	Compatibility
Consistency	Completeness	Confidentiality	Compressibility
Convenience	Conciseness	Creativity	Integrity
			etc.

Figure 1. Extraction from a list of quality dimensions identified by [9].

Each quality dimension, like one of those presented in the figure above, intends to describe a specific aspect of a compound notion of data quality.

Authors in [10] propose an information quality model based on a concept of data quality as "fitness for use", that is based on a user-centric view. However, most researches in the DQ field providing quality dimensions analysis ([11], [12], [13], [14], etc.) define them either in context of their projects (context dependent works where the dimensions play specific practical role; for example, too narrow notion of consistent representation defined in [9] as "the extent to which data is presented in the same format") or in terms of the most important features of "relevant" dimensions (based on public surveys or theoretical investigations; for example, objectivity is defined in [9] as "the extent to which data is unbiased, unprejudiced and impartial"). In the latter case, "relevant" means the most significant dimensions for a generic task in a wide range of applications.

As comparative analysis (based on [15]) shows, both practitioners and pure theorists consider a limited set of dimensions as the most significant one (however, sometimes giving a bit different notions

for the same dimensions). It should be noticed that up to now there is no firm definitions for the meaning of each dimension as well as there is no agreements on the set of important dimensions to use. This is because subjective nature of definition of data quality itself. Some authors define it as "fitness for use" [9] while others see it as "the distance between the data views presented by an information system and the same data in the real world" [16]. The later definition makes it difficult to assess the quality in case of data acquisitions from sensors. Another reason for absence of standard definitions of data quality is specific requirements for usage of data in particular types of applications.

Table 1. Quality dimensions addressed by major researches in DQ field, adapted from [15].

Dimension	[16]	[9]	[17]	[18]	[19]	[20]
Accuracy	X	X	X	X	X	X
Completeness	X	X	X	X	X	X
Consistency	X	X	X	X	X	X
Timeliness	X	X	X	X	X	
Currency	X		X			X
Volatility	X		X			
Interpretability	X	X	X	X		
Ease of understanding		X		X	X	
Reliability	X		X	X	X	
Reputation		X		X	X	
Relevance	X	X			X	
Accessibility		X	X	X		
Security		X	X			
Value-added		X		X		
Appropriate amount of data		X			X	
Availability			X		X	
Response time			X		X	

The table above has been populated with those quality dimensions found at least in two of the selected representative works, among which are:

- [16] that defined data quality dimensions using mapping functions from real world to information system. The mapping bases on ontological model.
- [9] that made their list of the dimensions and defined them from user perspective by interviewing data customers.
- [17], authors in which have concluded their list of dimensions working in the project "Foundations of Data Warehouse Quality". User role in a data warehouse environment was taken as a classification basis for that list (in particular, Design and Administration Quality, Software Implementation Quality, Data Usage Quality and Data Quality).
- [18] that represents empirical quality assessment from 2 perspectives: 1) automatic DQ assessment (conformance to defined business rules); 2) physical DQ assessment (ensures accuracy of data values by comparing them with real values by observations, surveys, confirmation with physical objects/processes owners, etc.).
- [19] that defines data quality dimensions and querying methods for molecular biology information systems (MBIS). In particular, they describe querying plans that assure quality of the resulting data received from corresponding sources. Their more recent work [21] comprehensively considers completeness model that mediators use in query answers planning. Such usage of mediators to plan getting a high quality data may be a part of data quality management in BI.
- [20] that defines the quality metrics connecting context independent characteristics with context dependent feature – data utility. In their framework authors revise definitions of quality metrics along with assessment techniques to incorporate contextual issues.

To focus on its specific aspects, various approaches to stratify a compound notion of data quality were proposed. For example, authors in [22] show separation of quality dimensions into extensional (data values) and intensional parts (schema of the data). Taking for the basis empirical generic definitions of

[9], some advances have been also done in definitions, interpretations or applications of those quality dimensions. In [23] authors have extended applicability of dimensions outlined in [9] to operational environment that produces the data, separating quality dimensions into four groups: intrinsic DQ, accessibility DQ, contextual DQ, representational DQ.

To represent a proper classification of the approaches for data quality dimensions definitions from the works above, we will use the following classification attributes proposed by [15]:

1) Modeling view on data – the data perspective taken for definitions of quality dimensions:
   a. *schema* (intensional definition of data, such as definition of a table),
   b. *format* (data representation, format of actual values),
   c. *instance* (extensional definition of data referring to the object from real world).
2) Measurement view on data – shows point of view for data quality assessment:
   a. *process* (assessment of processes producing the data, e.g. data acquisition from sensors, analysis of manual entry of corporate data etc.),
   b. *system* (consideration of the whole information system, e.g. in the case of distributed systems consideration of timeliness of data delivery to a destination node etc.),
   c. *product* (particularly related to the data and its perception from the end-user point of view and thus include subjective dimensions as *ease of understanding*, etc.).
3) Approach to dimension definition – shows methodology exploited by authors to derive and define the quality dimensions. Among those are:
   a. *intuitive* (dimensions are defined driven by a common understanding of the field),
   b. *theoretical* (definitions are based on a new theoretical model),
   c. *empirical* ones (dimensions are derived after a practical experiments, interviews, etc.).

From the selected representative six works, two of those have been developed for a specific project and thus are *project-specific*. The other four are oriented for generic purposes. We denote the groups in Figure 2 as 'project-specific' and 'generic' correspondingly.

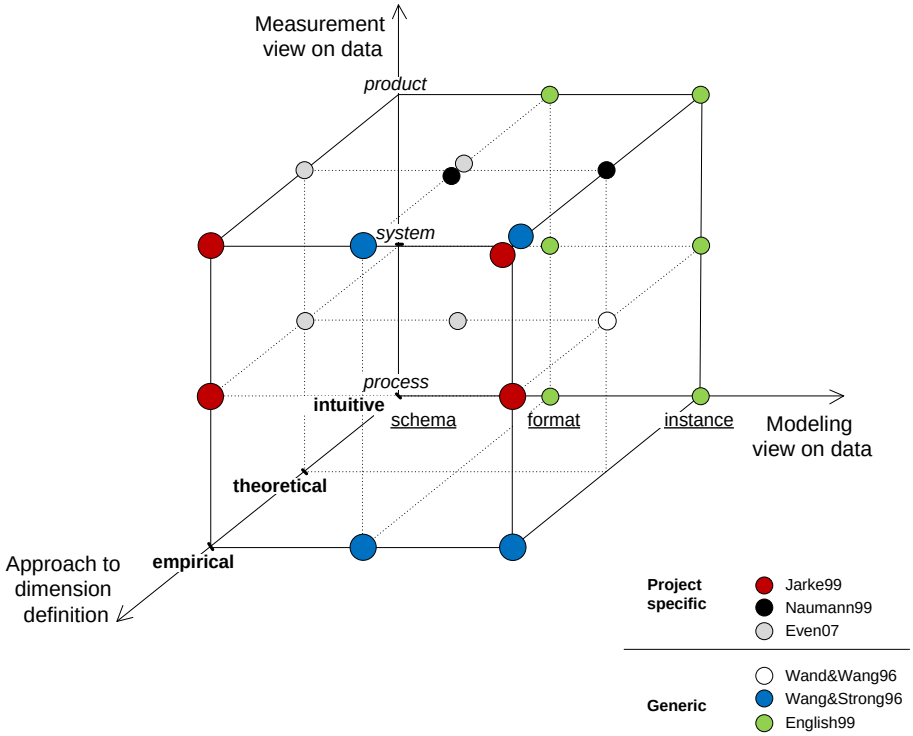


Figure 2. Classification of proposals for quality dimensions, adapted from [15]**.**

In fact, as authors in [9] concluded from the existing data quality works, there are three major approaches to study data quality: intuitive, theoretical and empirical. To the best of our knowledge, most of the existing works in DQ up to now follow one of those approaches for definition and study of data quality dimensions and the corresponding metrics.

For definition of proper quality dimensions for the ENS, we exploited both theoretical and intuitive or deductive approaches, which guided for derivation of relevant quality dimensions by using

one of the corresponding DQ analysis methodologies – AIMQ [13]. In fact, selection of a methodology is not vital for detection of potential quality issues of a system: its key role during potential quality issues analysis step is in guidance of identification and planning of proper management activities of the most common quality issues in a given system. Normally, system and data analyst are able to come to that from their own experience, which is also supposed by AIMQ. From the other side, AIMQ is the only one among other well-known DQ methodologies that drives objective and domain independent quality evaluation [24]. In fact, each qualitative characteristic of a compound notion of data quality, *dimension*, may consist from more than one quantitative ones, *metric*s (for example, consistency dimension may be composed from extensional and intensional parts, as we mentioned before). Those metrics, in their turn, may be measured by means of different methods of using different appliances [25] – see Figure 3 below.

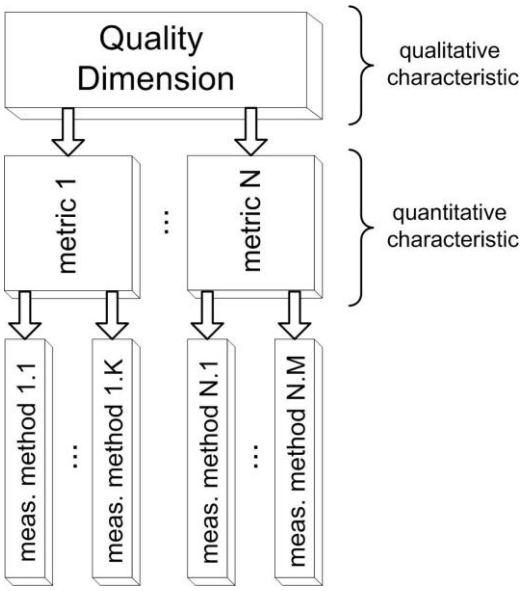


Figure 3. Data quality dimension components.

With a deductive approach and using the quality assessment methodology instantiated for the real-world project we had, for our studies we derived quality dimensions of ***consistency***, ***staleness***, ***popularity***, ***distinctiveness*** and ***representational importance***. The former two dimensions (consistency and staleness) are, by their nature, much more suitable for theoretical research in a data management community (we give an overview of the corresponding state of the art for definition and measurement of those quality dimensions in Section 1.1 and Section 1.2). The other three dimensions, in their turn, are extremely dependent on the application context, at the same time having highly volatile definitions and thus, measurement methods. For example, the dimension of distinctiveness can denote simply edit distance for single data cell values or exploit semantics at the same data granularity (e.g., are the two occurrences of a dog in a text refer to the same specimen or two different ones?). From the other side, authors in [26] see attribute distinctiveness as a reverse function of attribute granularity – it reduces with increase of the granularity. With such a context-dependency of definition of the quality dimension of distinctiveness, as well as for popularity and representational importance, these dimensions were left out of key research activities in DQ area. Given the application context we have, in our work we will demonstrate an approach for definition of all of the identified quality dimensions and the corresponding measurement methods.

In a context of quality evaluation of association rules, authors in [27] proposed quality dimensions aggregative or fusion functions which can be min, max, average, or more complex approaches to account specifics of a particular application. For example, overall freshness of a rule may equal to freshness metric of its least fresh data component. Aggregated consistency, from the other side, can be the highest consistency metric found among all components of an association rule.

In our work, we generalized the idea of fusion functions for each quality dimension identified in our system environment. In particular, we came to a proposal for their measurement and aggregation at higher data abstraction levels, coming from single values, records, to data tables and entire databases.

In the following subsections, we will present an overview of key works for the most elaborated quality dimensions among those we also consider in our system environment – consistency and staleness.

1.1 CONSISTENCY

In a geospatial community, consistency dimension of data quality concept is captured with the notion of *logical consistency* ("degree of adherence to logical rules of data structure, attribution and relationships") which consists (according to the standard ISO 191913 [28]) of the following parts:

- *format consistency*: "degree to which data is stored in accordance with the physical structure of the dataset";
- *domain consistency*: "adherence of values to the value domains";
- *conceptual consistency*: "adherence to rules of the conceptual schema";
- *topological consistency*: "correctness of the explicitly encoded topological characteristics of a dataset".

Besides the latter (topological) type of consistency, conceptual (including referential integrity), domain and format consistencies are considered to be the core parts of the data quality notion of consistency in a data management community, both by researchers and practitioners.

Some of pioneers in DQ research refer to consistency in terms of *data representation*, defining it as an extent of data format homogeneity [12], [9]. They propose to measure it as a difference of 1 and ratio of violations of a specific consistency type to the total number of checks for consistency. Hence, this measure ranges from 0 (data is completely inconsistent for the given checks) to 1 (data is consistent for the checks provided). Sometimes consistency can be seen as *correspondence of metadata to the actual data described*, as in a hybrid database and file system data management approach [29]. In its wider scope, consistency measures *equivalence of data in various systems*, applications and processes for making data equivalent [30], [31], [32]. The equivalence means representation by the same data of the same fact. Along with improving consistency, authors in [30] propose to reduce data redundancy across multiple tables or databases by means of manual inspection. A similar notion of consistency gives DAMA dictionary of data management: "the degree to which data values are equivalent across redundant databases" [33]. Authors in [16] examine consistency in terms of data values as well. In particular, by consistency they consider one to one mapping of an information system state to a corresponding state of the real (world) system.

Semantic consistency in [34] refers to the extent of the same meaning of a same concept (compare 'worker' vs. 'employee'); *structural consistency* in their work is an extent of data representation in the same format (values for 'date' field, for instance). *Structural consistency* can also be seen as conformance of attribute value to specific type of data, its length and pattern [35].

Authors in [36] describe consistency as a fact of accordance of *constraints* in data with the intended ones, at schema and/or data level. Due to the wide scope of the notion of consistency, authors note that one of the options to clean data and make it consistent is detailed manual or computer-aided data analysis for identification of the quality issues. In [37] authors consider consistency as correspondence by a database to a certain class of constraints that can be discovered by conditional functional dependencies (CFDs) [38] which are an approach to infer dependency rules in a database table between its fields of the form $\{F_1,F_2,\ldots\}$-> $\{D_1,D_2,\ldots\}$. For example, in a sample dataset, country code (F_1) and area code (F_2) may unambiguously define city (D_1). Another consistency measurement approach [22] suggests to use database *integrity constraints* – for relational data – and *data edits* which are rules manually defined for the input data – for non-relational data.

One of recent summaries on research for data quality dimensions [39] describes consistency in a wider scope for constraints satisfaction: it is a degree of correspondence by a data managed in a system to specific constraints (including database integrity constraints) or business rules. Those rules can be specified manually by user or derived automatically from training data. The measure of consistency is a fraction of tuples in a relation that satisfy the imposed constraints. Some authors propose to see consistency as "the degree to which data satisfies a set of integrity constraints" [40] and measure it as number of elements (values, records, relations) that violate referential integrity [41], [42].

1.2 TIME-RELATED QUALITY DIMENSIONS

In spite of the fact that time-related data quality dimensions have been studied by computer scientists ([43] explored how to ensure required currency level of materialized views) even before main data quality research in this area began ([9], [44]), they still lack a comprehensive measurement methodology that can be applied in practice.

Though both academics and practitioners find a time-related quality dimension among the most important ones [13], there is still need in common understanding and defining ubiquitous notions of

those. Because of this fact, such time-related quality terms as freshness, timeliness, currency, up-to-dateness, age, staleness, obsolescence may be used to denote the same quality problems (up to antonymous equivalence). Bouzeghoub and Peralta [45] have presented a structured view of some of those terms in the table below.

Table 2. Freshness factors and metrics according to [45].

Factor	Metric	Definition
Currency	Currency	The time elapsed since data was extracted from the source (the difference between query time and extraction time).
	Obsolescence	The number of updates transactions/operations to a source since the data extraction time.
	Freshness rate	The percentage of tuples in the view that are up-to-date (have not been updated since extraction time).
Timeliness	Timeliness	The time elapsed from the last update to a source (the difference between query time and last update time).

Though Bouzeghoub and Peralta [45] studied freshness-related metrics, they have concentrated on analysis of definitions of data freshness in literature. Measurement of those metrics was out of their scope. One of recent works where authors focus on evaluation of time-related data quality metrics, elaborating a set of corresponding requirements for them is Heinrich et al. [46]. We discuss satisfiability of our notion of data-driven staleness metric to those requirements in Section 3.

Authors of other related works (expanded list of the notions, definitions and measurement methods of those works see in Appendix 1) measure data freshness using known update rate of a monitored element [47]. In [48] they study incorporation of a freshness parameter into OLAP queries processing. The more recent work of Guo et al. [49] presented an integration of currency and consistency requirements of users into SQL queries; Labrinidis and Roussopoulos in [50] proposed an algorithm that allows users to get data based on their performance and freshness preferences. Qu and Labrinidis [51] introduced Quality contracts as a way to express user preferences for *speed* of data delivery vs. delivery of *fresh* data; in [52] authors propose a model that allows user to specify freshness constraints and read out-of-date data within a serialized transaction. For example, if prices for an item at an auction change, user may want to get quick response with old prices if staleness of the data will not be beyond a specified threshold. In our work, instead of aiming at satisfaction of user freshness preferences expressed in user queries, we measure current staleness of monitored elements based on prediction of the most recent updates that should have taken place in the past.

Golab et al. [53] define the notion of data staleness ("a difference between time *t* and the timestamp of the most recent tuple in table *T*") and study this quality measure for scheduling of updates for a real-time warehouse, focusing on management aspects of data synchronization between two systems, rather than on data-driven characteristics we pursue in our work.

Definition of freshness and approaches to measure it for data replication was given by [54], where authors study a problem of maintaining a cache at required level of currency, consistency, completeness and presence (defined by users). Cho and Garcia-Molina [55] have shown how web crawler should update its cache to keep it as fresh as possible. Following this work, Denev et al. [56] proposed a framework for assessment of a cashed web page data quality by the means of two dimensions: blur (expected number of changes a crawler would see from a source site) and coherence (number of unchanged pages a web crawler caught for a site snapshot).

Xiong et al. [57] described how to plan updates for data objects with known validity intervals. Akbarinia et al. [58] showed an approach to keeping replicated data in P2P systems consistently updated and fresh; in [59] authors demonstrate lazy replication technique with freshness guarantees by processing timestamps. In one of the most recent works, Xiey et al. [60] provide a method to ensure that outsourced database system correctly performs update operations (and hence, has data fresh). For this purpose, they have a twofold approach: adding timestamps to data signatures or adding fake operations of insert, delete or update. Afterwards they check for correctness of execution by an outsourced DB processing data modified by either of the two above-mentioned ways. Following the defined notion of freshness, we

abstract from data freshness guarantees which are driven by system properties (choice of the most fresh sources, assurance that sources execute update operation correctly, etc.).

In this work, we study the nature of data staleness that is a data-driven characteristic that depends on data element's frequency of updates. Hence, our notion of staleness does not depend on system properties that are usually considered in works studying time-related data quality notions in terms of replication and caching techniques (an overview of those notions with corresponding definitions and measurement methods see in Appendix 1 and Appendix 2).

One of such data-driven and time-related intrinsic notions was given by Ballou et al. [61] – by volatility authors refer to data element validity interval, which also depends on frequency of updates in the real world, but not state of currency. For instance, this property well measures historical facts which normally are nonvolatile. However, in our work, we aim at measurement of extent of non-synchronization of frequently changeable data elements stored in an information system and referencing a real world state (see Chapter 4 and Chapter 5 for more details) rather than stable ones. For the same reason, we cannot adopt a notion of temporal inaccuracy of Redman [11] which measures data aspects (how accurately current data element corresponds to the source of truth) rather than temporal ones (how much time passed since current data element represented source of truth). In fact, temporal inaccuracy is considered as a special case of accuracy dimension, taking into account a binary correspondence of a data element (whether a data element up to date or not) while considering time aspects along this dimension.

Cho and Garcia-Molina [62], [63] gave one of the most relevant time-related quality notions and freshness and age, that we adapted in our work, together with notion of currency proposed by Cappiello et al. [64] in the context of time-related factors evaluation problem in a distributed multichannel information system; Cho and Garcia-Molina studied a problem of keeping a cache (replica) of certain web data as fresh as possible by means of discovery of update rate (frequency of updates) of the data in a source system(s), and by the following selection of a corresponding strategy to efficiently synchronize the cache with the source system(s). In our work, we will demonstrate in Section 2.5 of Chapter 4 and in Chapter 5 how our data-driven notion of staleness comprehends and extends the semantics of the most relevant existing notions of freshness, age and currency elaborated by Cho and Garcia-Molina, and Cappiello et al. in a system-driven context.

2 QUALITY VIEWS AND QUALITY PROFILES

There are different notions of quality views and quality profiles in different works. For example notion of quality view introduced in [65] and [66] expresses users' quality processing requirements in terms of workflows. They propose a framework that captures views of users involved into processes of data creation (in particular bio tests) on data quality. Such views can be compiled, embedded into data processing environment and computed during execution of data processes. In spite of the fact that we want to use similar notion of quality views but from the other side of the entire data life cycle – to take processed data from data warehouse and let user tune a personal view to it – we want to use a concept described in these works of data quality-aware accountability of user experience via personalized views. The authors also show how user (data expert) can not only specify personal quality model, but also how one can derive effective criteria for data acceptability.

Data quality profile in [67] denotes a universal view of a particular database rather than a custom view that we aim to have for our quality-aware BI approach. An important notice is made in [67] regarding the fact that quality metadata should have its full derivation history (how, why, by whom an assessment was provided, etc.) to present to the end user not only results of quality assessment, but to give a basis for reasoning on how much user may trust to the results. This fact is important for users in our data warehouse since it gives them ability to manually tune quality metadata parameters (creating own quality profiles or quality-aware views [3]) to see the data assessed in a custom way.

For our BI scenario, we will use both terms of quality views and quality profiles as synonyms representing user personal knowledge and custom settings for quality metadata.

3 DATA QUALITY FOR BUSINESS INTELLIGENCE

There have been number of works done in data quality with respect to business intelligence and in particular to data warehousing environment, including healthcare area [68] we refer to in the corresponding chapter. However, we are interested in cross-domain works in both data quality and BI that introduce such areas of new challenges as *quality-aware data provenance*, *quality-aware identity resolution*, *uncertainty in databases* due to quality issues and others. Here we will overview related works in these areas.

Since data stored in a database is usually a result of data retrieval and processing stages, for better understanding potential data quality issues we are interested in its provenance [69], which can help us to better understand how a certain piece of data is obtained and why do we have it in a database [70]. A particular interest for the data quality scope we have in recent provenance works dealing with incomplete and probabilistic databases [71] and provenance metadata management [72]. In some cases, a custom metadata annotation and propagation management system for a relational database may be needed to handle quality metadata. Such a system, coupled with the corresponding extension for a query language (SQL) was presented in [73].

Identity resolution is another closely related to data quality area. It essentially tries to find a mapping of an entity representation from the real world or an external information system to the one represented by one or more data items at hand. The key deduplication techniques via record linkage with help of various clustering algorithms are presented in [74]. Rule-based approaches [75], [76] ease the task of mapping at the price of user intervention: creation of a number of training rules must be human-supervised. Another practical approach for the entity resolution is efficient aggregation of approximate match ranks for a given query for a relational DB [77].

Understanding the nature and approaches to deal with uncertainty in data is a twin problem for data quality issues resolution. Number of works proposed a data warehouse design which is driven by data quality and uncertainty [78], [18], [79], [80]. These approaches, in particular, aim to set quality goals, driving the corresponding quality issues resolution processes. Recently, there have been works focusing on design of relational databases dedicated to management of uncertain data [81], [82], [83], [84]. In these works, data in the databases is annotated with a probabilistic value mainly indicating degree of confidence that the data is of high quality and in particular, it is accurate and valid. Various techniques are then proposed to handle those annotations to answer user queries. However, those approaches do not provide solutions to go beyond queries. For example, for generation of a report one should solve a problem of annotation aggregation for data of different granularity like cell, tuple or table. Moreover, different quality measures (consistency, validity, precision, etc.) may and normally, will be used for the annotations. Corresponding representation of a combined measure of those at a report level is another challenge that fell outside of scope of those works.

As data warehouses usually acquire data from various sources, they often meet problem of data fuzziness. Authors in [85] has separated the root causes of data fuzziness into the following categories:

- **Uncertainty**: indicates truthfulness of available information. For example, temperature of a patient *may* be 38 C. Such statement may be a result of missing data or data received from a sensor that has certain precision or it may be a result of prediction or intentional data concealing (for example to preserve privacy), etc.
- **Imprecision**: non-specificity of available data. For example “the temperature outside is between 30 and 33 C” or “doctor says that the patients doesn’t have flu” or “doctor says the patient is ill with unidentified disease”.
- **Vagueness**: model includes vague concepts. For example “patient Smith is ill”.
- **Inconsistency**: conflicting data. For instance, patient requests reimbursement for an operation 300 euro while hospital reports 250 euro as total costs for that type of operation.
- **Ambiguity**: lack of common semantics, agreement on domain terms, etc. For example, in a report the cost of provided operation may be regarded differently without explicit indication whether it includes taxes or not.

To operate with fuzzy-related data notions in the subsequent sections, we will use widely accepted in the literature term “*uncertainty*” that for simplification includes all of the abovementioned aspects of fuzziness.

4 OPEN ISSUES

Understanding of data quality effects on the end customer was always among the key DQ issues. In business intelligence this issue has a particular impact on the entire organization if DQ aspects are ignored or their effect is misunderstood. Intra-dimensional quality relations, as well as their aggregation and propagation to a reporting level, are among another important issues to study in this area.

In an information system context, DQ focus shifts to a deeper understanding of a composed notion of data quality, measurement individual quality dimensions (like consistency, staleness, etc.), and efficient exploitation of the measurement results.

Having diverse set of notions for time-related quality dimensions (see Appendix 2), the DQ community is still open for new approaches for their understanding (especially while looking beyond a system-driven context), efficient measuring and the measured result exploitation and delivery to the end user or a system component.

This dissertation aims at addressing all of the abovementioned open research issues, providing theoretical contribution and experimental evaluation of the proposed principles and methods.

Chapter 3
Data Quality in Business Intelligence Applications

To start our deeper exploration of data quality concepts, we would like to first outline a bigger scenario where from the one hand, data quality penetrates all the stages of operational data environment, from data origination to processing and consumption, and from the other hand, it invades all those stages in such a transparent manner that normally one may not care of the DQ aspects in small or medium BI applications. However, the effect of those aspects may result in misrepresentation of the real world business related facts.

In this chapter, we will explore some of key data quality dimensions one may consider in a typical business intelligence application, as well as their interdependency aspects, and extent of their potential misrepresentation of real world facts during aggregation and at reporting level. We also study personalization aspects of interactive reports presenting quality-aware data, and demonstrate a corresponding possible BI data quality-aware architecture.

1 INTRODUCTION

Nowadays, Business Intelligence (BI) solutions play more and more critical role for the companies using it in their everyday business. The role of BI is particularly important for applications where human health depends on efficiency of a solution, on quality of its design and ultimately (from an IT perspective), on quality of data produced for decision makers like doctors, pharmacists and provisioners. In spite of the fact, that it is not a secret that low quality data from any of a source system may lead to low-quality data represented in reports (provided the data from problematic source is not processed for quality enhancement and it is a part of a final report), a proper design of a BI solution mitigating or solving low-quality data problem is still an enormous problem these days.

Data quality problems in BI applications are becoming more and more pervasive. This is mainly due to the fact that today many organizations tend to employ data warehousing techniques in their everyday data processing routines. They get data from various sources, clean it and transform into a usable for operation form (completing an ETL process), loading data into a data warehouse (DW) or a database. With a wide variety of types of operational environments and data types of source systems, ETL processes become more and more complex as well. Such a complication inevitably leads to risks of having errors in sourced and aggregated data, not speaking of challenge of providing correct data quality enhancement (for example, data cleaning and validation) procedures.

Consider, for example, a BI solution comprising DW, ETL tools, which put data into the data warehouse from various source systems like hospitals or laboratories. For getting data out of the DW, there should be querying, aggregation and maybe data mining modules which will feed report generation component of the BI solution. As a result, reports generated after such a process (depicted in Figure 4) may have such a degree of data quality that they may be unusable for a decision maker, or, even worse, they may be misleading because of misrepresentation of factual data.

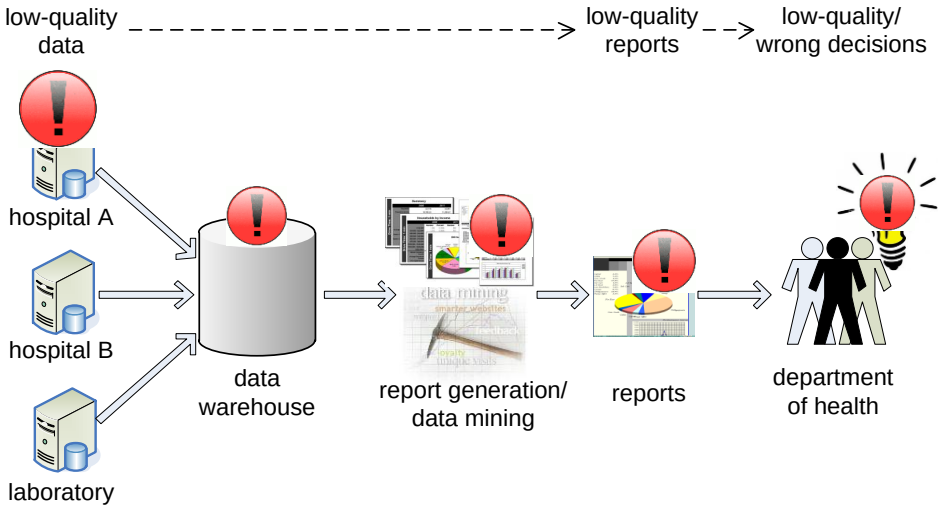


Figure 4. The risk of low-quality data propagation in a healthcare BI.

Usually, a healthcare scenario looks even more complicated. Consider, for example, a patient with suspicious flu symptoms visiting a doctor. After an examination, doctor sends patient to a laboratory for more sophisticated test results of which are further sent to a hospital. Eventually, all the information related to that patient, including legacy records that may be processed from various data carrying mediums (paper or digital) by information processing units (which may be a separate company), is then sent to a central data warehouse of a Ministry of Health (see Figure 5).

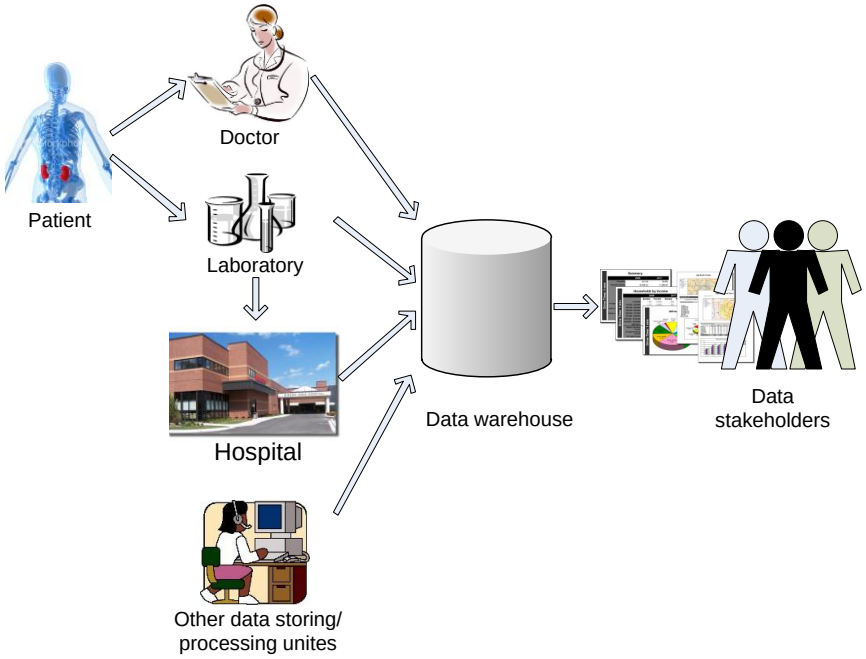


Figure 5. Healthcare BI scenario.

A problem of having quality data seems to be one of the most critical for data stakeholders. Even having an incoherent view on the same data may already lead for far-reaching consequences, like in the following use case.

The Ministry of Health wants to analyze number of flu cases for last season(s) in order to take corresponding measures for the next one. For that, they need accurate estimates (to a statistical error boundary, which is usually negligible compared to data quality errors) to predict demand in doctors' visits, all flu-related drugs and other medical resources for the next season. From the other hand, it depends on a BI infrastructure whether data from all the sources will be coherent, i.e., treatment of the same patient from the real world as a unique entity in all the information systems, treatment of the same diagnoses pervasively (have an agreement on notions like 'flu' and 'influenza'), etc. More to that, data may be simply incomplete (due to delay in delivery from a source), outdated or incorrect (there is always human factor). When the estimates are provided based on low-quality data, insufficient resources for mitigating flu during next season may be negotiated. As a result, manufacturers may be unable to satisfy non foreseen ad-hoc requests for the necessary drugs which are not reserved in advance; doctors and laboratories may be not well-prepared for an excessive number of patients, providing only limited number of vaccinations.

In the above scenario, not only problems with quality of data from source systems may lead to wrong decisions, but unawareness of such problems do the same. By knowing that report at hand has specific data quality problems with a certain portion of data affected, data analyst would have more options to consider previously hidden quality data issues for the resulting decisions.

One of the key contributions of this work is in demonstration of an approach to the latter problem – enabling quality degree awareness at the report level. For that, we set a notion of quality-aware reports for a BI application. Such reports aimed at exposure of existing key quality issues which we will discuss in Section 2.2. From the other hand, we have to keep in mind that those reports are normally used at executive level for which information in the reports must be presented in "ready-to-act" way. The stakeholders of such reports are seeking at understandable representation of potential quality issues underlying the data. For example, how useful could be a report indicating 90% consistency and 7 days staleness of its data? Simply exposure of a measured quality degree does not solve the problem.

However, the main aspect (from an IT perspective) we want to explore is computation, association and propagation of a relevant quality metadata (obtained from base data) to the report level. Definite data quality challenges exist in these lines. We want to keep awareness of the key quality issues that may arise in a resulting data (which may be due to errors in source data, erroneous ETL processing or simply because of inconsistent interpretation of a data concept throughout a dataflow), at the same time allowing users to deduce root causes of those. Hence, provenance of data quality issues plays an equally important role for the decisions makers who can not only account existing issues in the source data, but

also take corrective measures for enhancing current data or prevent from error re-occurrence by improving related data handling processes. For example, a report may demonstrate low confidence in a summary of certain procedures from hospital of St. John's. An analyst, while knowing which piece of the report is important for his current decision, may estimate or directly contact the hospital to ensure the estimated figure would realistically reflect real situation, disregarding of reason for the low confidence data.

This example, in fact, shows an interesting but very complex to study feature of quality aspect of data: its subjectivity. The nature of subjectivity becomes apparent while an analyst use quality measures in suggestive manner that drives him to a proper decision based on own knowledge. In some cases, personal knowledge of internal data handling procedures by some organization or individuals says much more than different quality measures and provenance information.

For the quality-aware reporting solution subjectivity aspect means necessity of definition of personal quality-aware views on a data prepared for a report (in contrast to accounting user quality requirements for data processing before generating reports, as [65] and [66] showed using the notion of quality views). Taking into account personal knowledge about data sources or portions of data in reports, analyst should have ability to include or exclude from the consideration necessary portions of data constituting the report. In this way report should provide interactivity for reflection of the required data. It may have a slider indicating level of confidence in the correctness of data represented, with corresponding data coverage. Changing the coverage, an analyst should see to what extent one may rely on correctness of the exposed data, how much data one may miss while relying only on the data of high confidence of correctness and from which sources.

From the other hand, subjectivity aspect means that there may be as many views on data quality aspects as there are users consuming reports. Each of the users, in fact, may contribute to the problem of quality issues identification and representation, by providing a feedback. This can be seen as crowd knowledge, as in the following example. Several users may note that data from hospital of St. John's is questionable for a particular period of time. This valuable information may be then used by a dedicated analyst that can change corresponding processed in a BI architecture, or simply enrich metadata with the provided and potentially verified knowledge.

Another challenge in having quality-aware reports in a BI application is incorporation of quality aspects into data querying and mining procedures. For instance, how to provide a root cause analysis (e.g., using association rule mining) provided that some portions of data is more trusted in terms of accuracy, while other portions of data is known to have less errors.

Given an enormous problem space for the challenges we described, in this work we give an overview of an approach to tie together all the points in a feasible and pervasive BI architecture, focusing on exploration of a relation between data quality in source systems and its proper representation at reporting level. In particular, in this work we present a data quality-aware BI architecture, discussing such issues as: related challenges in implementation of such architecture; selecting, measuring, aggregation and mapping of key quality dimensions from source to report level; modeling of quality-aware views or personal quality profiles.

1.1 A QUALITY-AWARE BI SCENARIO

For demonstration of a quality-aware BI approach we propose in this work, we will refer to a healthcare environment example which is generic enough to combine features of many other relevant BI applications where the approach can be implemented.

To predict necessary medical resources for each season of different flu types, a Ministry of Health collects all relevant data from all regions of a country. In each region, data is collected from hospitals, laboratories, family doctors and other units providing necessary information. The data is then gathered and aggregated in a central data warehouse for further extraction and analysis by relevant stakeholders from the Ministry who make decisions about planning and distribution of the resources to be prepared for a flu season. While Figure 5 depicts this scenario, Figure 4 also demonstrates that problem in at least one data source (hospital) may eventually lead to contamination and misrepresentation of facts at reports level, which endanger objective reality comprehension by the decisions makers, and hence, most probably, it may eventually lead to wrong decisions. This, in its turn, may lead to undesirable human health and economic effects which could be avoided otherwise.

We have to accept that data quality problems are such a reality that may not always be corrected, but in most cases it can be explored and brought to attention. The reasons for data quality problems may

be quite different – from human tiredness and harmful intentions to incorrect design of data processing stages and unhandled software and hardware system failures. For instance, Figure 6 demonstrates possible quality problems, their reasons and possible responsive actions to correct those problems.

Figure 6. Possible data quality problems in a healthcare environment.

In the figure above, table 'Diagnosis' aggregates data related to flu which is received from various sources, like hospitals from Rome, Milan and Trento. A third party (fraudsters) also inserted some records (ID=4) in order to skew the results. Whenever such records are identified (methods for that are outside of scope of our work), for the purposes of enhancement of reports quality, they should be skipped. The same strategy one can apply for erroneous records, like for the one with ID=5, which is known to be sent from Rome, but reporting about diagnosis in a different city (which may be a correct on its own, but has violation of a business rule about non-duplicative records).

Among other types of problems is semantic equivalence of subject concepts, like in records 1 and 2 in Figure 6. The diagnosis given in these records refers to the same disease type which means they should be treated as such instead of two different types of diseases. Another problem we may have is human factor error like typo (record 3 in the figure).

Missing values (record 6) or entire datasets (dashed table with record 7 and onwards which represents expected but not received items by the due time) would probably be the easiest quality issues to identify. Note that in the case with all other types of issues we described above, an analyst, while looking at the data, may put a degree of confidence that one or another portion of data has certain quality issues. This data may be then used in reporting or other purposes. Missing data, in contrast, usually gets simply excluded from the reports or replaced sometimes with its estimations. Nevertheless, reporting a percentage of missing data in all analyzed items for a report, is another valuable insight for a decision making.

The example above demonstrates how some of quality issues originated from source systems may affect quality of data down on the stream (including conventional data mining procedures), and thus, it demonstrates importance of identification and representation of such issues. This allows the analysts to make their decisions better informed, with consideration of those issues as well. Summarizing research challenges highlighted by the scenario described above, we can generalize contributions towards relevant solutions for most of similar BI applications.

2 DATA QUALITY IN BUSINESS INTELLIGENCE

2.1 A COMPOUND NOTION OF DATA QUALITY FOR BI

As many researchers and practitioners concluded, a notion of data quality should be seen in terms of the intended or potential use of it: data is as good as it fits for its use. In spite of the broad notion of data quality, there are number of quality characteristics, or dimensions which constitute this compound notion. Among those are both objective (like 'consistency') and subjective (like 'appropriate amount of data' or 'relevancy') dimensions [9].

Despite of lack of ubiquitous methods for measurement of key objective data quality measures, as well as estimation of key subjective ones, in this work we focus on selection and elaboration for understanding and measurement of those dimensions which support our primary goal – to bring awareness of key quality issues and their extent to decision makers. In particular, we wanted to expose

those issues resulted from human or non-foreseen system factors rather than from BI solution design (e.g., a persistent error in an ETL procedure). For these purposes, we propose to operate with the following quality dimensions: completeness, consistency and confidence. In the following parts of this section we will describe each of these dimensions in detail, while in the following Section 3 we will demonstrate effect of their exposure and mapping at report level. Note that in reality, as [13] suggested, most quality dimensions are related to one or more others. For example, completeness can be studied with relation to time (Appendix 3) compared to static and self-sufficient quality measure normally explored in a research community. However, for better understanding of the basis components of compound quality notions, in our work we focus on studies of individual quality dimensions.

2.2 DATA COMPLETENESS IN BI

Defining dimension of completeness, authors in [12] have separated it into completeness of schema, column and population. The last two parts are more intuitively called in [3] vertical and horizontal completeness metrics (of *extensional* measure), following the definitions of density and coverage constituting notion of completeness in [86]. Representation of schema completeness [3] as *intensional* metadata we will use in our work as well.

Hence, we define vertical completeness as a measure of presence of data values in a column. One can measure it as ratio of existing values in a given column to all the values in the same column, including expected but absent (NULL) or specially encoded values which may present missing or incorrect items (such as '99/99/9999'). Note that in some cases NULL can also represent an important fact of absence of data item due to valid reasons (a person never took a flu vaccination, for instance). While measuring incorrectly missing values, this measure normally implies quality of particular procedures data from a source (e.g., manually entered field of 'vaccination data' is not always correctly translated into electronic format).

Horizontal completeness, in its turn, measures fraction of all expected records in a table to the total number of record present there. It usually measures a fact of presence of an entire procedure, like surgery or vaccination, provided by a hospital. This measure normally implies a wider scope of completeness – whether expected procedures were provided or recorder correctly, or entire data source has data delivery problems. For the latter problem (missing data from an entire source) consider, for example, a case, when only two of three sources in accordance to specifications delivered their data summaries to a central data warehouse by due time. After instant of delivery by last source, any authorized person should be able to aggregate data to see up-to-date summary for all the expected source systems. However, the fact of data missing from some source may skew the aggregated result, or may prevent its production. In contrast, we want to allow the analyst to aggregate and use in other ways the existing data, letting one know about the issue with one of data sources coupled with timing information of that issue for historical consideration or issue resolution activities. In this sense, the problem of completeness becomes closely related to that of staleness– even if a part of a data element or report are out of date, we want to demonstrate this, highlighting places with potential issues.

Thus, the dimension of completeness in a BI application, consisting of the vertical and horizontal completeness measures, represents a fraction of effectively present data to the expected, according to specifications, data in a data warehouse. Figure 7 demonstrates in a sample table the two components of completeness presented above (in addition to consistency issues that will be described in the next part), aggregating data from several hospitals of different cities. This figure also highlights batch availability problem from an entire data source – from hospital of Bolzano.

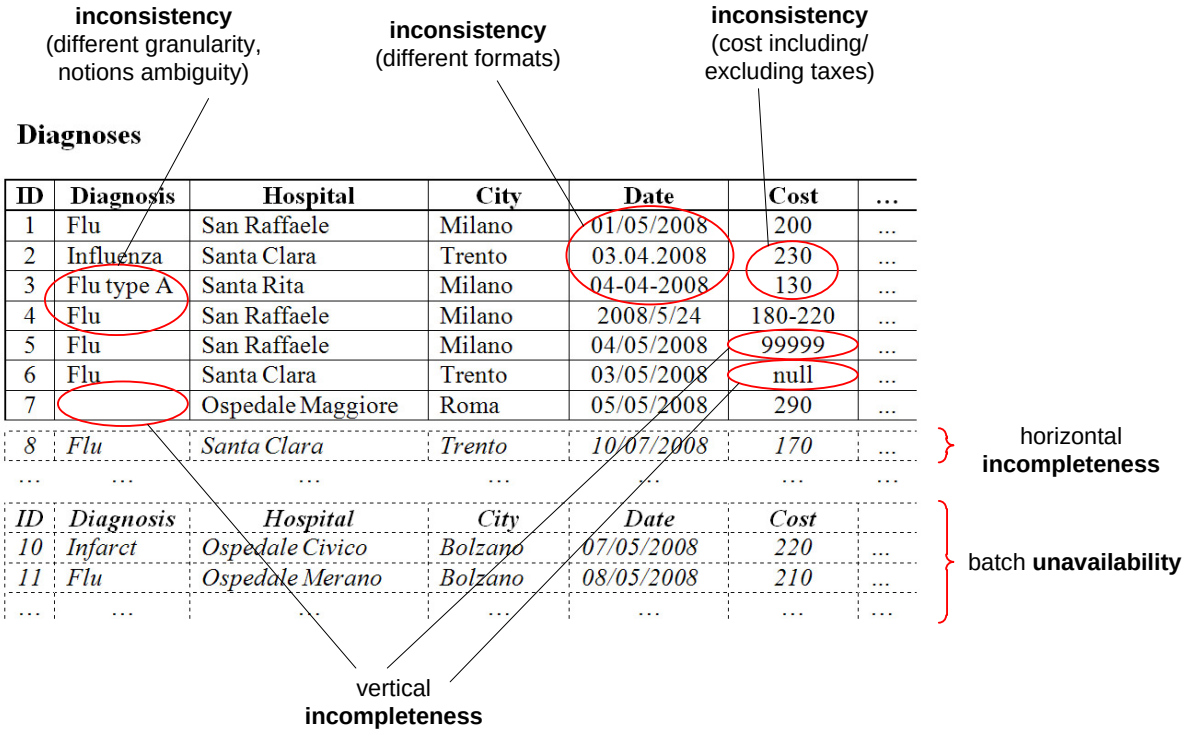


Figure 7. Completeness and consistency problems in the DW (dashed records represent expected but unavailable data)

In this work, however, we did not focus on calculation of intensional (metadata-enriched) or extensional (explicit) measures of completeness, which can be a subject of another research direction [86], [87]. As *intentional* measure of completeness, consider, for example, aggregation of data that may be 90% consistent, 95% believed to be true and 99% precise. Would the final report be complete in terms of data from the real world? It may depend on expert judgment which can enrich missing data or it may depend on expert's (one or more) confidence in additional information, etc. From the other side, a comparison of present data to a given specifications let one have exact measure of *extensional* completeness.

2.3 DATA CONSISTENCY IN BI

In the scope and for goals and limitation for a BI application described above, we consider data consistency as a measure of uniformity and agreement (between sources) on a data present in the same column.

The dimension of data consistency in a BI application has two parts: semantic and syntactic consistency. Syntactic consistency refers to a measure of data format uniformity, like formats of date in Figure 7. Usually, these kinds of issues are detected and corrected (or only denoted as having consistency issues without modification of data itself) during data cleaning stage. Nowadays, such cleaning procedures are provided by various vendors of BI and data management solutions (like SAP, IBM, Oracle, etc.) and they are not interesting for researchers to focus on. However, for demonstration of this part of a consistency dimension refer to 0 where we exemplified it in context of the ENS.

Semantic consistency, from the other hand, measures degree of compliance of a set of data items to specified or implied business rules. Implied means that even if a system engineer would not foresee, for example, that someone may give cost of a procedure excluding implied taxes (in contrast to all the rest users who may follow the assumption of having all the taxes included into the cost value entered into a system), because of a business rule absence for this case this would not mean that the incorrect cost value provided will be consistent with others. This can be resolved with a help of a reference table for cost of typical medical procedures, for instance.

Among other reasons for suffering of this dimension from semantic issues, can be (see Figure 7) inconsistent interpretation of 'date' field that may refer either to procedure of vaccination or diagnosis, usage of different granularity/precision level ('Flu' vs. 'Flu type A'; rounded value of cost etc.) and others.

As in the case with data completeness, this dimension has intensional and extensional components: apart from compliance with business rules [18], [11] (extensional consistency), inconsistencies may occur due to different treatment of the same input data by different persons (intensional side of consistency). An example of the latter case may be determination of severity of a diagnosis with resulting decision on treatment made by family doctor and specialist in a hospital. In this case, a metadata indicating potential inconsistency should be attached to the suggested treatment in order to be considered during actual decision making.

2.4 DATA CONFIDENCE IN BI

In contrast to the quality dimensions of completeness and consistency described above, dimension of confidence has only intensional components. However, it represents an aggregation of several factors related to the underlying data that are important to consider for a decision making. Among those are degree of trust that data value, record, tables or entire set from a source, is of appropriate accuracy, completeness and consistency for relying on it in a decision making process.

In this way, this dimension is a function of underlying key quality dimensions (in our context these are completeness and consistency) coupled with additional expert knowledge about provenance procedures of a data item measured. Knowledge about probabilistic outcome of data integration and entity resolution processes is one of the most important aspects to expose in reports. As the outcome it can be a measure of confidence that two or more items refer or represent (and hence, could be merged) to the same real world entity. Among other reasons potentially affecting data confidence are: lack of trust in data processing proficiency of a source, potential errors or uncertainty due to data cleaning procedures [88], [77], detected outliers [89], [90] and others.

As a result of confidence estimation function, cell value, record, table or data set from a source should be tagged with the measure of confidence, as demonstrated in Trio [82].

2.5 WAREHOUSE QUALITY METADATA

Having set measurement procedures for quality dimensions like those described above (the list of dimensions is not closed), we want to demonstrate potential challenges and approach for association of the resulting measures to actual data in a data warehouse.

In the association task, we want to connect three main components: DQ problem (characterized by quality dimension which has one or more corresponding measures), scope (data at different granularity – from cell values to data sets from a source) and provenance (including such metadata as who, when and how got that data).

Association of metadata with raw data at flat granularity (cell, entire records, columns or tables but nor their fractions) in a data warehouse seems to be a straightforward task, while such an association with data surrogates (normally obtained using filters or aggregations) may uncover some challenges. As surrogate data one can see, for example, a view which is a virtual table in a database serving for aggregation of specified data and exposure it to certain groups of users. Similarly, there can be other virtual data sets identified by a query (i.e., one can get those on the fly during query execution) rather than physical storage. Hence, we would like to associate views or other virtual data sets with existing quality metadata as well. In fact, association of a function, or a SQL query, with quality metadata, provides lots of flexibility and control, as [91] showed. Figure 8 demonstrates how such an approach can be implemented in a data warehouse.

Dimension	Measured value	Associated data	Comments	Author	Date
Vertical completeness	30%	references to a table and a column	…	n/a	04/05/2008
Horizontal completeness, batch availability	20%	data source, batch identifier, missing time window	…	n/a	03/05/2008
Confidence	90%	references to a table, a column and a cell	…	n/a	05/05/2008
Consistency	95%	select Diagnosis from Diagnoses where Doctor = 'John Smith' and Date < '1-1-2008'	…	J. Peter	10/07/2008
Confidence (entity resolution)	80%	references to a pair of tuples candidates for merging	…	P. John	11/05/2008

Figure 8. Example of the quality metadata association with raw and surrogate data in a DW.

Note that commentary information in the figure above explaining various aspects behind quality measurement procedures provided by their authors, is one of keys to understanding by data consumers various aspects of provenance, including reasons and consequences of certain values of quality measures, and even semantics of those, since not every consumer of data is expected to be aware of all the quality dimensions and corresponding measures used by a data warehouse architect, not speaking of their meaning. A description of a quality rule, cleaning procedure and rationale may help data analyst to understand better the resulting report enriched with quality metadata.

3 FROM DATA QUALITY TO REPORT QUALITY

Usually, reports are combination of methods for getting specific data (which are designed in advanced or are ad-hoc queries that one can through on the fly) and the corresponding formatting guidelines for bringing the data into suitable form for consumption by analysts. For demonstration of ideas we discuss in this work, we consider the most widely used table-based reports with basic charts. Hence, we want to understand what kind of quality metadata mapping one will have from database level to report level, including consideration of aggregations provided for reports.

Therefore, we will treat reports as queries or virtual views (for a database) that will eventually be represented in tabular and/or graphical form. Those views are usually pre-computed after completion of required data loading stages, or on the fly at any instant. As we will demonstrate later, it is important to differentiate between non-aggregated and aggregated reports. Aggregated reports are those having at least part of them computed using aggregated functions like average(), sum(), max(), etc.

In the previous sections, we described quality aspects peculiar to data in a data warehouse or database. In this section we want to demonstrate a connection of quality metadata representation at database level and the one at report level, allowing user effectively interact with those reports by consuming quality metadata and also providing corresponding feedback (we describe this in Section 4).

In spite of the fact that in a BI world it is well understood a correlation between quality of incoming data and adequateness of business decisions [18], [11], to the best of our knowledge, effective communication of proper quality metadata at reporting level still has not yet been thoroughly studied. In this section, we try to make a step further to outline a basis which can serve for implementation of a specific quality-aware reporting solution of a BI application; we explore how base data quality aspects affect those at report level. In particular, we want to outline mapping of the quality dimensions that we defined above as important for a BI application (completeness, consistency and confidence) to similar quality properties for reports, keeping in mind a different approach for more straightforward non-aggregated and more complex aggregated reports.

Data completeness issues from a base data are simply carried over to a report level: missing values, records, data sets are represented correspondingly (depending on a report query). Figure 9 demonstrates this case: from a base table (left hand side) with several quality issues, one may select certain data using a SQL queries like those in the boxes on the right hand side in the figure. As one can see, the resulting table for non-aggregated report misses diagnosis from 'Ospedale Maggiore' (tuple 4),

diagnosis record (tuple 6) and entire data set from hospital of province of Bolzano – every quality issue existing in a base table and affected by the query.

However, not all reports may "inherit" quality issues from base table – in some cases one may accidentally query or filter only complete portions of data. In general, this can be applied to all the quality dimensions, but in this work we focus on a problem of handling the quality issues rather than probability of their occurrence etc.

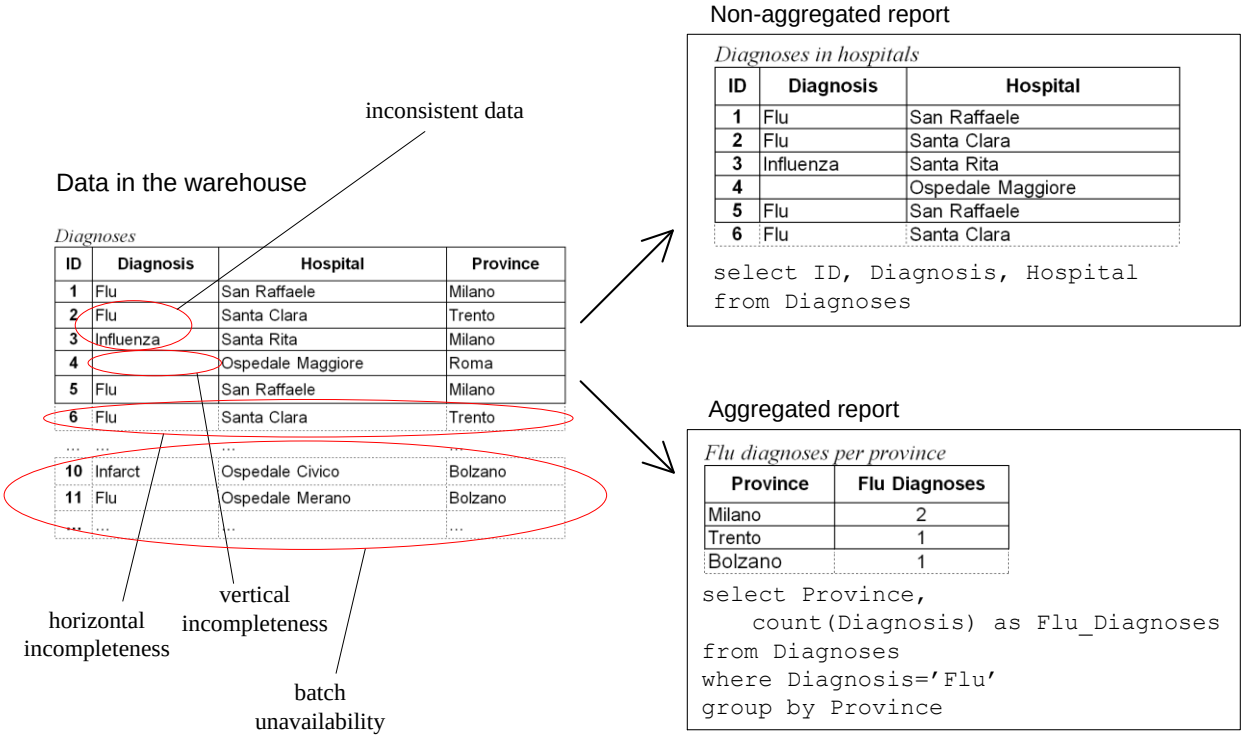


Figure 9. Effects of data completeness and consistency on report quality (dashed lines represent expected but not present data)

For the aggregated reports, completeness issues at report level from base table may only occur when an aggregation function of a query affects its entire scope which has the quality issues in base table. For example, if the function is max(), and the highest value data item of the selectable scope is missing, this will affect the report. Due to their nature, aggregation functions like sum(), avg() or count() are even less tolerant to the missing items in their scope in a base table. This is demonstrated with the example of aggregation report query in Figure 9, where missing value in a cell or missing entire data set affected final report which has omission or misrepresentation of real values. Misrepresentation, on its hand, is not a missing data, but a bias from the real world. In this sense we can say that incompleteness in aggregation functions may affect confidence (in correctness of resulting values).

Hence, completeness dimension at base data level may affect not only same dimension at report level, but also dimension of confidence, however, for aggregated reports (see Figure 10). Later on, we will compare each of the identified quality dimensions at base data level to themselves and the rest dimensions at report level.

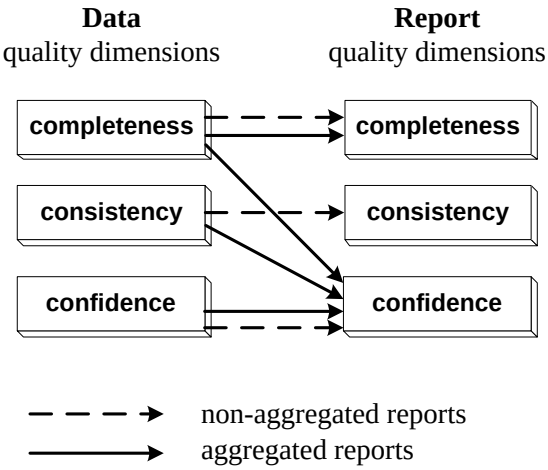


Figure 10. Mapping of raw data quality properties into report quality properties.

To demonstrate the mapping of raw data quality properties into report quality properties, we will use schematic tables with base and the intended corresponding report data, where dashed records represent expected but missing entries, as before. In such schematic drawings, we will show important reporting concepts rather than precise output data and resulting queries as in Figure 9. For the dimension of completeness, for example, the schematic mapping would look like in Figure 11 where incomplete or missing items will result into probabilistic aggregations (using various issue mitigating options) or actual missing items in non-aggregative reports. One of the options to mitigate missing aggregates due to the completeness issues is using the complete base data, highlighting this fact to the end user. Another one may be predictive evaluation of missing items in the reported numbers.

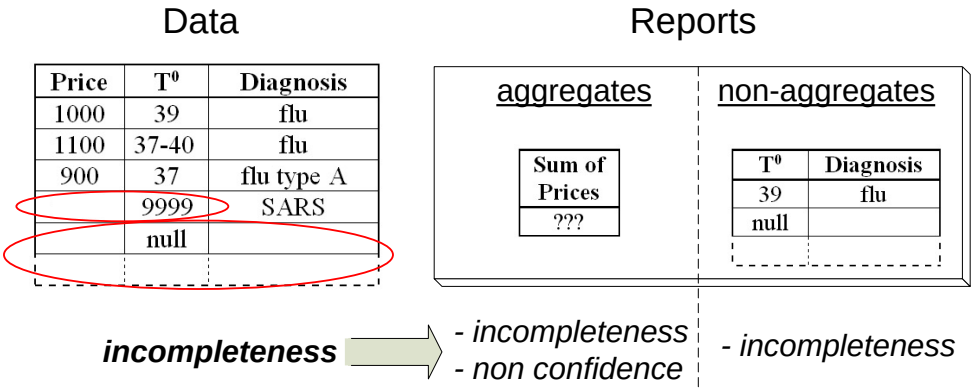


Figure 11. Effect of completeness issues on reports.

Data consistency issues from the base table for non-aggregated reports will have the same representation at the report level – misinterpretation of semantics, or misuse of an abstraction level or measurement units will be represented as such. In our example in the Figure 9 above, the values of 'Flu' and 'Influenza' are simply selected and transferred to the final view.

For the aggregated reports, base data consistency problems usually affect only *confidence* degree of a report. In fact, misuse of measurement units of misinterpretation of data values semantics in most cases will not affect quality dimensions if such aggregation function as count() is used, as in our example in Figure 9. From the other hand, exploiting functions like sum(), avg(), max() may lead to diminution of confidence degree in the resulting values – one cannot be sure if the resulting value account all the relevant items of a base data ('Flu' vs. 'Influenza' will make difference while grouping on Diagnosis field), and if those items are correct in terms of consistency issues we mentioned. For example, sum() over cost values entered including and excluding taxes, will bring a result of low confidence in terms of its correctness.

Schematically, consistency issues from base table would be mapped into report level as in Figure 12 where inconsistent usage of granularity level of a diagnosis in one base table would result into lack of confidence for aggregations querying only expected level of granularity disease ('Flu'), or it may result into same inconsistency issues for non-aggregated reports.

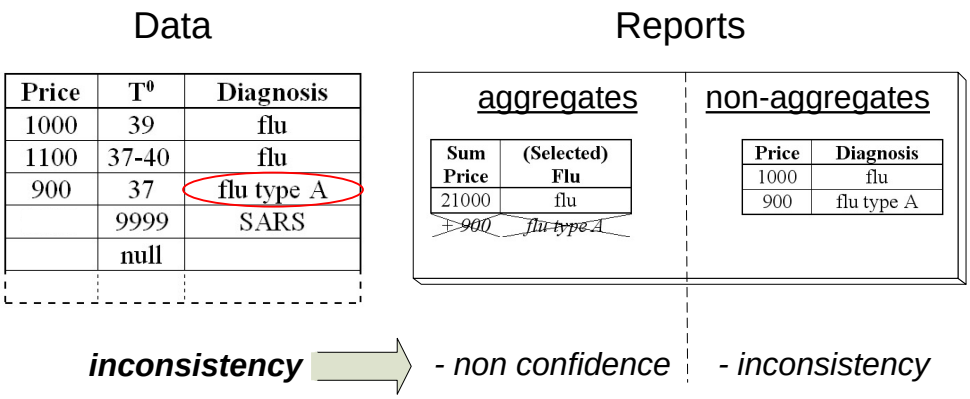


Figure 12. Effect of consistency issues on reports.

Data confidence issues from the base data level directly affect report confidence, whether aggregated or not. For example, Figure 13 demonstrates how a low confidence value of cost "180-220" (entered by a clerk or resulted from a cleaning procedure) is directly transferred to a tuple of a non-aggregated report. Note, however, that standard SQL queries should be rewritten to account quality annotations, range values, etc.

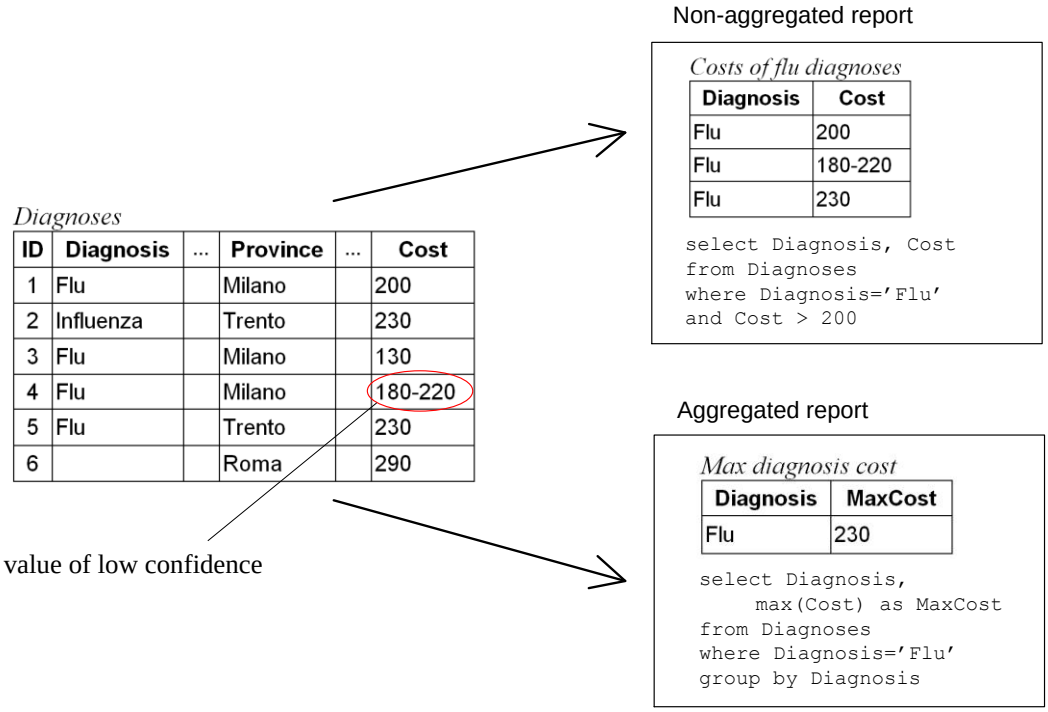


Figure 13. Effects of data confidence on report quality.
Note that the value "180-220" is intended as an indication for low confidence in absence of a precise specification of quality.

For the aggregated report, this value may influence a degree of confidence of a corresponding aggregating function's outcome: in the example in Figure 13, this quality issues could affect result of max() function of a report provided that the range of low confidence values went beyond the highest value of other items in a selection.

Schematic mapping of confidence issues from the base data to aggregated and non-aggregated report on can see in Figure 14. In this figure, volatile (which can be seen as having low confidence) value of temperature for one of entries of diagnoses inserted, is resulting in aggregates into volatile interval for queries inquiring for typical temperature for certain diagnosis, or into a potentially incorrect, but in any case non certain answer for a query trying to get maximum registered temperature for a certain disease. For non-aggregates, it may result into volatile result for a query requiring potential resolution other quality issues, like inconsistent granularity level usage. The query may ask for registered temperature for SARS, which is a kind of flu, but not flu type A. An inference system may infer inclusion of disjoint

terms ‘SARS’ and ‘flu type A’ into ‘flu’, estimate the worst case for existing temperatures for diagnosis of ‘flu’ in the base table, outputting max temperature estimation for SARS as 39-40 (^{0}C).

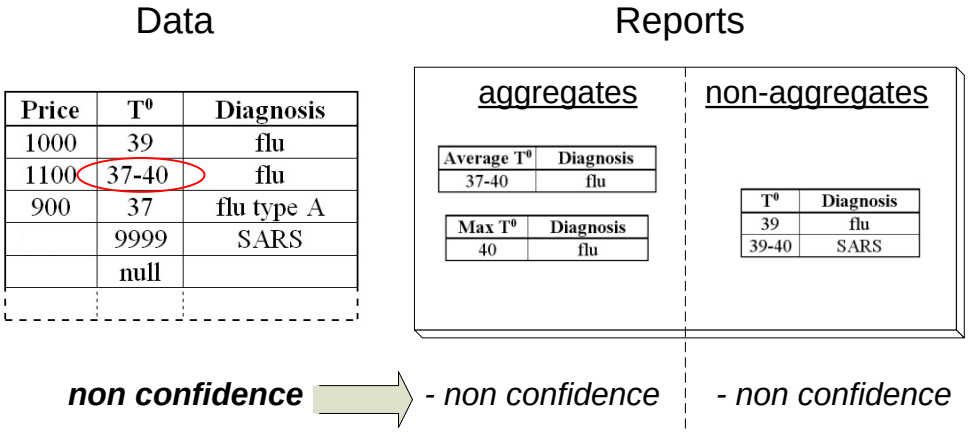


Figure 14. Effect of confidence issues on reports.

Figure 10 summarizes our observations of mapping of quality issues peculiar to raw data in a data warehouse (data level) to the quality issues at report level, using straight selection of aggregation queries. This makes us consider a proper aggregation model for such dimension as confidence, to informatively represent quality issues behind a report user sees. This may include principles of measurement of confidence dimension at base level, methods of aggregation of other quality dimensions coupled with alteration of a query language to account 1) quality metadata bind to data in a DW and 2) calculated quality measures aggregation procedures. While research of most of these issues is heavily dependent on a specific application context (e.g., means of measurement of confidence, as well as interpretation of this dimension), for some of these issues there have been approaches proposed (for example, consideration of quality metadata in SQL queries [49]).

In the following section we will show how a BI architecture may account a user-centric approach, providing one with potentially customized quality-aware reports.

4 USER INTERACTION WITH QUALITY-AWARE REPORTS

Having identified the key dimensions for demonstration of major quality issues to data analysts in the reports of a BI architecture, we want to efficiently visualize the corresponding quality measures and support user interaction with the quality-aware reports. For this, in the following sections we will first describe interactive data visualization aspects and then an entire architecture, as well as its functional representation, of a BI architecture supporting execution of the reference scenario.

4.1 INTERACTIVE DATA VISUALIZATION

Visualization of personalized quality-aware reports in a BI application deals with representation of quality metadata and its effective base data in a way appropriate for user consumption. In fact, definition of “appropriateness for user consumption” is subjective (to a higher extent), and cannot be elaborated for a generic BI application. However, relevant studies in human-computer interaction for each application type should provide proper solutions in each case.

Expanding quality-aware data representation, we claim that the report should also be interactive. This means that analyst should be able to transparently (i.e., without making actual queries to data warehouse, but using intuitive control elements provided in report) slice and dice base data, changing base data coverage by setting acceptable limits on quality measures. For example, analyst may be interested in the portion of data in a warehouse that is of high degree of confidence. Using a quality measure control element (like slider for setting a confidence threshold in Figure 15), one may impose the corresponding query to the warehouse.

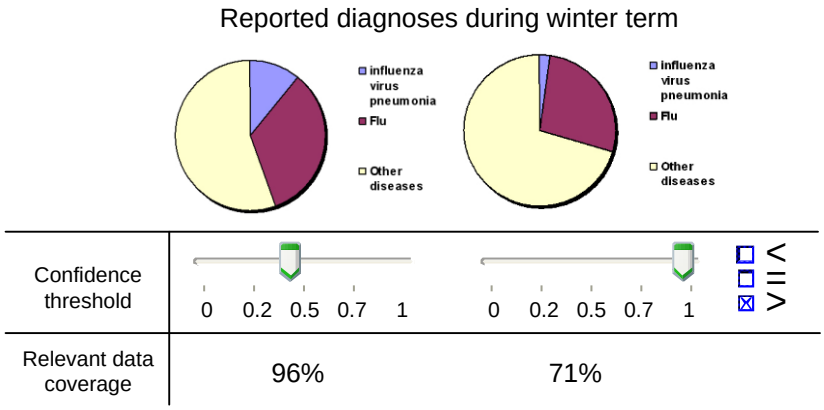


Figure 15. Interactive quality-aware report.

However, limiting output data on a quality measure means restriction in base data coverage – not all the data may satisfy the limits. Representation of the coverage relevant to the imposed limits is as important as representation of data itself. After all, it is up to the analyst to choose how to base own decisions and estimate best or worst consequences of those – either taking into account only a small portion of data of high quality, or nearly complete set of data of lower quality, or considering both of those cases. In this way, user may exploit personal beliefs or knowledge of factors to consider in a decision making, mitigating rigidness of a set of factors (like quality measures) pre-defined on a design phase of a BI system. Same kind of mitigation also applies to user quality profiles – not always it will be enough to say that certain customer is interested only in data that is more than 90% consistent.

4.2 User quality profiles in a BI architecture

The quality profiles are user-specific settings for custom filtering of data to be presented in reports, as well as configuration for its output format. The filtering may be either common for entire data warehouse ("trust data from source A") or report-specific ("don't integrate data which has less than 90% confidence measure").

Both of these types of user quality preferences can be stored in a user profile available in a BI application. Hence, each time user requesting data from the data warehouse, a process responsible for report generation (by potentially exploiting results of data mining quality-driven procedures) is then limiting (by specified threshold for quality measures) and enriching the requested raw data according to the corresponding quality metadata, formatting the report as requested by the user. After consumption of the presented results in an interactive manner (we will give more details on that in Section 4.1), user may alter quality preferences in the profile (including missing data treatment strategy), as well as give quality-related feedback to a quality expert responsible for creation and maintenance of data processing infrastructure in the data warehouse. Alternatively, such users as analysts may directly make ad-hoc queries for raw data in the data warehouse omitting report generation and rendering phase. Figure 16 demonstrates the above scenario of usage of user quality profiles in a BI architecture.

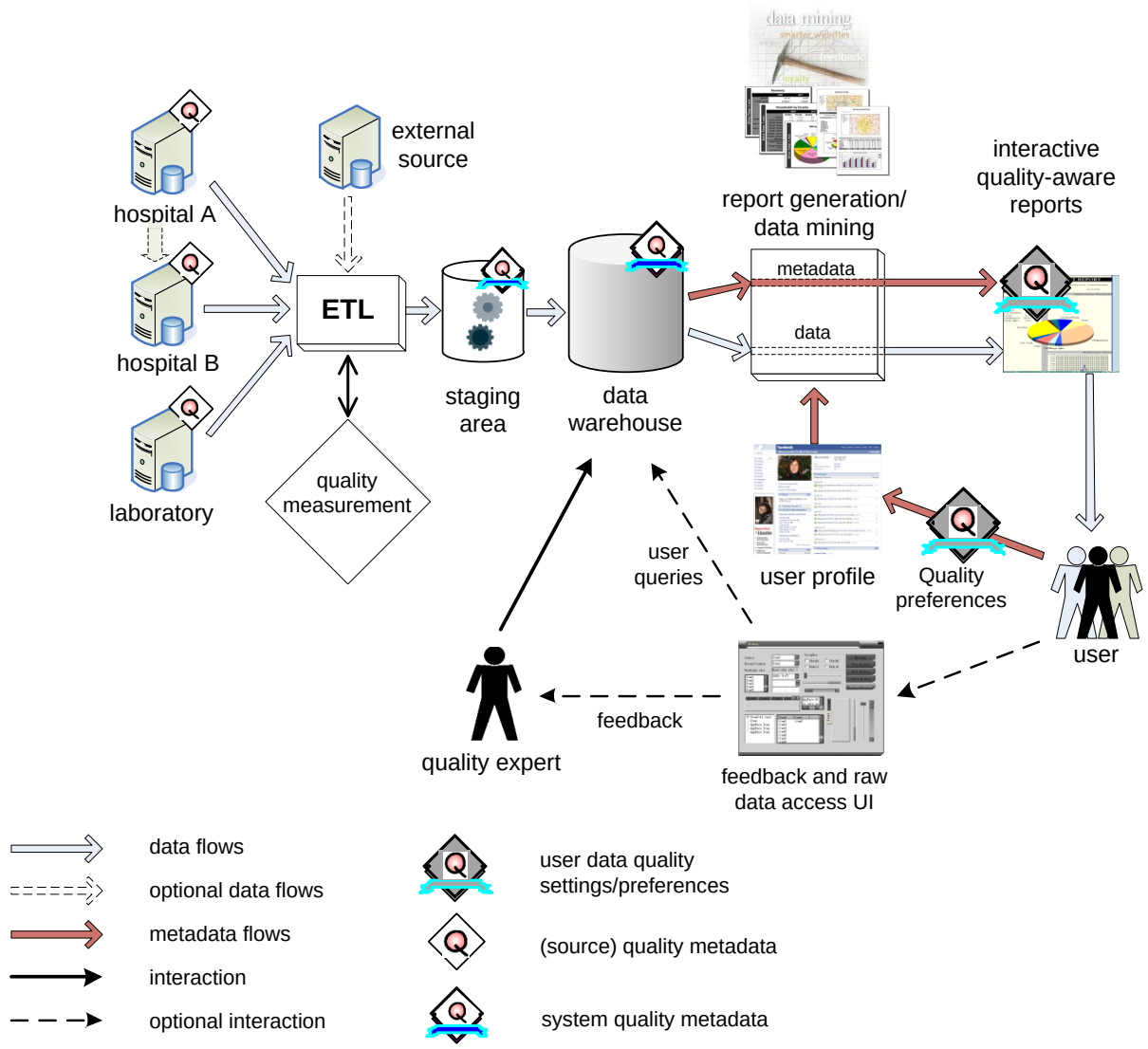


Figure 16. Quality-aware reporting and interaction model in a BI environment.

Another important application of user quality profiles is been a part of monitoring and alerting infrastructure which is responsible for notification of signed up customers about events they are interested in. For example, one may be interested in getting a notification by email or SMS about batch data non-delivery from a certain source by the corresponding deadline. Monitoring of satisfaction of acceptable level of quality measures thresholds is another use case scenario for the monitoring and alerting infrastructure. Such a monitoring and alerting components, as well as data quality measurement, processing and reporting modules are represented in functional decomposition of quality-aware BI architecture in Figure 17.

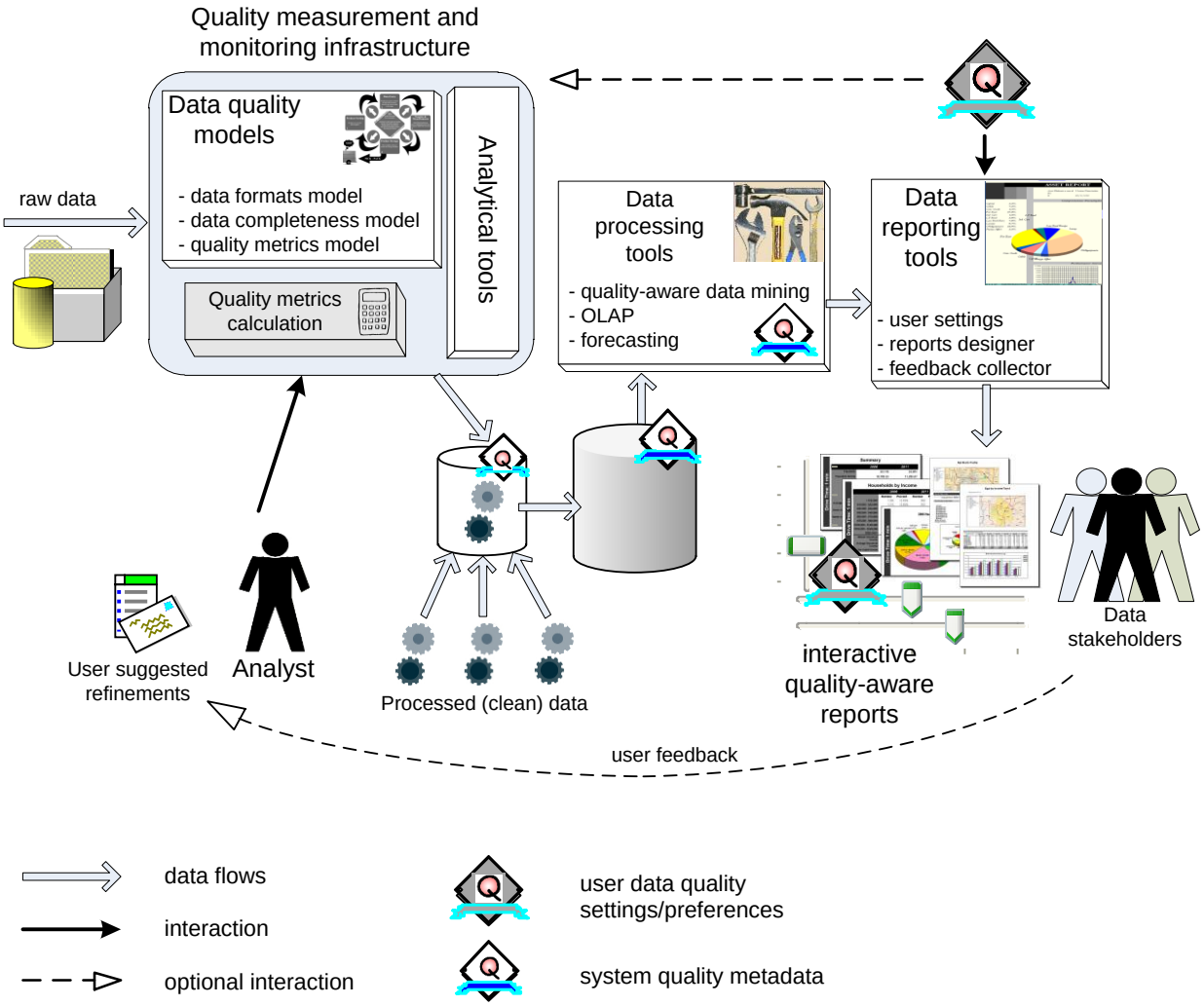


Figure 17. Functional decomposition of quality-aware BI architecture.

Of course, number of quality profiles will probably generate that number of different views on the same base data. The most vital example is different level of trust users may have for the same data source. Conflict resolution of such quality-related issues, as well as accountability of measured data imprecision and uncertainty in classical OLAP models (e.g., [92]) is a separate direction of data quality research along these lines.

5 CONCLUSIONS

In this chapter, we studied a problem of data quality awareness in business intelligence applications of nowadays, taking a healthcare scenario as an instantiated example of one of them. In particular, in such a scenario, we considered the fact that some of data source systems may supply their data having quality issues, to a central data warehouse. For such a real life case, we studied: 1) the effect of data with quality issues on the ultimate data consumers, i.e., an extent of real world facts misrepresentation due to the data issues; 2) interdependency aspects of data quality dimensions in such an application; 3) representation of quality issues aware data at report level. As we have demonstrated, each of these components is vital for a modern BI application.

In our work, we considered only some of key quality dimensions (namely, completeness, consistency and confidence) a BI application design may account. To a higher extent, a particular BI application may take a similar approach to derive certain quality dimensions prone to the application's business use cases, studying similar representational (storage and processing) and presentational (reporting) data quality aspects, as well as interdependency between them.

Chapter 4
Data Quality Measurement for the Entity Name System

One of key areas of interest of researchers in the data quality area is providing approaches for objective (whenever possible) measurement of the most important or the most applicable data quality dimensions in a system context. In this chapter, we will continue previous chapter's user-centric data quality studies with exploration of objective assessment of quality dimensions for a real world system like Entity Name System (ENS), which deals with management of global unique identifiers of entities from the real world, operating with structured or semi-structured descriptive data taken from and designated to users, who are the ultimate quality assessors.

For this system, we will demonstrate how a sample methodology for data quality assessment (AIMQ) drove us to deduction, definition, measurement, aggregation and usage of relevant quality dimensions for data retrieval, processing and dissemination. Before that, we will give an overview of the ENS and describe its typical data lifecycle to identify stages at which potential data quality problems may occur. We will also show how a set of quality dimensions influences quality requirements for input data imposed for users of the ENS.

1 OVERVIEW OF THE ENTITY NAME SYSTEM

Entity Name System, or ENS, is a system that was developed during an FP7 European project – Okkam. One of the key goals of the ENS was implementation of research outcome as for management of global unique identifiers or potentially any entity from the real world. The problem of getting and having such identifiers can be demonstrated by the following example. In the real world, or Web of meanings, one may see real world entities and the corresponding relations between them (Figure 18). In a Web of links (Internet), from the other side, those entities may have their representation, and the mapping may be many-to-many. For a spider or even a user surfing or mining Web of Links for the entities and their relations, it can be often a problem (among others) to identify and disambiguate entities [93].

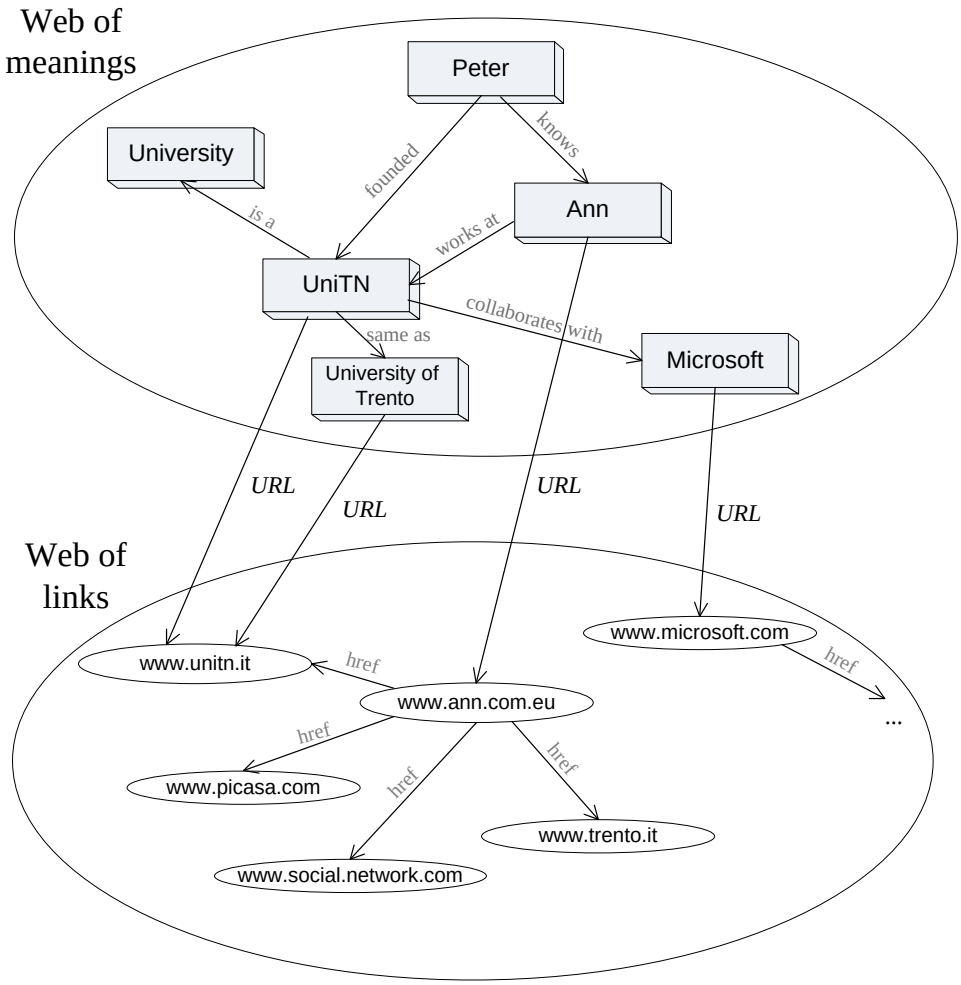


Figure 18. Web of meanings vs. web of links.

Consider, for example, the following real life case. Among the most cited authors in computer science there are Professors Lee and Zhang; though, there are more than a dozen of computer scientists with those two last names and different first names among top 5000 researchers. Now, Let us consider the case in which somebody will search for another person whose name is Lee but he is not a scientist. Automatic disambiguation becomes a first place problem here, and the ENS is aimed to help in this task [94].

For the current work, however, our primary goal in this project was study of data quality aspects in the system context, for which we explored DQ requirements, designed proper dimensions which were partially (for instance, consistency checks) implemented in the system. For these purposes, we relied on the conventional data structures of the ENS, as well as data lifecycle and key system components which we describe below.

The key data abstraction in the ENS is *entity* [95], [96] which is a data object having arbitrary number of attributes which are simply name-value pairs (by default, while we mention attribute, we always consider both its name and value, unless we explicitly indicate one of those). Moreover, each

entity has one of the following types (as it has been decided by the research consortium): person, organization, event, artifact, location or other [97]. In the rest of the chapter, we will refer to *description* of an entity (e.g., a set of attribute-value pairs describing an entity, together with its name and type) simply as *entity* unless we explicitly refer to a real-world entity.

To help user in the definition of a new entity or finding one already in the ENS, there are groups of pre-selected *default (*or *canonical) attributes* which are specific for each type of entity, like 'name/surname' for person, 'location' for event, etc. Those attributes help in the unification of descriptions of various entity types, playing an important role in data quality measurement and maintenance, as we will show in Section 3.

1.1 OVERVIEW OF THE ENS ARCHITECTURE

Since ultimate data product is a result of a process that creates or modifies data, while measuring data quality in the ENS, we have to consider both *processes* and *data*. Figure 19 represents data assets (denoted by clouds) and processes/software modules (rectangles) which create/modify data assets of the ENS. Below, we will consider each of those elements in details.

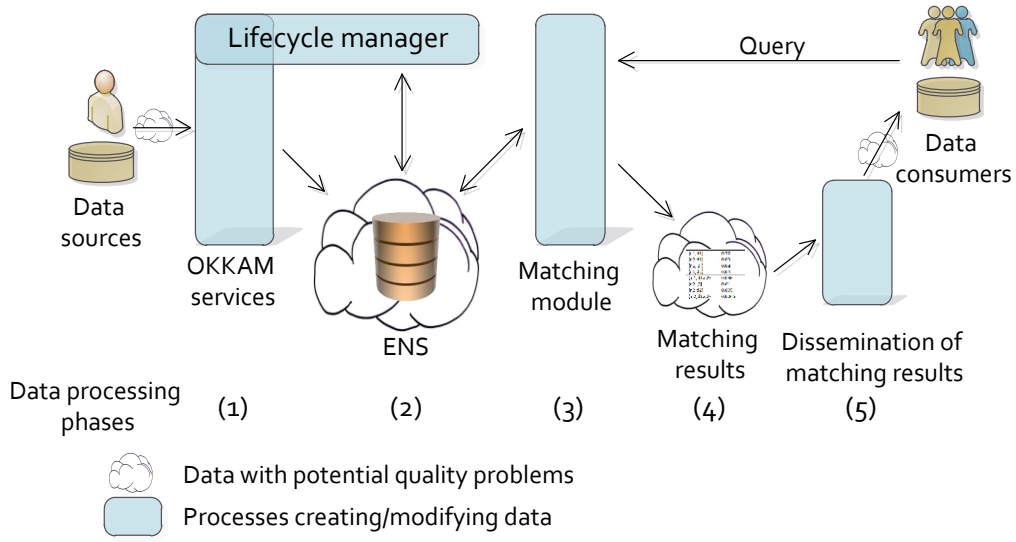


Figure 19. Main processes and the related data assets in the ENS.

Phase of data acquisition (bulk importers, manual entries) and creation of unique IDs take place in a primary data processing stage denoted as (1) in Figure 19. Such a process may include initiatives to suggest or force user (during manual data entries) to follow specific formats that are preferred by the system for data quality consideration. For example, it would be better to use a unique format for date values to avoid confusions in interpretations ("1/5/2010" – is it May 1st or January 5th?).

In some cases, one may provide a complex analysis of order of input fields and their values to decrease probability of data entered by user to be erroneous – this can be done, for example, by using Bayesian network [98]. However, such an approach of dynamic analysis is applicable mostly to standard surveys (can be seen as equivalent of set of attribute-value pairs proposed to user of the ENS) with a set of questions that constitute together with the corresponding answers a basis for learning a Bayesian network. In our work, we do not focus on this aspect of data quality prone to input data, but rather on its more universally defined and controlled quality aspects.

With the help of a rule-based approach to data quality control [99], we focus on *static quality analysis* that catches some quality problems already at the time of data input (phase 1, Figure 19). In particular, by checking data for *consistency*, we ensure that the most common and important consistency threats that must be taken under control in the ENS are mitigated already during data entry or importing process, either in online or offline manner correspondingly.

Continuously operating with input data and data in a repository (phases 1 and 2 in Figure 19), one of pillar software components, called Lifecycle manager, includes some of the ENS services such as data acquisition and offline quality measurement. In particular, Lifecycle manager controls data editing, merging/splitting of relevant/disjoint entities in order for proper representation of entities in the ENS against their real world counterparts. Moreover, it invalidates "stale" or invalid items, querying relevant external sources for updates.

Another pillar component, called Matching module, creates (upon a user request) during phase 3 a data asset (phase 4), which is a result of semantic entity matching process. Later on, the result has to be disseminated and conveyed to user by means of a corresponding set of dissemination processes depicted in phase 5 of Figure 19. The importance of this process should not be underestimated – even if the asset (4) will have data of perfect quality (according to agreed notion of data quality for a system), but the user will not be able to comprehend the results, the entire system will be useless for the user. This means that the quality of data in its common meaning as "fitness for use" will not be satisfied.

In the following sections, we will provide a set of relevant quality dimensions that we selected for measurement and controlling quality of data and ultimately, how the corresponding metrics can be used to achieve a goal of user satisfaction.

2 DATA QUALITY DIMENSIONS IN THE ENS

In our work, we have customized quality dimensions, deduction of which was driven by a methodology for data quality assessment called AIMQ [13]. According to it, we have separated levels of quality issues, which are important to consider in our repository into a) data/system level and b) representational level. At each of them we have identified the most important quality dimensions dealing with the following aspects in our system:

- compliance of data characteristics with relevant standards defined in the ENS (*consistency*);
- validity of data with time passage (*freshness/staleness*);
- relevance of entities to user queries (*popularity*);
- extent of contribution of attributes to help user find relevant entity (*representational importance*);
- ability of an entity to be found upon provided set of keywords (*distinctiveness*).

We believe that exactly these five dimensions best serves our goal to measure quality of data in the ENS as objectively as possible. Among them, the first two dimensions (at data/system level) are used for measurement, assessment, improvement and exposure of data quality in the repository itself, while the last three (at representational level) are intended to define how good data for the designated use is, i.e., for presentation of relevant data to user with respect to user queries and expectations. Note that popularity, together with representational importance and distinctiveness are among quality dimensions that are considered less commonly in the research community, but are very important for the ENS.

Among quality dimensions we could also consider at **data/system level**, are semantic consistency, completeness and accuracy, mentioned by number of researches in the DQ field as some of key quality dimensions [22], [47], [25], [9]. However, we cannot exploit them in our system because of the following reasons. While assessing data along *consistency* dimension based on a set of syntactic consistency rules, we do not enforce any semantic consistency constraints, which may be very hard or impossible to get (e.g., even if all the stored postal codes are up-to-date and valid at every query time, or not). For the same reason we do not focus on *completeness* dimension.

Though in traditional data management approaches *accuracy* is one of key quality dimensions, for our system such a dimension is out of scope of data quality analysis. The reasons for this are the following: 1) getting referenced real-world object or its property is often hard or impossible; 2) description of an entity in different languages (Munich and München which denote the same city and can be also considered as inaccuracy) can be easily handled by the ENS by dereferencing given descriptions of the same entity which is one of the main purposes of the system. Alternatively, in similar data systems for the resolution of inaccuracy of the latter case, dictionary or ontology could be applied [39].

Among the rest of quality dimensions suggested by the AIMQ methodology at **representational level**, not all of those can serve our goal of data delivered to user to be "the expected one", or "fit for use". As we have mentioned before, in this work we focus on objective assessment of data quality, reducing subjectivity as much as possible. In fact, representational level quality dimensions are hard to be objectively measured on their own, but one may get approximations of their objective measures and use them to guide proper data dissemination (phase 5, Figure 19).

According to the AIMQ methodology, apart from the selected quality dimensions, representational level may also include:

- *concise representation* – max/min limits on number of attributes per entity, number of characters in attribute values, etc.;

- *accessibility* – when data is presented, user may want to understand its provenance in order to query it again or use it in another manner; for example, if synthetic (aggregated) entity would be presented to user as an answer to a query, the nature of the entity should be clear – whether one can get an entity presented with a URL, or it is only a virtual copy created under certain conditions;
- *interpretability* – avoidance of abbreviations or ambiguous names (for example, does "Amsterdam" mean only a city in the Netherlands? Or a city at all?), verification of presence of measurement units, where appropriate (size = "5" does not say much about the size unless measurement unit is known);
- *relevance* – correlation of presented data to user query, e.g., if a user queries for a city using its geographical coordinates, they should be shown on top, otherwise, if the coordinates are not used in a query, according to the representational importance metric, they should not be among the top attributes in a query answer.

Having a set of selected dimensions covering key data quality aspects in the ENS, none of the rest of main quality dimensions discussed in the research community is suitable and required for implementation of our vision of data quality management in the ENS.

Figure 20 demonstrates relevancy of each stage of a dataflow in the ENS to the data quality dimensions previously identified.

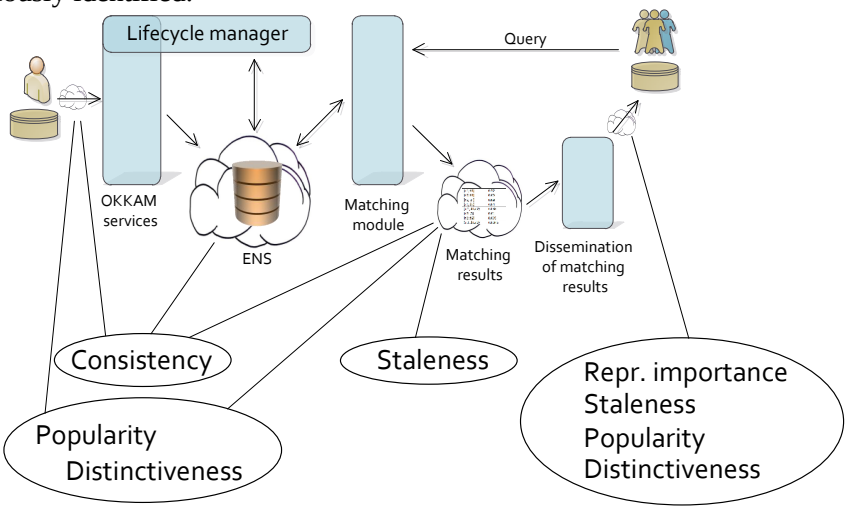


Figure 20. Relevance of data quality dimensions to each stage of a dataflow in the ENS.

In the figure above, one can see that entry point of data flow in our system (when user or an automated importer feed in data) is subject to detecting and monitoring such quality aspects as consistency, popularity and distinctiveness. The same set of aspects, together with a time-relates measure of staleness, must be elaborated at entity matching stage, before user gets query results. At the stage of dissemination of matching results, an additional aspect should be also considered – representational importance. Consistency, from the other hand, is no more an issue at this stage since it is prone to earlier data processing stages where integration and standardization processes take place.

In the following sections, we will give more detailed explanation of each of the selected quality dimension in context of the ENS (Sections 2.1 to 2.5). We will also describe an approach for getting the relevant measures for the selected dimensions followed by a demonstration of their usage in our system (Section 3).

2.1 CONSISTENCY

In its generic sense as a data quality dimension, consistency denotes a degree of correspondence of data to a set of constraints or business rules (see Section 1.1 of 0 for more details). By means of consistency constraints, we define requirements for data to be of as high quality as possible at the earliest possible data processing phase, rather than providing cleaning of inconsistent data. As inference of business rules is not a part of our work, for the ENS we consider the following integrative parts (metrics) of this dimension: 1) uniformity of data formats and 2) conformance to a set of constraints. Later on, we will provide corresponding approaches to measure both of these metrics, but first, consider the following use case of this dimension in the ENS.

For the notion of consistency in the ENS, we want to treat only identical or equivalent values referring to measurement of the same real world item as valid values. It can be a measurement of width of a certain part of a certain item from the real world. Such a measurement, however, may be previously stored by one user in inches, while another user wants to input the same measurement in centimeters. There should not be contradiction as far as those measures are equivalent (record 1 in Figure 21), but as soon as given values for same attributes are inconsistent (record 2) or can be treated as such due to absence of measurement units (record 3), we want to mark those as having consistency issues.

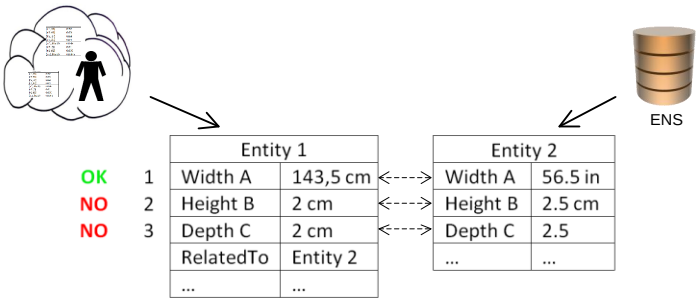


Figure 21. Sample consistency issues in the ENS.

Hence, the first consistency metric mentioned above, ***uniformity of data formats***, can be composed of the following open list of exemplary elements relevant to our system:

a) Dates

As a part of efforts on disambiguation of date values, we require that user consciously provides date values using proper format accepted in a system instead of using one's local format by default. For example, we cannot say anything of the given date "05-03-04" unless we know the format it follows. Another example is "9/11": does it mean September 11 (2001) or November 9 (1989), which are both historical dates known to millions of people? In fact, date format can be deduced or implied from supplementary information, like user location. However, we focus on unification of date values without touching context behind them (which is a separate research problem).

Date format unification is a problem that requires an approach that will be a trade-off between operation with user-friendly formats and machine-friendly ones [100]. While examples of machine-friendly formats are described by ISO 8601:2004, user-friendly formats can be "May 25" (of current year), "2003年4月2日" (in somebody's native language), or using other calendars rather than Gregorian.

If a similar system would have aimed at users in a particular region or country, one would design a solution to disambiguate date formats, making them less machine-friendly in a system and more user-friendly in local community. However, in this case one would also have to consider technical environment like other systems that may interact with the original one.

From the other side, some users may provide date as set of few attributes like year = "2005", month = "03", day = "04" instead of a single value (date="2005-03-04"). Since interpretation of date in the former case is a straightforward task, our system focuses on disambiguation and unification of single valued dates, as we mentioned before.

For these purposes, user interface in the ENS proposes on a web form to follow a certain format like the one defined by ISO 8601:2004, according to which the extended date format is "YYYY-MM-DD" (e.g., 2005-03-04 for March 4-th, 2005). Automated importers are tuned to cast foreign date values to this format as well.

b) Addresses

To ensure lower boundary of quality level of addresses in the ENS, we require presence of at least the following components in corresponding address attributes of an entity: ZIP code, country, city, street, house number. Absence of any of those components will result in notice message visible to user upon manual data entry, marking the fields for potential data quality problem. Due to a free form of entity description, user may ignore the message if he or she considers given information sufficient to provide. Is this case, those addresses will be stored "as is".

c) E-mail addresses

Sample recommendations [101] describe syntactically valid email addresses as those of format *local_part@domain*, where *local_part* and *domain* are up to 64 and 253 characters long correspondingly. Local part may consist of any combination of alphanumeric characters or any of the following special

characters: ! # $ % & ' * + - / = ? ^ _ ` . { | } ~. A period character must not start or end local part; neither can it occur twice in a row. Any other character apart from those above, must be quoted in the local part.

Domain part of email addresses must follow existing requirements for formats and syntax of domain names [102], [103], [104], coupled with currently registered top-level domains [105].

d) Phone numbers

We require that each phone/fax number follow International format according to ITU-T E.123 (02/2001) [106], for example +31 42 123 4567.

The second metric of the consistency dimension we mentioned before is ***conformance to a set of constraints***. For the ENS, we consider the following set of constraints for each attribute/entity that they should comply with:

a) Syntactical validity of names of attributes

Each attribute name must be a valid qualified name (QName), which are valid identifiers for elements and attributes defined by W3C [107]. It has namespace and local part separated by colon sign. In particular, namespace must be a valid URI or null.

b) Limits on length of names of attributes

Each attribute name is intended to be compact and representative. Basic check for size may highlight potentially problematic names. In some cases, names may be represented as references to ontological concepts (e.g., "plant_type"). For those names upper limit on name's size is defined only by a type of variable used to process them. Hence, as a general rule, we require that each attribute name consist of 2 to 70 characters for multi-word names and minimum 2 characters for single-word names, unless we can find name in one of available ontologies.

c) Limits on length of values of attributes

We set the upper limit on number of characters of any canonical attribute value to 70, and to 255 for non-canonical attributes. This is done to force the users to give the intended data instead of arbitrary one. For example, we do not expect that user enters full address as a value of attribute "street address". Instead, we expect that the intended value is of maximum length of 70 characters. In case if the upper limit is not respected, we issue a warning that will indicate a possible quality issue. Such an issue can be then either ignored (if user considers the value overriding the imposed limit as a correct one) or corrected by the user during data entry process.

d) Limits on number of attributes per entity

We advise users to have from 2 to 50 attributes per entity, otherwise a warning message is shown; in the latter case user is still able to produce an entity with 1 or more than 50 attributes.

e) Consistency of measurement units

For those attributes destined for indication of measurements, we require that together with the values, those attributes also contain measurement units. Those units, in their turn, should correspond to a measurement type, which is declared in attribute name. For example, if attribute name is "height", then its value most probably should contain a measurement unit of length.

With such kind of check, we can mitigate "default measurement units" assumed by people: for instance, when human's height is said to be 175, there is only a small chance that somebody using metric system will guess the measurement unit incorrectly. However, it is not always so easy to deduce the correct measurement unit for a machine. As we have mentioned before, in our quality measurement, we do not rely on semantic analysis, and thus, we do not deduce "implied" data, but rather we request user to provide it to lessen subjectivity of quality measurement as much as possible.

f) Detection of occurrence of different values for the same attribute name

In most cases each attribute should describe a different property of an object from the real world. Hence, each attribute name should occur at most once in entity description in the ENS, and we expect that attribute-value relation is normally 1-to-1. However, this is not always the case: for example, one can give for some entity alternative IDs as a set of valid attributes with the same name and different values.

To notify user of potential quality issue, a warning message is shown when same attribute name occurs more than once for the same entity.

g) Deduplication of name-value pair occurring in one entity

Occurrence of the same or equivalent name-value pair in a description of the same entity is superfluous and must be corrected.

h) Check for presence of NULL values

While creating a new attribute, user must also give a proper value (compliant with all the imposed constraints). Creation of attributes with NULL values will be blocked.

Each of the presented consistency requirement is a result of a trade-off between the required control over quality and feasibility of implementation of such a control in our system.

*By the term **consistency requirement** (CR) we call each aspect (items a-d and a-h) which is a part of each metric of consistency we presented above, namely 1) uniformity of data formats and 2) conformance to a set of constraints.*

Therefore, we measure level of consistency of data element by checking its actual compliance with the intended CRs, as we will show in Section 3.1.

2.2 POPULARITY

For popularity measurement, we rely on statistical analysis of selected entities from answers to past user queries. As a result of application of this metric in generating answers to current queries, a list of the most popular items for those queries among past users is produced. Measurement procedure implementing this approach is described in Section 3.2.

2.3 DISTINCTIVENESS

For each new entity user intends to create, we want to understand if candidate entity is new for a repository (i.e., distinctive enough, given a threshold, from existing entities) or it duplicates one of existing entities. For that, we want to calculate metric of *distinctive uncertainty*, which indicates an extent of uncertainty about a supposition that new entity is distinctive enough from any similar one that can be found by matching module of our repository. Note that this metric differs from the one indicating probability of attribute to be used by user to find entity of interest, which is explored in separate studies [97]. In that work, authors provide results of surveys showing which attributes users exploit to describe and query certain entity types (empiric metric of attribute *dominance*), and extent to which an attribute helps to make entity of certain type distinctive from entities of other types (called *conceptual distinctiveness*). Our metric of distinctiveness supplements these two notions obtained from the social studies with data-driven metrics.

To demonstrate rationale for distinctive uncertainty metric, consider the following example. User wants to create entity describing actor Morey Amsterdam. By giving only one attribute "$name_{new}$= Amsterdam" for its description, user underspecifies real world entity she or he wanted to describe. Suppose that after an attempt to create such new entity, matching module will find three other entities with similar entity profiles – all of them have same attribute name-value pairs: "$name_1 = name_2 = name_3$ = Amsterdam". Those entities also have set of additional attributes with identical/equivalent values ($country_1 = country_2$ on Figure 22) or different ones ($country_2 \neq country_3$). Our goal is to disambiguate new entity from any similar one, or otherwise ensure that new one is duplication of one of existing entities and hence, should be merged accordingly.

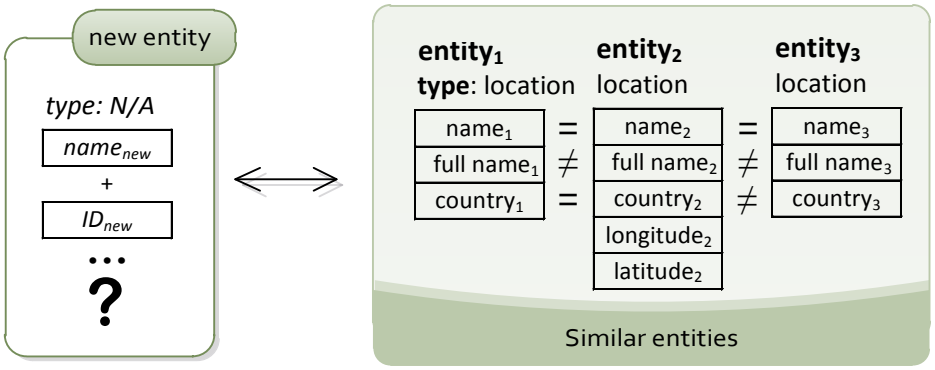


Figure 22. Example of new entity creation: calculation of distinctive uncertainty metric will guide for additional distinctive attributes.

For disambiguation, our user may either implicitly indicate which entity among similar ones he or she wanted to describe, or add some distinctive attributes like *position* or *latitude* (which will enable comparison of new entity with 2nd similar one) or even some *ID* (which may not help in comparison of

new entity with similar ones due to absence if attribute *ID* in any of those). From the other side, such attributes as *full name* and *country* may disambiguate new entity from the three similar ones (in case if distinct values of those attributes are provided for new entity).

Since we do not know which attributes user will prefer to use, the question would be: to what extent each existing and further provided attribute contributes to the goal of distinction of new entity from all similar ones from a repository? Ultimately, with distinctive uncertainty metric we want to answer the following question: *how many attributes should user add in order to disambiguate new entity, given current attributes of new entity and set of similar entities*. We will show how to answer it in Section 3.3.

2.4 REPRESENTATIONAL IMPORTANCE

The measure of *representational importance* is used to show to user only those attributes that he or she can be interested in the most. For example, if user searches for a city, attributes *country* and *population* are much more valuable for that user than such attribute as *geographical_coordinates*. An approach for measurement of this quality dimension we discuss in Section 3.4.1.

2.5 STALENESS

While residing in a data repository, data may naturally become obsolete, i.e., an entity from the real world which is represented by a data element in a repository, may change its representation, which may not be reflected by the corresponding data element.

Evaluation of temporal validity is usually provided by means of such dimensions as timeliness, currency, freshness or similar ones. In spite of the fact that DQ research community has not come to a common definition of any of those terms, there are works that try to systemize notions for time-related quality dimensions. In particular, Bouzeghoub and Peralta [45] propose a structured view on existing definitions of such notions as timeliness and currency.

To measure a time-related quality dimension in the ENS, we aimed to have a notion indicating how much a copy of data element looses its time-related quality measure with passage of time, compared to the state of a corresponding real world data originator. In our work, we have adapted the relevant notions of both freshness and age given by Cho and Garcia-Molina [62]. In their work, they focused on problem of keeping cache of web data as fresh as possible by means of discovery of update rate of web data and defining data query strategy. Following their semantics, our aggregative notion of *staleness* measures a duration of time passed from instant when copy of a data element[3] in a repository was last in a synchronous state with the corresponding "original" element in a source system or real world state (or how much time passed from the time when data entry is changed in a source system without synchronization in ours). Hence, staleness is a measure of non-correspondence of data element to its corresponding object or its properties in a source system.

Our notion of staleness also closely correlates with "currency level" studied in [64] and denoting a portion of data in a database that can be considered up-to-date due to equality of its corresponding copies in other databases since the latest realignment. However, we define and measure staleness at finer granularity level (also showing its aggregation to the coarser ones – see Section 3.5) and under the supposition of a real world-driven periodical updates in source data systems that we try to predict. Thus, we measure the difference between current time t_{cur} and time of expected update in the "real world". Since representation of real world objects is hard or impossible to track all the time, we will consider updates from external sources as those from the real world. Hence, we want to predict t_{pr}^{e}, an instant of update of a data element *e* that should take place in a corresponding external system. Note that for some data elements, like those representing date of birth, we do not need to measure their staleness due to their stability. Those elements are normally never updated (in case they are, one should see an issue of accuracy, validity or similar ones, rather than staleness).

To demonstrate the semantics of notion of data staleness from a system perspective, consider the following example. There is an entity composed of parts from three different sources (S_1, S_2 and S_3), which are regularly sending their planned updates for corresponding parts of the entity in our system. Suppose that only two sources sent their updates in due time, and the third one did not do it due to technical reasons. As soon as instant of due time for update of a S_3 will pass, our entity would be partially stale in our system (dashed timeline in Figure 23), and it will remain stale till expected update will arrive.

[3] Data element is an entity or an attribute value. For the data staleness measurement, granularity of the data element is not important.

If at some point of time that entity will be queried, we want to measure staleness value (of a corresponding part) to convey it to user or use it in a matching and aggregation procedures that may consider most fresh data.

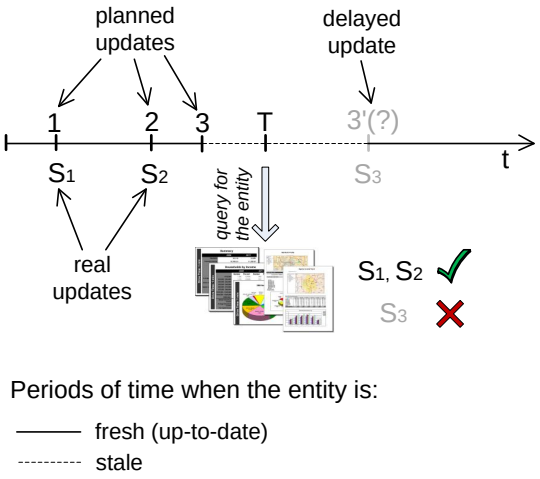


Figure 23. Data staleness issue of entity composed from multiple sources S_1, S_2 and S_3.

A separate question on management of outdated values may arise: what should one do with old values, store or replace them with more current? In fact, if an update rate of an attribute is low (update of a data element on average takes place once a month or more rarely) we can store all old values; otherwise we will replace them with more current ones.

An approach for measurement of staleness may rely on data element's history of updates (timestamps), by means of which we may predict an instant when it should be updated in the nearest future, or it already had to be updated before current time (but we do not see this for the element at hand). In the latter case, we consider data element as stale and calculate its staleness value using a proper (statistical) method for prediction of each next update for each data element that we monitor, as we will show in Section 3.5.1.

3 MEASUREMENT OF DATA QUALITY IN THE ENS

In Sections 3.1 to 3.5 we propose an approach to objectively measure (or to approach to such measures) quality dimensions that are most important in the ENS. We will also demonstrate how we aggregate them at different granularity levels. For aggregation of values measured for each quality dimension, we consider the following granularity levels: attribute, entity and entire repository (ENS). At attribute level, we operate with data which is either attribute name, or its value, or both. At levels of entity and repository, quality dimension aggregates level-specific aspects and also relevant metric of lower level(s). An important part of how one can use measures of each dimension to control data quality is followed in each section.

A demonstration of usage of quality measures at different granularity levels one can see in Figure 24, where input for quality analyzer can be entire entities, as well as their parts – attributes. Quality of input data is measured along each quality dimension using corresponding vocabularies, quality policies, etc., while deduplication is provided with help of a matching module. As a result, quality analyzer gives quality assessment of attributes (e.g., attribute is stale, incomplete, etc.), which is further aggregated to quality measures of entire entities (staleness of entity is 5%, completeness is 90%, etc.).

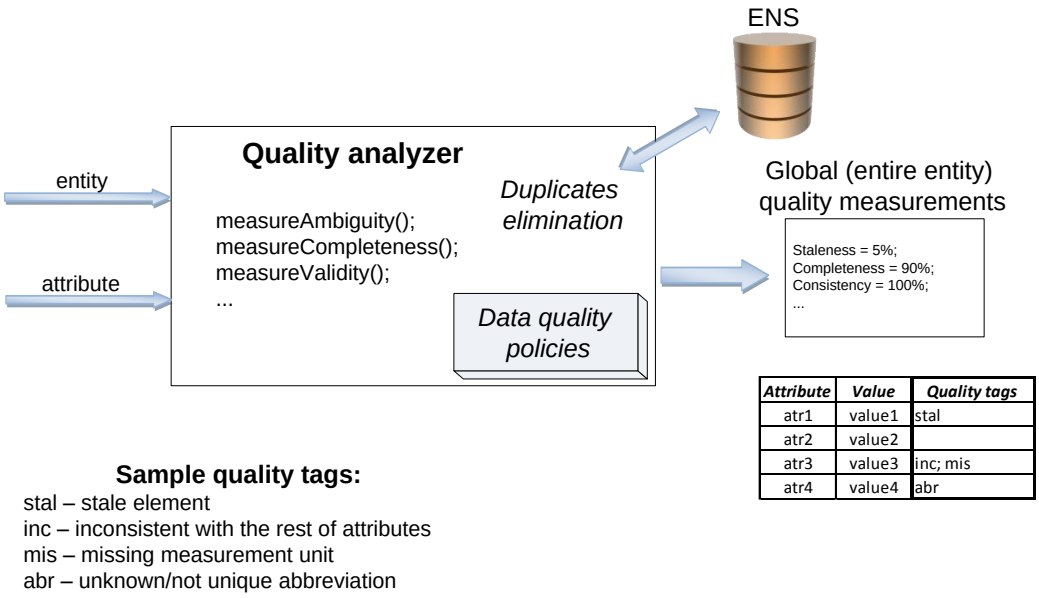


Figure 24. Attribute-level and entity-level quality measurement.

To measure data quality in the ENS, we introduced number of custom notions for data elements of attribute, entity and repository levels – they are summarized in Table 3 for further convenience.

At each data element level (attribute, entity or entire repository) we will provide interpretation signs of corresponding quality metrics with the following semantics: +[-] Q↑ signifies higher/[lower] value of a metric means higher quality of corresponding data element along its quality dimension.

Table 3. Base notions for measurement of data quality in the ENS.

cr, CR	consistency requirement
cr_a	certain consistency requirement for attribute a
a_{CR}	CR-dependent attribute (having at least one cr_a)
a_F	CR-free attribute
CR_a	set of CR's applicable to attribute a_{CR}
A^e_{CR}	set of all CR-dependent attributes of entity e
A^e_F	set of CR-free attributes of entity e
a_{def}	default attribute
a_{ndef}	non-default attribute
a_n, a_v	attribute name and value
$a = (a_{CR} \vee a_F) \wedge (a_{ndef} \vee a_{def}) \wedge a_n \wedge a_v$	attribute which is a name-value pair, whether CR-dependent or not and whether default or not
$A^e_{def} = \{a_{def1}, a_{def2}, \dots, a_{defu}\}$	set of default attributes of entity e
A^E_{CR}	set of all CR-dependent attributes in a repository
A^E_F	set of all CR-free attributes in a repository
$A^E = A^E_{CR} \cup A^E_F$	set of all attributes in a repository
$e = A^e = A^e_{CR} \cup A^e_F$	entity, set of all attributes composing it
$e_{CR} \mid \exists a_{CR} \in A^e_{CR}$	entity with at least one CR-dependent attribute
$e_F = A^e_F$	entity with only CR-free attributes
$E = \{e_1, e_2, \dots, e_m\}$	set of all entities which compose entire repository
$E_F = \{e_{F1}, e_{F2}, \dots, e_{Fh}\}$	set of entities with attributes that are only CR-free

3.1 CONSISTENCY

3.1.1 MEASUREMENT OF CONSISTENCY

Following the definition of data consistency from Section 2.1, we now present an approach to calculate the corresponding metrics.

As we mentioned, a key element of consistency analysis in our system is consistency requirement (CR). Consistency of an attribute is a measure of its compliance with agreed set of CRs. For a format uniformity analysis, intended CRs of an attribute are detected by checking name of an attribute (whether it contains "date", "address", "e-mail/email" or "phone / cell / mobile / fax / tel / telephone" keywords). A corresponding value is then verified with a template of detected type of attribute, and possible corrections are attempted to be enforced at entity creation time.

As one can see from description of each consistency requirement (Section 2.1), they are designated for use at attribute level. While aggregating the measures (upon formats uniformity analysis) to entity/repository level, we want to calculate an extent of consistency of fraction of attributes or entities that should be checked for consistency issues in an entire set of attributes composing that entity or respectively, in an entire set of entities of the repository.

a) Attribute level (+ Q↑)

By definition of our consistency measure, it can be calculated only for attributes that are related to one of CRs specified before. Hence, we suppose that consistency of CR-free attributes (i.e., in our case those attributes not storing date, address etc., as we mentioned before) is $Cons(a_F) = 1$, that is they are completely consistent. Consistency of a CR-dependent attribute a_{cr} according to consistency requirement CR can be either 1, if that attribute satisfies it, or 0 otherwise:

$$Cons(cr_a) = \begin{cases} 1, & CR\ is\ satisfied\ by\ attribute\ a_{CR} \\ 0, & otherwise \end{cases} \quad (4.1)$$

And the overall measure of attribute consistency we can calculate as:

$$Cons(a_{CR}) = \frac{\sum_i Cons(cr_a)}{|CR_a|} \quad (4.2)$$

where $\sum_i Cons(cr_a)$ is total number of CRs satisfied by attribute a_{CR}; $|CR_a|$ is number of CRs applicable to attribute a_{CR}.

Thus, $Cons(a_{CR})$ indicates a degree of compliance of attribute a_{CR} with its intended (that must be checked) CRs. It ranges from 0 (violation of all intended CRs) to 1 (compliance with all intended CRs).

b) Entity level (+ Q↑)

A degree of consistency of only CR-dependent attributes of entity *e* with regards to a set of relevant CRs, can be expressed as follows:

$$Cons(A^e_{CR}) = \frac{\sum_k Cons(a_{CR})}{|A^e_{CR}|} \quad (4.3)$$

where *k* is index over attributes to be checked for consistency issues (CRs), $k = 0,1,2, \dots, |A^e_{CR}|$.

Consistency of entire entity *e* can be calculated as follows:

$$Cons(e) = \frac{|A^e_F| + \sum_k Cons(a_{CR})}{|A^e|} \quad (4.4)$$

Therefore, $Cons(e)$ indicates a degree of compliance of CR-dependent attributes from subset A^e_{CR} with their intended CRs, considering total number of entity attributes $|A^e|$. Since $0 \le Cons(a_{CR}) \le 1$, we have $\sum_k Cons(a_{CR}) \le |A^e_{CR}|$, meaning that entity-level consistency metrics has the following range: $0 \le Cons(e) \le 1$, where 0 means violation of all intended CRs by $\forall a_k \in A^e_{CR}$ with no CR-free attribute, and equality to 1 means compliance with all of intended CRs by all CR-dependent attributes of an entity.

c) Repository level (+ Q↑)

By analogy with the metric at entity level, consistency at this level can be calculated as follows:

$$Cons(E) = \frac{|E_F| + \sum_l Cons(e)}{|E|} \quad (4.5)$$

where *l* is index over entities that have a_{CR} attributes that must be checked for consistency issues; $l = 0,1,2, \dots, |E| - |E_F|$. Like at entity level, $Cons(E)$ ranges from 0 to 1 with the same semantics.

3.1.2 USAGE OF CONSISTENCY MEASURE

By definition of consistency for the ENS (Section 2.1), this dimension has precise measure indicating extent of compliance of data element with intended requirements. In particular, we operate with requirement of two groups: a) uniformity of data formats and b) conformance to set of constraints. The requirements of these groups ensure elimination of syntactic differences and compliance with a set of defined quality rules.

One of key use cases for this quality dimension in the ENS is finding relevant data in repository. For example, inconsistency (in form of variation) in date formats may lead to missing relevant results matching user query.

3.2 POPULARITY

3.2.1 MEASUREMENT OF POPULARITY

Entity popularity (which we will also call *entity rank*) is a measure indicating how many times an entity was selected by users among given results for certain query. We denote it by r_e^{qp} which means number of picks of entity *e* by users from lists of candidate entities given for query q_p. Before we give an exemplifying scenario and formulas to measure popularity of entities, we first present a conceptual schema of popularity ranking in the ENS (Figure 25).

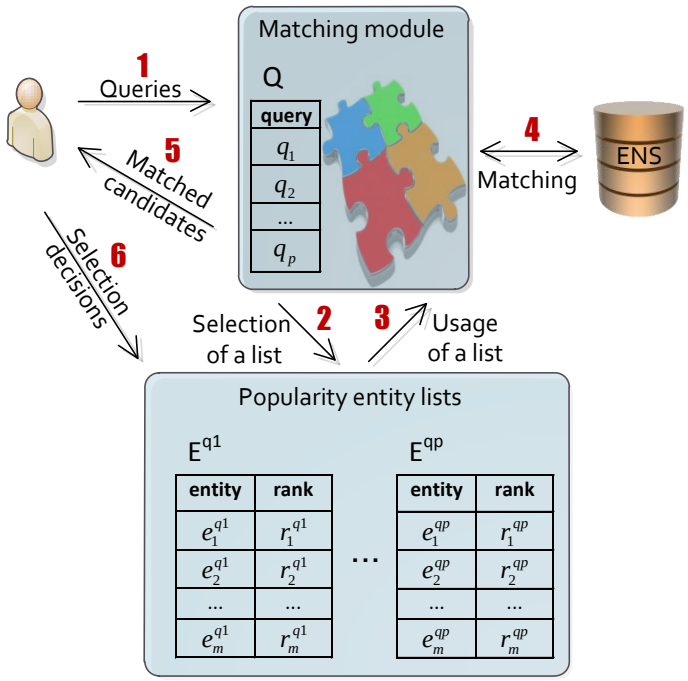


Figure 25. Conceptual model of popularity ranking in the ENS.

In step 1 each unique query of all users are stored in query list $Q = \{q_1, q_2, \dots, q_p\}$. For each query $q \in Q$ we have a corresponding popularity entity list E^q which indicates ranks of entities for query q. Each such list may have at most *m* entities.

As soon as user selects entity from a set of candidate entities given (by matching module of the ENS) for user query q_i, the corresponding popularity entity list E^{qi} eventually gets increased rank for entity selected (step 6). As a result, entities that are selected by users more often for certain query, get higher rank which means superior place in candidate sets presented later to other users. In step 2, matching module checks for existence of list E^q to answer user query (step 3) or use it as an auxiliary mean to find entities from a repository relevant to user query (step 4).

Since popularity list should be built for each unique query, in reality, due to limited computational resources, we can track only top-k queries used most often by users. For the rest queries, matching module exploits its entity matching algorithms with no popularity measure provided by users.

Now, consider a real life scenario demonstrating usage of entity popularity (Figure 26), where user wants to find an entity describing city of Paris by giving query "Paris". This query is checked for

presence in list of queries Q' previously used by other users. If query "Paris" is present in Q', there is also a corresponding ranked list of entities that were previously selected by users for this query.

Since such list is based solely on ranks of entities, matching module uses it in advisory manner. This means that it may also use diverse metadata such as user location, sessions, user preferences and other quality dimensions in order to finalize list of candidates for certain query of each user. For example, if entity *e22* is relevant to city of Paris (and it is the most popular for query "Paris" among most users) but another entity *e11* describes local night club named Paris and located in city of Amsterdam where user issued a query lives, for that user *e11* should have higher rank than *e22*. However, aiming at getting objective and universal quality measures, in this work we focus on measurement of data-driven popularity metric that does not depend on those additional factors.

Hence, having generated a list of candidate matches (e11, e22, e33, ...), matching module outputs result to user who may select entity that s/he considers as proper representation of the real-world instance meant by query. Selection of a corresponding entity description (e22) adds a ranking point to selected entity in proper popularity entity list.

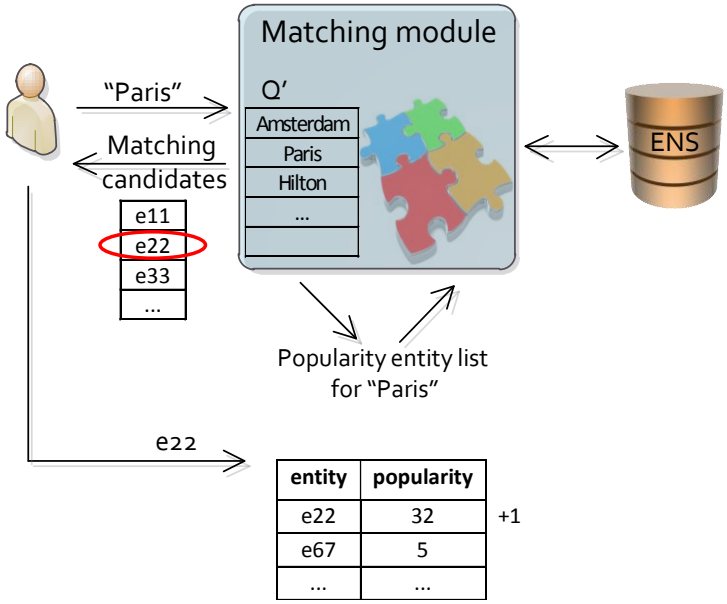


Figure 26. Query processing using popularity lists.

Below we propose methods to calculate popularity measure at different granularity levels.

a) Attribute level

By design of the ENS, popularity of attribute in the defined context cannot be measured on its own. Approximation of a corresponding metric from entity level does not make sense for quality analysis.

b) Entity level (+ Q↑)

Relatively to each single query *q*, we increment popularity of entity *e* if user selects that entity in candidate answer list generated in response to query *q*: $Pop(e)_q = Pop(e)_q + 1$. Overall popularity of entity *e* is averaged over all popularity measurement enabled queries Q having *e* as candidate answer:

$$Pop(e)_Q = \overline{Pop(e)} = \frac{\sum_p r_e^{qp}}{p - t} \tag{4.6}$$

where r_e^{qp} is rank (popularity) of entity *e* for query q_p; *p* is total number of unique popularity measurement enabled queries; *t* is number of unique measurement enabled queries for which popularity of entity *e* has not been measured and hence, $r_e^q = 0$.

c) Repository level (+ Q↑)

Weighted popularity of query *q* at repository level is a sum of popularities of all entities *0,…,m* relevant to this query divided by total number of unique popularity measurement enabled queries *p*:

$$Pop(E)_q = \frac{\sum_m Pop(e)_q}{p} \tag{4.7}$$

3.2.2 USAGE OF POPULARITY MEASURE

Entity popularity metric helps us to make matching results better fit to those probably expected by users. This is done by means of statistical model showing correlation between user queries and corresponding entities selected: the more users select a particular entity for certain query, the more probability that entity should be of interest for other users.

Weighted repository popularity metric for certain query, as well as for entire set Q indicates how popular is that query or a set in a particular repository, given equality of numbers of unique popularity measurement enabled queries among measurable repositories (p). Note that comparison of two or more repositories along only weighted popularity of repository over entire set Q, in some cases may not allow one to determine repository having higher quality data. For example, consider two repositories, E_1 and E_2 such that $E_1 \subseteq E_2$. Suppose that all of our users are interested only in a particular set of entities $\{e_1, e_2, e_3\} \subseteq E_1$; moreover, any entity outside of this set is not interesting for our users and thus can be considered as trash data in both repositories. Since $|E_2| \geq |E_1|$, users are likely to make more selections from matching candidates from repository E_2 than from E_1, which in fact would be attempts to find one of their entity of interest among numerous entities proposed. Hence, according to our model of popularity measurement, we have $Pop(E_2) \geq Pop(E_1)$. In fact, comparing the two repositories and their users, it is obviously that E_1 contains data of higher quality, since it contains less "trash" entities and portion of "useful" entities is higher. Due to the latter fact, users will find entities of their interest faster and with less effort, which make E_1 better "fitted for use" than E_2. Hence, in some cases consideration of other relevant quality dimensions to determine higher quality repository may be essential.

3.3 DISTINCTIVENESS

To have our data quality metric normalized, we want attribute distinctive uncertainty to range from 0 which corresponds to sufficient (see below) specification for entity to be distinctive in a repository, to 1 meaning nondeterministic specification which does not allow one to distinguish entity of that attribute from other entities in a repository. Hence, if an attribute (name-value pair) would be unique in a repository, it will have the least uncertainty for its entity distinction, and thus, it may be enough to have only one of such attributes for the distinction purposes (note, that this goal goes against entity descriptiveness, for which having one attribute is indication of bad quality entity; however, as mentioned before, with quality dimension of distinctiveness we want to solve a separate quality problem). From the other side, attributes having some repetitive equivalents in a repository will be required to have more attributes while their distinctive uncertainty metric increases. A hyperbolic threshold function like $f(x) = \frac{1}{1-x}$ in Figure 27 accounts semantics and interval range of our metric. It represents possible dependency of required number of distinctive attributes $|a|$ on their distinctive uncertainty metric $D(a)$.

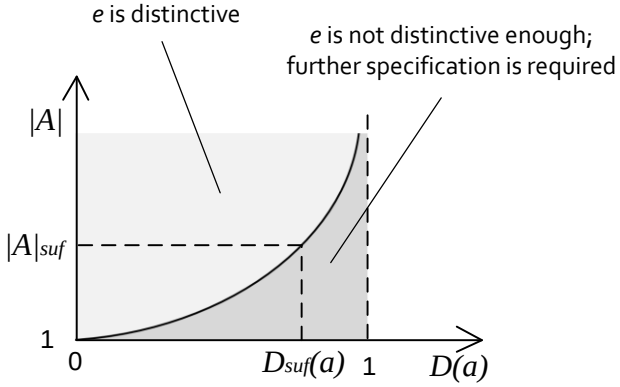


Figure 27. Sample distinctiveness threshold function for new entity creation.

In practice, asymptotical convergence of such threshold function to attribute full non-distinctiveness $D(a) = 1$ can be limited with *sufficient* level of uncertainty $D_{suf}(a)$ to make new entity e_{new} distinctive enough in a repository. For new entity with no attributes initially provided, $D_{suf}(a)$ defines lower limit on number of attributes having in worst case distinctive uncertainty $D(a) > D_{suf}(a)$ as following: $|A'|_{suf} = \frac{1}{1-D_{suf}(a)}$. For example, by setting $D_{suf}(a) = 0.9$, we require that in order to be distinctive enough, e_{new} should have at least 10 attributes. In the following subchapters we will show

how to measure and use metric of distinctive uncertainty for candidate entities already having some attributes (the main use case for this metric) for entity similarity matching step.

3.3.1 MEASUREMENT OF DISTINCTIVE UNCERTAINTY

Since attributes with all name-value pairs that are equal among new entity e_{new} and set of similar entities E_{sim} (identified by the matching module of the ENS) cannot contribute to new entity distinction, we take them out of consideration during calculation of distinctive uncertainty metric both at attribute and entity level, setting the corresponding metrics of uncertain distinctiveness to 1.

a) Attribute level (+ Q↑)

Distinctive uncertainty of name part of attribute $D(a)$ can be measured as a degree of its usage in profiles of entities from set E_{sim} which are similar to new entity e_{new}: the more attribute is used by similar entities, the more distinctive it can potentially be for new entity (for example, attribute "country" in Figure 22):

$$D(a) = \begin{cases} 1, & a_n,\ a_v\ pair\ is\ same\ for\ \forall e \epsilon E_{sim}\ and\ e_{new} \\ \dfrac{|E_{sim}| - |E^a_{sim}|}{|E_{sim}|}, & otherwise \end{cases} \tag{4.8}$$

where $|E_{sim}|$ is number of entities that are similar to e_{new}; $|E^a_{sim}|$ is number of entities from E_{sim} having name part of attribute *a* in their description.

In contrast, attributes that are never used by any similar entity (like "ID" in Figure 22) may not make new one distinctive enough (due to inability to compare entities by those attributes) unless their number is sufficient to meet entity distinctiveness threshold $D_{suf}(a)$.

b) Entity level (+ Q↑)

Since high attribute uncertainty $D(a)$ means high uncertainty in identification of its entity, in our system we require for attributes $\{a_1, \dots, a_k\} \in e_{new}$ to satisfy the following condition: $\forall a_i \in \{a_1, \dots, a_k\}$, $D(a_i) < D_{suf}(a)$. In this way, we treat separately distinctive attributes, those with high distinctive uncertainty ($D(a) \geq D_{suf}(a)$) and attributes specified only for new entity and not for any other from E_{sim} (we set $D(a) = 1$ for them, as for attribute "ID" in Figure 22).

For attributes $\{a_1, \dots, a_k\} \in e_{new}$ (excluding repetitive name-value pairs common for all entities of E_{sim} and e_{new}) we define entity distinctive uncertainty (without consideration of entity category) as their combination:

$$D'(e) = \frac{D(a_1) \cdot |A_1| + D(a_2) \cdot |A_2| + \dots + D(a_k) \cdot |A_k|}{|e_{new}|} = \frac{\sum_k D(a_k) \cdot |A_k|}{|e_{new}|} \tag{4.9}$$

where $|A_i|$ is number of attributes for which distinctive uncertainty is $D(a_i)$.

Apart from set of attributes, distinctive uncertainty of entity also depends on specification of its category ("location", "person", "artifact", etc.). We define metric of categorical uncertainty as follows:

$$D(e_{cat}) = \begin{cases} 1 + c, & category\ of\ e_{new}\ is\ N/A \\ \dfrac{|E_{cat=e_{new}}|}{|E_{sim}|} + c, & otherwise \end{cases} \tag{4.10}$$

where $|E_{cat=e_{new}}|$ is number of entities in E_{sim} with same category as e_{new}; $|E_{sim}|$ is total number of entities that are similar to new entity e_{new}; *c* is a categorical distinctive uncertainty corrective factor which reflects possible overlapping in semantics of categories or types of entities (e.g., some entities may fall into both "location" and "artifact" categories). In our system, we set an application-dependent parameter *c=0.1*. In case if categories for entities are fully disjoint and none of entities can fall into more than one category, one can set *c* to 0.

Categorical distinctive uncertainty $D(e_{cat})$ ranges from 0 indicating that new entity introduces new category with respect to a set of similar entities, to 1 meaning either equality of category of new entity with category of all entities from a set of similar ones, or absence of category of new entity.

Now, we define category-aware entity distinctive uncertainty as follows:

$$D(e) = \frac{1}{2}\ D(e_{cat}) + \frac{1}{2} D'(e) = \frac{1}{2}\ D(e_{cat}) + \frac{1}{2} \frac{\sum_k D(a_k) \cdot |A_k|}{|e_{new}|} \tag{4.11}$$

Having entity distinctive uncertainty with provided attributes, we want to calculate how many additional attributes we will need to make entity distinctive, i.e., reduce the uncertainty. Hence, we want to solve the following equation:

$$D(e) - \left(1 - D_{suf}(a)\right) \cdot |A|_{suf} = 0 \tag{4.12}$$

which gives us solution to the problem of finding sufficient number of distinctive attributes that should make a candidate entity provided by user distinctive enough among similar entities:

$$|A|_{suf} = \frac{D(e)}{1 - D_{suf}(a)} \tag{4.13}$$

where $|A|_{suf}$ should be rounded up to the nearest integer value.

c) Repository level

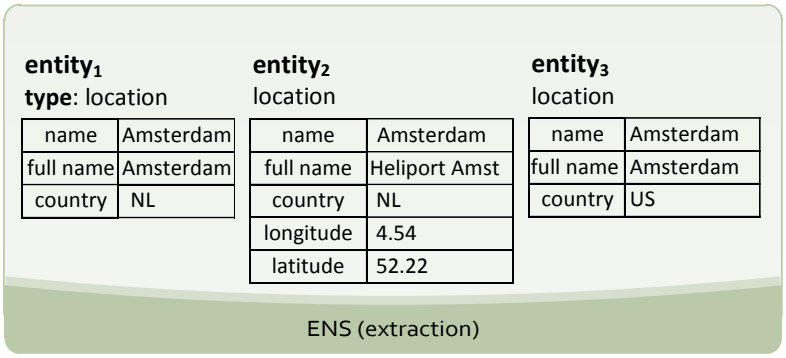
Measurement or aggregation of distinctive uncertainty at repository level does not make sense due to the notion of this metric defined for entity to be created and not for existing entities.

3.3.2 USAGE OF A DISTINCTIVENESS MEASURE

Distinctive uncertainty is mainly used for creation of new entity. With this metric, we want to avoid creation of duplicated entities by means of suggestion to user number of distinctive attributes for candidate entity. To demonstrate calculation and usage of such metric, we take our example with entity description of actor Morey Amsterdam (Figure 22).

Suppose, user intends to create new entity e_{new} by specifying set of corresponding attribute(s). After the end of initial creation process, matching module of our system runs entity matching procedures to find E_{sim}, a set of entities in a repository which are most similar to e_{new}. Calculation of distinctive uncertainty of attributes of e_{new} and entity itself will then take place. Based on analysis of frequency of occurrences of distinct attributes in E_{sim}, suggestions of attributes that user can use for new entity are given. In fact, using the most frequently used attribute(s), user will either prove or disprove similarity of new entity with others. In our example (Figure 28), attributes most frequently and commonly used by similar entities are {*name, full_name, country*}. After elimination from consideration of first attribute (*name*), maximal distinctiveness may have the other two common attributes. Nevertheless, user is not obligated to use only those attributes already used for specification of other (similar) entities. The ultimate goal here is to provide such attributes that will allow one to make new entity distinctive enough with regards to a threshold $D_{suf}(a)$.

Note that such attributes as “description”, “short description”, etc. cannot be considered as those making entity distinctive enough even in case of their inequality with the corresponding attributes of similar entities. The reason is that the same entity can be narratively described in different ways, which is not always possible to notice by machine even if semantic analysis of a description is provided. To get objective measures, in our work we rely on strict equality of attributes and their values while comparing new entity with similar ones.



E_{sim}={entity1, entity2, entity3}

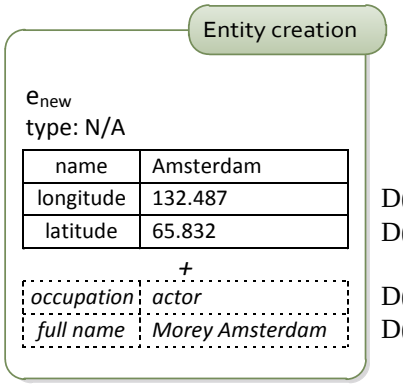


Figure 28. Example of new entity creation with evaluation of its distinctive uncertainty $D(e)$ (Entity creation: solid cells are given by user; some of possible attributes are dashed cells).

Setting $D_{suf}(a) = 0.9$, we can find necessary number of distinctive attributes as following:

$$|A|_{suf} = \frac{D(e)}{1 - D_{suf}(a)} = \frac{\frac{1}{2}\left(D(e_{cat}) + \frac{\sum_k D(a_k) \cdot |A_k|}{|e_{new}|}\right)}{1 - D_{suf}(a)} = \frac{\frac{1}{2}\left(1.1 + \frac{\frac{2}{3} \cdot 2}{2}\right)}{0.1} = 8\frac{5}{6}$$

Hence, after initial specification of three attributes {*name*, *longitude*, *latitude*}, among which one is identical to attributes of similar entities (and hence, is not distinctive), user will need to provide at least nine additional attributes to make new entity distinctive enough, assuming any distinctiveness those attributes may have (consider attributes *occupation* or *full_name* from our example in Figure 28). This suggestion will be given to user in order to support entity creation/disambiguation process.

3.4 REPRESENTATIONAL IMPORTANCE

3.4.1 MEASUREMENT OF REPRESENTATIONAL IMPORTANCE

a) Attribute level (+ Q↑)

Representational importance at attribute level is a relative metric indicating extent to which user will be interested to see that attribute compared to other attributes of an entity. Thus, this individual attribute metric always depends on other attributes of their entity.

For system entity types (person, organization, event, artifact and location) we manually define representational importance for each default attribute:

$$\begin{cases} RI(a_{def}) = c \\ 0 \le c \le 1 \\ RI(A^e_{def}) \le 1 \end{cases} \quad (4.14)$$

where a_{def} is default attribute; A^e_{def} is a set of default attributes of entity e, $A^e_{def} = \{a_{def1}, a_{def2}, \dots, a_{defu}\}$; c is a constant that is assigned manually for each default attribute.

Representational importance of set of default attributes A^e_{def} is sum of representational importance metrics of each attribute $a_{def} \in A^e_{def}$:

$$RI\left(A^e_{def}\right) = \sum_u RI(a_{def}) \tag{4.15}$$

The constraint that we impose on set of default attributes ($RI\left(A^e_{def}\right) \leq 1$) allows us to keep this metric normalized.

For non-default attributes, this metric can be calculated taking into account such factors as: distinctive uncertainty, relevance of attributes used most often to search corresponding entities and place of attributes in entity profile. Study of these factors is outside of scope of current work since it involves feedback from users (studied for our environment in [97]) while we explore data-driven metrics. Hence, to simplify studies of this metric, we assign $RI(a) = 0$ for each non-default attribute.

b) Entity level (+ Q↑)

Since we have $RI(a) = 0$ for each non-default attribute, representational importance of entity is the same as the one of set of default attributes of that entity with the same constraints for normalization of this metric that we imposed on it at attribute level:

$$\begin{cases} RI(e) = \sum_u RI\left(a_{def}\right) \\ RI(e) \leq 1 \end{cases} \tag{4.16}$$

Alternatively, representational importance can be defined and measured by means of a combined quality dimension of both popularity and importance. This consideration is driven by the work of [13] which demonstrated interconnectivity of data quality dimensions. However, as we noted before, in our work we focus on definition and formalization of individual quality dimensions applicable in a real world system we worked with.

c) Repository level

Measurement of representational importance at repository level does not make sense since this quality metric aims at relative comparison and selection of attributes and entities and not repositories.

3.4.2 USAGE OF REPRESENTATIONAL IMPORTANCE MEASURE

When user queries a repository, a list of relevant entities can be presented in various ways, among which some are more efficient than others. The efficiency is measured in terms of the main notion of data quality – to present the data that fits for use. We aim at presenting only that data that will help user to find searched entity if it will exist in our repository. From the other hand, we filter out redundant data which does not help to find entity relevant for user query.

A number of attributes to be shown is defined by a threshold that may vary for different entity types. For example, for entities of type "Location" user may be interested the most in looking at attributes describing country, city and probably address; for entities of type "Event" apart from those three attributes of "Location" type, user may be interested also in start and end date of an event.

Note that limitation of entity presentation influences only the way how user will see entity and which of its attributes will appear in user interface (Figure 29); it does not deal with internal entity representation in a repository (Figure 30).

Let us consider an example when user queries for "Amsterdam" to find actor Morey Amsterdam.

Figure 29. Example of search result for query "Amsterdam" in the ENS (short representation of actual entity searched by user is highlighted)

As one can see in the figure above, for each entity in the list of candidate answers for query "Amsterdam", representational importance metric was used to expose only the most important attributes of each entity. In contrast with representation of full profile of selected entity ("Morey Amsterdam", Figure 30), in the list of results only last, first and alternative names are displayed. Such a presentation eases finding entity of interest in a presented list.

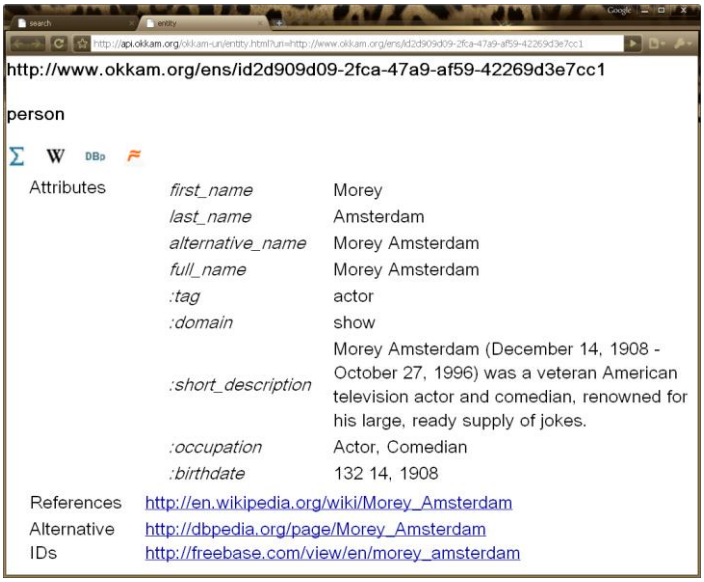


Figure 30. Extended (full) representation of entity "Morey Amsterdam" in the ENS.

3.5 STALENESS

3.5.1 MEASUREMENT OF STALENESS

While measuring data staleness according to the definition of this notion (*a measure of non-synchronization between element's original copy stored in a corresponding data source and its copy in our system*) given in Section 2.5, we should note that since in our work this measure is based on probabilistic methods of updates prediction, we can get only estimated value which reflects the real data staleness as good as implemented prediction algorithm allows it. However, for a data element with aperiodic updates it is hard or impossible to accurately measure data staleness through prediction of updates. Moreover, both computational resources required for the prediction and its accuracy have to be carefully evaluated to be adequate in a particular data repository (a theoretical foundation and details on data staleness measurement algorithms with experimental results see in [108] and in Chapter 5).

a) Attribute level (– Q↑)

As far as attribute is synchronized with its copy in source system, its staleness is considered to be zero (perfect freshness); otherwise staleness is computed as difference between current instant and instant when attribute became non-synchronized (compared to predicted update instant of data element t_{pr}^{a}):

$$S_{t_{cur}}^{a} = \begin{cases} 0, & t_{cur} \leq t_{pr}^{a} \\ t_{cur} - t_{pr}^{a}, & t_{cur} > t_{pr}^{a} \end{cases} \quad (4.17)$$

where t_{cur} is an instant when measurement takes place; t_{pr}^{a} is an instant when, according to a prediction by our system, attribute *a* should be updated.

Since our staleness measurement is based on probabilistic model of updates, this measure is estimation and cannot be considered as a precise one for this data quality dimension.

b) Entity level (– Q↑)

At entity level, staleness measures of attributes are aggregated with corrections for their representational importance (see Section 3.4):

$$S_{t_{cur}}^{e} = \sum_{j} S_{t_{cur}}^{a_j} * RImp(a) \quad (4.18)$$

where *j* is an index over attributes of an entity *e*, $\{a_1, a_2, \dots, a_j,\} \in e$.

Note that this aggregative absolute measure may not be always useful for user or in automated data manipulation operations. Often, it may be more informative to have an additional relative measure of entity staleness which will indicate number of attributes that are stale among all attributes of an entity:

$$\hat{S}_{t_{cur}}^{e} = \frac{|a_j|}{|e|} * 100\% \quad (4.19)$$

where $|a_j|$ is number of stale attributes (such that $S_{t_{cur}}^{a_j} > 0$) of entity *e* and $|e|$ is total number of attributes of entity *e*.

c) Repository level (– Q↑)

At this level, absolute staleness measures of entities are aggregated with corrections for their popularity (see Section 3.2):

$$S_{t_{cur}}^{E} = \sum_{m} S_{t_{cur}}^{e_m} * \overline{Pop(e_m)} \quad (4.20)$$

where *m* is an index over entities of the ENS, $\{e_1, e_2, \dots, e_m,\} \in E$; $\overline{Pop(e)}$ is averaged popularity of an entity *e* over a set of queries for which its rank is more than zero (Section 2.2): $\overline{Pop(e)} = \frac{\sum_p r_e^{qp}}{p-t}$

As with staleness at entity level, this measure at level of entire repository may unveil hidden data processing issues if relative aggregated measure is calculated as simply number of entities having at least one attribute stale:

$$\hat{S}_{t_{cur}}^{E} = \frac{|e_j|}{|E|} * 100\% \quad (4.21)$$

where $|e_j|$ is number of stale entities ($\hat{S}_{t_{cur}}^{e_j} > 0$), $|E|$ is total number of entities in a repository.

3.5.2 USAGE OF STALENESS MEASURE

As we have already mentioned in Section 2.4, one of the goals of staleness measurement is to make user aware of possible data validity problems due to the passage of time.

Another use case is automated data processing: this quality measure helps to select fresher candidate entities to answer user query. For quality issues monitoring tasks, it may be useful to control relative staleness measures which may indicate software or hardware problems in data pipelines. Absolute measures, from the other side, may indicate issues in business processes: somebody is not updating an entity due to absence of a process prescribing this.

4 CONCLUSIONS

In this chapter, we studied data quality aspects for a modern system in a Web era like Entity Name System (ENS). In particular, we demonstrated how one can derive, design, objectively measure and aggregate key data quality dimensions for such a system: consistency, popularity, distinctiveness, representational importance and staleness. More importantly, we demonstrated how those measures can serve the ultimate goal of an information system – to better serve end users. For each of the dimensions, we showed its place in an entire system architecture, as well as concrete use cases driving to have data of higher quality.

Chapter 5
Defining and Measuring Data-Driven Quality Dimension of Staleness

With a growing complexity of data acquisition and processing methods, there is an increasing demand in understanding which data is outdated and how to have it as fresh as possible. Staleness is one of key, time-related, data quality characteristics, that represents a degree of synchronization between data originators and information systems having the data. However, nowadays there is no common and pervasive notion of data staleness, as well as methods for its measurement in diverse applications.

In this chapter, we continue even deeper exploration of system data quality aspects, providing a definition of a data-driven notion of staleness for information systems with frequently updatable data. For such a data, we demonstrate an efficient exponential smoothing method of staleness measurement, compared to naïve approaches, using the same limited amount of memory, based on averaging of frequency of updates.

We also propose an approach for evaluation of algorithms for measurement of data staleness. Space requirements of those algorithms are evaluated against accuracy of their prediction of data staleness. The evaluation approach is tested in our experiments by implementation of proposed three update prediction and staleness measurement algorithms run on Wikipedia's metadata (history of updates of articles), which serves in our experiments as representation of real-world update patterns of web data.

1 INTRODUCTION

Growing diversity of data sources and technologies makes solutions for data synchronization problems more and more complex. Nowadays, data is created, processed, and eventually consumed by the time when it can be already obsolete. The dynamics of today's world raise new challenges for data management, one of which is having data fresh at instant of its consumption.

Since the notion of "data freshness" (or "staleness", as the opposite) is not ubiquitous among researchers, below we will first give a real-world example demonstrating importance of such a time-related quality dimension as staleness. Necessary limitations on possible approaches for data staleness measurement are followed by a description of an information system that may incorporate an approach we will present later in this chapter.

While there are numerous works on data caching and synchronization, including those that consider system-driven data freshness issues (see Section 1.2 of 0), there are underexplored aspects such as understanding and measuring data-driven quality dimensions relevant to time. In particular, when users are the only source of data, system-driven aspects cannot help us as much to detect if a data element[4] is fresh, since those aspects deal with management of data rather than with interpretation of its nature. We will demonstrate this statement in our following examples involving use cases of getting insights from articles of Wikipedia and from entries representing real world entities in an information system called Entity Name System ([109], [6], [110]).

Source systems supplying data to a repository like ENS, can be of two types: push and pull. While systems of the first type send their data to our repository, systems of the second type store data and give interfaces that allow querying for it. For the different types of the sources the goals and methods to measure data staleness differ (Figure 31).

For *push-sources* we build/adjust update models based on factual updates taking place in the ENS. Subsequent updates are then analyzed against update models to find misalignments between predicted and real updates. Inconsistencies are measured to get the staleness estimation, which is used to notify the corresponding sources and/or data consumers about potential staleness in data.

For *pull-sources* we monitor committed updates to those sources (from other systems or users), build proper update models and try to predict occurrence of next updates. Whenever potential updates will be predicted and the corresponding real ones will not appear, potential data staleness will take place. It can be later used to plan queries from pull-sources by exploiting various existing synchronization techniques.

For the pull-sources we assume that we do not have a priori any update model provided from a data source and thus we do not know when we should query for updates. One of the possible solutions in this case will be to use already existing for certain data types (names, addresses, telephone numbers, etc.) statistical data of how often it can be usually updated in other sources. Another solution is to roughly estimate frequency of updates and evaluate costs of having stale data in our system. That will drive in finding a uniform frequency to query corresponding data without building actual update models. However, in this work we focus on understanding the nature of data staleness and its measurement rather than on initialization of prediction algorithms by finding proper update frequency. Once we have initialized the update models, we have to further adjust them based on external information about the monitored entities (based on statistical analysis of their types or based on planned update frequency of the entities reported by external source system) and/or based on constant monitoring of updates at source systems which can be done using "black box" technique.

Measurement of update rate with help of "black box" implies monitoring of a data element with such an update frequency that is greater than the real one at current time. This adjusting method should be used to detect increasing update frequency of data elements that are stored in pull-sources. Decreasing values can be detected by monitoring of elements even with their current update frequencies. Since push-sources make updates on their own, we do not have to implement a "black box" technique to monitor evolution of update frequencies of their data.

[4] *Data element* is an abstraction that denotes attributes ("name-value" pairs) or entities (aggregated set of attributes of notions like person, place, event, abstract object, etc.).

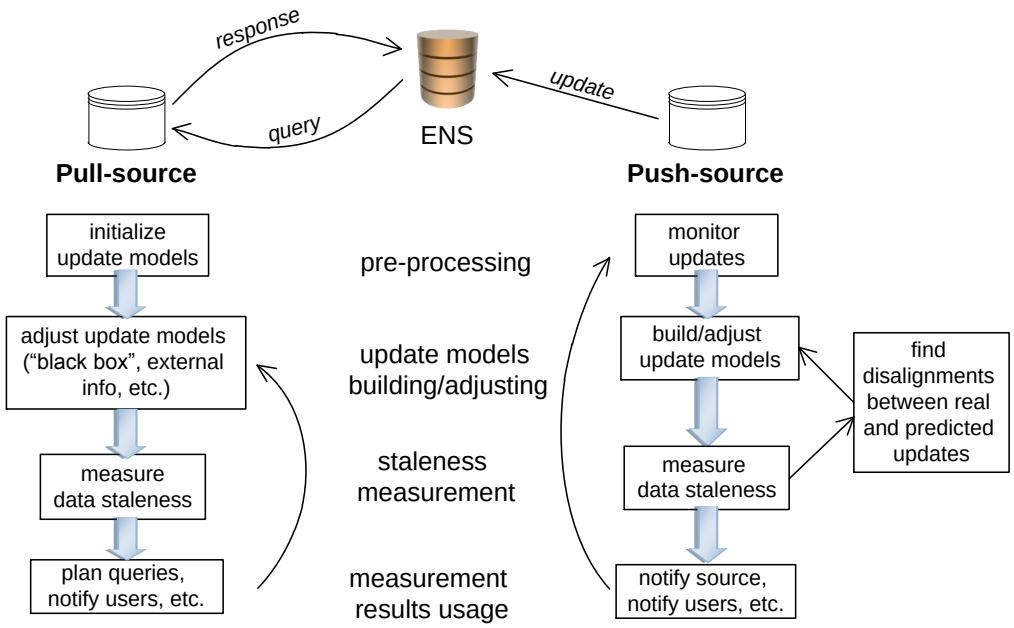


Figure 31. Goals, methods and workflows for staleness measurement for different types of data sources for the ENS.

By having staleness degree of data in a repository measured, one can distribute resources of an information system in such a way to keep the repository partially or entirely as fresh as needed, by means of various synchronization and updates scheduling techniques available. Apart from driving an update strategy, measured staleness values may be reported to end-users to make them aware of a time-related quality aspect of data at hand. Another possible application of the staleness metadata is resolution of inconsistencies in bound data (also assuming that the latest version of data has the most correct values), in data matching (which portion of data better match user queries in terms of its freshness), etc.

As one can see, measurement of staleness of data from pull-sources relies on probabilistic basis. However, since the main goal in this work is to measure data-driven metric of staleness precisely, we consider only push-sources (for a generic database system). As our system is far from pervasiveness of other push-sources, for our experiments we considered Wikipedia as such a source.

In our work, scope of interest in Wikipedia analysis was in analysis of its articles' revisions history, or series of updates of high frequency (about 0.4 to 2.5 updates per day, as we will show later). The lower limit of the update rate is driven by characteristics of the algorithms we explore in this work (Section 4) while the upper boundary is dictated by the natural limitations of Wikipedia of having insignificant number of articles with high update rate throughout selected evaluation interval (we selected history for about 8 years – see details in Section 7). Hence, we target to deal with articles of least 1000 revisions per their lifetime.

Updates to Wikipedia may result not only from legitimate updates from users, but also from robotic or vandalism actions, which are presumed by openness of this resource. Even different viewpoints on the same aspect may result in updates. In our work, we consider all those kinds of updates as a part of an "eco-system", where those events are natural with respect to provided resources and corresponding data alteration rules. Hence, we can build a model that will predict an update of a data element based on history of updates, but not reasons for them. Given that, one can apply our approach to data-driven staleness measurement in other information systems operating with data-driven updates.

One of main questions we want to answer is *How stale is an article now?* For example, given an extraction from revision history of article from Wikipedia "List of web application frameworks" (Figure 32), a disruption of nearly daily updates for more than 2 years may indicate a staleness issue of the article.

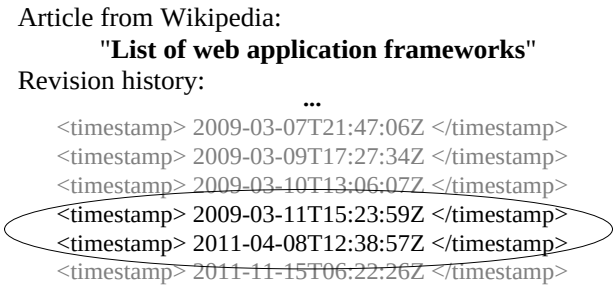


Figure 32. Example of potential data staleness issue in an article from Wikipedia.

Note, that from the other hand, for the measurement of staleness we want to account its semantics as close to real world as possible. In the example above, it would be improbable, that somebody forgot about updates of a popular article on Wikipedia. Thus, for a reader inquiring on how fresh the article is, we want a measurement method resulting, for instance, in "2 days stale" if inquired on 2009-03-14, but "fresh" if inquired a month or later after 2009-03-11, the last day in a series of frequent updates in year 2009[5]. The motivation for that is the following. If one sees a noticeable delay in frequent updates, most probably it will indicate an interruption of updates, i.e., system issues. From the other hand, if one sees a delay, which is much bigger than an average update interval (e.g., a delay of month for a daily updatable data), it may indicate data issues coming from the data originator, rather than storage, replication, propagation or other system issues (which still may be the case). Hence, in the latter case, the data items may be up-to-date (fresh), compared to the data originator, but in the former one, it may probably be not the case due to the different state of real world data originator and its representation user sees.

Since we want to know at any time how stale is a data element, we impose the following strict limitation on the required memory budget, and consequently on the use of a revision history. We want to measure data staleness without storing and analyzing even a (recent) part of update history, but considering it indirectly with help of just a few variables. Those variables must represent update history concisely, at the same time been adequate for accurate measurement (compared to coarse-grained estimation) of a time-related quality dimension. In this work, we will show how we accomplish such a requirement with an exponential smoothing method, comparing it to averaging methods.

Without a limitation on time-series representation, our solution for the problem would become infeasible in real time information systems that must convey to end users staleness measures of millions of entities, with at least thousands of timestamps each. In fact, the problem would eventually fall into another extensively explored problem area – analysis and prediction of time-series with performance optimization aspects.

An example demonstrating not only need for staleness measurement and its exposure to end user, but also its potential propagation to an information system, is an Entity Name System (for the rest of the thesis, we will refer to it as *ENS*, or simply, *repository*) developed within OKKAM[6] project. ENS aims at management of global unique identifiers for entities on the web, storing description of those as sets of attribute name and value pairs for each entity. Users of the system can modify the repository at any time. Usually, the modification reflects updates that took place for corresponding entities in the real world, but as we mentioned before, in this work we study a set of updates as is, without separation of modifications depending on underlying reasons for those.

The question for staleness measurement in the ENS is similar: *How stale is each attribute of each entity?* Another interesting question would be: *Given a staleness value, should an attribute be updated?* In spite of the fact that the latter question is out of scope of the current work, it is of particular interest for data synchronization works, whenever a system like ENS will decide to synchronize some entities with some of external sources like Wikipedia (which may have data staleness measured internally, as we mentioned in the example above). In fact, some works for data caching and synchronization techniques presuppose presence of either fresh data at sources or its measured degree of staleness (see Section 1.2, 0) that can serve for a decision about selection of a source to query.

[5] Naturally, a measure of staleness in this case is relevant only soon after interruption of updates (on 2009-03-11), but not even a month after it: the interruption with the article was due to a new status of the original item – redirecting article.

[6] http://www.okkam.biz

In this chapter, we focus on understanding the nature and setting a data-driven notion of data staleness (Section 2) which is based on existing state of the art that we overview in Section 1.2 of 0. The corresponding approaches to measure the staleness dimension proposed in Section 2 establish a niche for a more generic data staleness measurement by providing a methodology to create, implement (Section 4) and evaluate efficiency of staleness measurement algorithms (Section 6). The implemented sample algorithms demonstrate how the approach can be exploited. We also provide a satisfiability analysis of our notion to base data quality requirements (Section 3).

Along with theoretical reasoning grounded on state of the art and the goals of the ENS, we provide parametrical analysis of staleness measurement triggers (Section 5). As a result, we come with a set of recommendations on choosing corresponding strategy to deal with staleness measurement of data with various update rates.

Section 7 shows experimental results of those approaches implemented and tested on update metadata from Wikipedia, with following comparative analysis of their predictive accuracy in Section 8.

2 NOTION OF DATA STALENESS

Quantitative measurement of a data quality (DQ) dimension requires strict notion defined. Due to diversity of application areas, there is no coherent view on notions of DQ dimensions even in a DQ community. This section aims at a deeper analysis (started in Section 1.2 of 0) of notions for time-related DQ dimensions presented so far. In particular, we will inspect formalization of notions of "freshness" and "age" that we found to be the most relevant ones to the goals of our work.

Cho and Garcia-Molina [62], [63] define *freshness* as a binary measure that indicates whether a data element *e* from the real world is synchronized with its copy in information system. Another quality indicator employed in their works is *age* that has the following semantics: age of fresh data is always zero while age of non-fresh data continuously and linearly grows starting from synchronization misalignment point (i.e., when a copy of a data element is not synchronized anymore). Thus, age measures time elapsed from synchronization misalignment point until instant of measurement.

One of major differences in notions of DQ measures between work of Cho and Garcia-Molina and ours is in definition of misalignment point. They study correspondence of a local copy of web data to its "real-world state" on a remote system, while in our work we do not have a remote system to query the data updates. We rather consider users that may update the data at any time.

From the other hand, both "freshness" and "age" in [62] served for studying best strategies to keep cache of web data fresh. In our work, we consider a different problem, namely prediction of next updates of each monitored data element without analysis of history of updates (revisions). As we mentioned before, the history, nevertheless, must be considered indirectly. Thus, for enabling measurement of data staleness, in our work we define it via linear function corresponding to that of age defined by Cho and Garcia-Molina [62], but with their semantics of freshness.

2.1 DEFINITION OF DATA STALENESS MEASURE

Ideally, a data-driven notion of staleness should not consider the fact of synchronization between systems, but it will rather approach to derive a measure of correspondence (in time) of a data element's value to its real world's instance. In reality, it is often too hard or just impossible to get real-world's value. Hence, we consider user updates, which potentially represent alterations in the real world, but in any case, those updates represent changing web data we want to study in this work.

We suppose that whenever a system misses update that has to take place, it will have data element that does not correspond to the real world. Instant of such a missing update we call potential synchronization misalignment point, or instant of potential update. To measure staleness of a data element e at current time t_{cur} we have to find potential synchronization disalignment points of that element, in respect to its real updates (e.g., made by users to our system) and those predicted by an algorithm.

Whenever predicted (P) update will precede the corresponding prospective real (R) one, there is a possibility of presence of staleness. In this case, data staleness $S^e_{t_{cur}}$ of element e at current time t_{cur} can be measured as difference between current time t_{cur} and time of predicted update t^e_{pr}; otherwise, if predicted update will take place on or after current instant, staleness will remain zero:

$$S^e_{t_{cur}} = \begin{cases} t_{cur} - t^e_{pr}, & t_{cur} > t^e_{pr} \\ 0, & t_{cur} \leq t^e_{pr} \end{cases} \quad (5.1)$$

Graphical examples of two possible cases of staleness measurement (according to eq. 5.1) are shown in Figure 33, where diamonds represent real updates, whether they already took place (solid lines) or they are potential future ones (dotted lines).

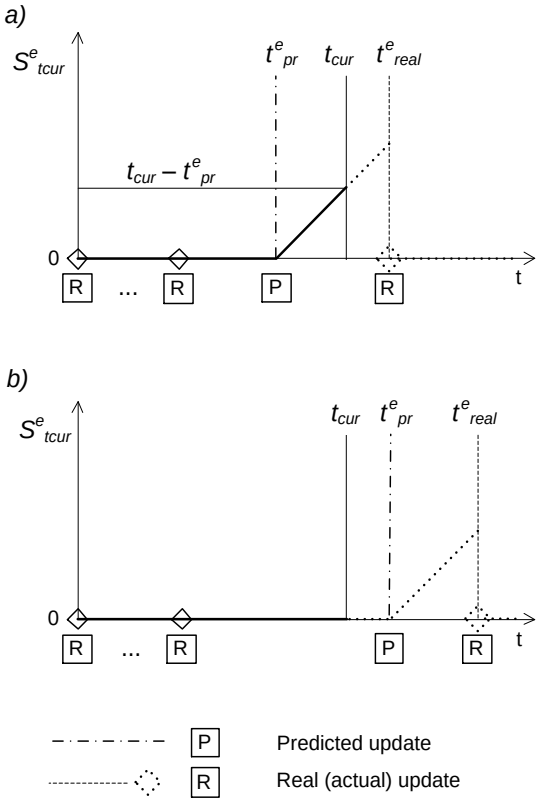


Figure 33. Possible cases of staleness measurement (eq. 5.1)
a) instant of measurement (t_{cur}) falls between predicted and real updates;
b) instant of measurement precedes both predicted and real updates.

3 REQUIREMENTS FOR TIME-RELATED QUALITY METRICS

There are various requirements for DQ metrics from researches of the field. Heinrich et al. [46] have gathered a coherent set of key requirements for time-related metrics. Among those are (R1) normalization, (R2) interval scale, (R3) interpretability, (R4) aggregation, (R5) adaptivity and (R6) feasibility. In the following subsections, we will describe each of these requirements and demonstrate satisfiability to each of them of our notion and approach to measurement of data staleness. We will also demonstrate compliance of our notion to principal requirements for DQ dimensions outlined by Pipino et al. [12].

3.1 NORMALIZATION

Depending on an application context, one may need normalization of a data quality metric while operating with more than one metric (instead of a sole one we considered in our examples dedicated for staleness measurement). Data values normalization in these lines means mapping of all possible measurement values of a data quality metric to interval 0 to 1. For our staleness metric, one of the possible options to obtain a normalized measurement in interval]0;1] is by using an exponential function:

$$S^e_{N,t_{cur}} = \begin{cases} \exp(-S^e_{t_{cur}}), & t_{cur} > t^e_{pr} \\ 1, & t_{cur} \leq t^e_{pr} \end{cases} \quad (5.2)$$

with values ranging from 0 (absolutely stale data element – reachable only at infinite value of an exponential function argument) to 1 (the best possible quality of data element on the dimension of staleness), according to [12].

We should note, that in systems enforcing specific update policies, state of "absolute stale data" may be reachable at a finite time, putting $S^e_{N,t_{cur}}$ to 0 according to given data aging policies. For example, consider a system where each user must change own password every 6 months. Those passwords without been updated during more than 6 months, are not valid in the system, and can be treated as absolute stale elements. However, such cases are outside of scope of our work, since policy enforcement and data invalidation is covered to a higher extent by another quality dimension – validity.

As we mentioned, normalization and its specific cases may be needed while comparing quality metrics in an entire system context rather than only measuring them. As in this chapter we aim at the measuring and experimentation with a sole metric of staleness, the rest of the DQ requirements will be tested against staleness metric defined in eq. 5.1.

3.2 INTERVAL SCALE

To support interpretability of quality measurement results for their comparison with measurements of other dimensions, those results should support interval scale property. This means that the same interval should denote the same quality improvement or degradation.

This property is supported by our (non-normalized) definition of staleness due to its linearity: difference of an element's staleness between values 3 and 4 (days) means the same as the one between 6 and 7 (days).

Note, that interval scale function may differ from the linear function (eq. 5.1), or be impossible to pursue in some cases – normalization is one of those for our notion. Hence, depending on application, staleness measurement function may be either normalized (to facilitate comparison between quality metrics), or interval scaled (for better interpretation of the results).

3.3 INTERPRETABILITY

Easiness of interpretability of measurement results by end-users is also important for definition of a quality metric. For example, when an analyst has data with freshness metric equals to 0, does it mean to have fresh data at hand? What about freshness equals to 10 (suppose, we do not stick to the notion proposed in [62])? Is it even fresher? Similar issues may arise with the notion of age: e.g., with age *A(e) = 0*, we cannot undoubtedly speak about positive or negative data characteristic because of a semantic meaning of "age" that mostly corresponds to a neutral notion of "period of time" [111]. Unless specific notion of freshness or age is communicated to the end-user, interpretation of that may be ambiguous. To reduce such an ambiguity, we came with a notion that comprehends time-related characteristics of data, simplifying its perception by end user.

Considering the abovementioned example, with staleness $S^e = 0$, we speak about absence of (a time-related) negative feature, while $S^e > 0$ indicates potential quality issues with data, measuring an extent of those according to eq. 5.1. Hence, from a user perspective, the notion of data staleness satisfies the requirement of interpretability, suggested by [20].

3.4 AGGREGATION

While measuring quality metric at one level, it is important to get aggregate value at higher one(s). For example, having result of staleness measurement for attribute "name", how it will influence staleness of a corresponding entity, table and entire database?

We define aggregation property at database level as ratio of weighted sum of all measures of staleness of all measured data elements (attributes, entities, etc.) in a database, to their amount:

$$S^{DB} = \frac{\sum_{e \epsilon DB}[imp^e \cdot S^e]}{|e \epsilon DB|} = \frac{\sum_{e \epsilon DB}[\frac{Q^e}{Q^{DB}} \cdot S^e]}{|e \epsilon DB|} \quad (5.3)$$

where importance rate imp^e represents weight of an element *e* in a database, Q^e is number of queries for data element *e*, Q^{DB} is total number of user queries in a system.

Note that definition of importance rate imp^e depends on its application and system context. Since in this work we focus on measuring data staleness dimension, for simplicity reasons we consider normalized number of queries for a data element as its measure of importance – the more data element is queried, the more important it is in our system.

3.5 ADAPTIVITY

Usually, to interpret measurements of quality metrics, those metrics should be adopted for a given context. While this is true only for some metrics, as [9] noted, most of them are context-independent and can be objectively evaluated without such an adaptation.

By definition, staleness is one of those objective metrics. However, as we have mentioned before, one can enforce adaptability of low-level measurement results for higher-level DQ assurance goals. For example, data administrator may set a warning if attribute staleness reaches a certain threshold, and may set an automated request for update if staleness will reach even higher threshold.

3.6 FEASIBILITY

Techno-economical requirements of applications where quality measurement takes place, imply feasibility of getting the results. For example, getting a measure of reputation of an external source may be infeasible in some cases.

As we will show in the next section, our approach for getting staleness measure relies on parameters that are essential and normally easy to get for a data element at a source system – total number of updates, timestamps of first and last update, etc.

4 APPROACHES TO DATA STALENESS MEASUREMENT

In the previous sections, we have established a data-driven notion for quality dimension of staleness (Section 2) and properties that the corresponding measurement methodology should possess (Section 3). In this section, we will demonstrate examples of possible approaches to measure data staleness under given constraints on variables for representation of history of updates.

In particular, in Section 4.3 we will present an exponential smoothing method that indirectly considers (by means of a few representative variables) history of updates. This approach is preceded by two naïve ones, enhanced averaging method and shifting window, that calculate instant of predicted update t_{pr}^e for eq. 5.1. In Section 7, the naïve approaches serve as a baseline to compare accuracy of prediction results of the proposed methods.

4.1 ENHANCED AVERAGING METHOD

Consider a data element *e* that has nearly periodic updates committed by users of an information system. We call such updates "real" ones (as opposed to "modeled" that are approximations of real updates, extrapolated by a measurement algorithm, and "predicted" that is a sole next update from time of measurement, forecasted by the algorithm).

According to eq. 5.1, to measure data staleness $S_{t_{cur}}^e$ of element *e* at instant of time t_{cur} (a test point which is usually current time), we need to have instant of time of predicted update t_{pr}^e, that most probably should have taken place for *e* in the period between instant of last update t_{last}^e and t_{cur}. It seems to be a straightforward task for periodically updatable data element. Number of all updates committed for data element up to test point t_{cur}, gives us an average update rate during past time interval. By extrapolating for one more period, we can have an instant of potential update t_{pr}^e, and hence, $S_{t_{cur}}^e$ (see Figure 34).

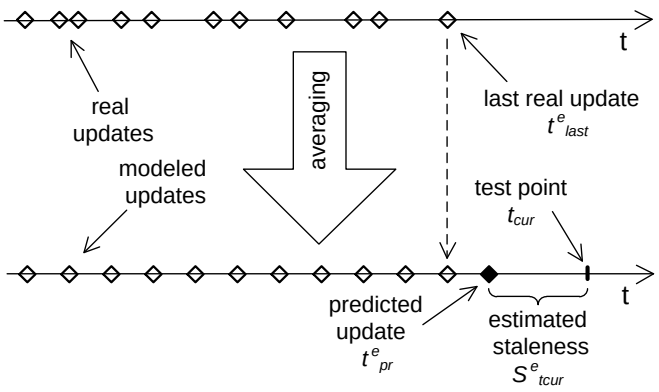


Figure 34. Example of prediction of element's update by enhanced averaging method.

This method, however, may give an approximation of staleness with as few as only two stored parameters for each monitored data element – number of updates and time interval during which they have been committed. In this work, we study an *enhanced averaging method* using one more stored parameter – timestamp of last update.

Formally, we define instant of predicted update of element *e* using this method as following:

$$t^e_{pr} = t^e_{last} + \frac{1}{\lambda^e} \tag{5.4}$$

or, substituting update rate λ^e with corresponding ratio of number of updates N^e during element's lifetime from instant of its creation t^e_0 to its last update t^e_{last} over that period of time, we have:

$$t^e_{pr} = t^e_{last} + \frac{t^e_{last} - t^e_0}{N^e} = \frac{t^e_{last} \cdot (N^e + 1) - t^e_0}{N^e} \tag{5.5}$$

This gives us staleness measure of element *e* at time t_{cur} using enhanced averaging method:

$$S^e_{t_{cur}} = \begin{cases} t_{cur} - \dfrac{t^e_{last} \cdot (N^e + 1) - t^e_0}{N^e}, & t_{cur} > t^e_{pr} \\ 0, & t_{cur} \le t^e_{pr} \end{cases} \tag{5.6}$$

Hence, to measure data staleness using this naïve method we would need to store 3 variables for each data element: instants of first and last updates t^e_0, t^e_{last} and total number of its updates N^e.

In spite of the fact that most of web data we operate with, is following Poisson distribution (showed by [62]), entities with aperiodic or irregular updates motivates us to consider more accurate prediction methods that have to account such irregularities. Those methods, from the other hand, must be at least as efficient (in terms of variables per data element) and accurate as the enhanced averaging method.

4.2 SHIFTING WINDOW

One of the simplest methods accounting entities with aperiodic updates without boosting number of variables for staleness measurement, is shifting window [112]. It implies analysis of a given time series by means of grouping of consecutive time series points into time intervals of fixed or adaptive size, depending on the intended method. A key advantage of this method in our work is in localizing periods with different frequencies of updates λ^e_1, λ^e_2, ... within certain intervals – windows (Figure 35).

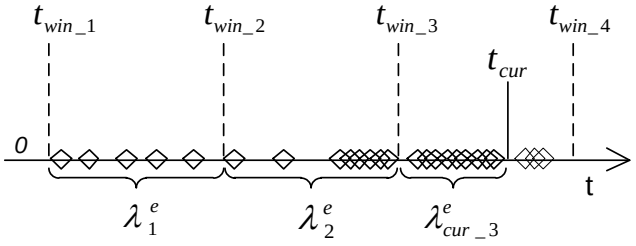


Figure 35. Irregular updates delimitation by shifting window.

Shifting window can be seen as a refinement of the averaging method described before (Section 4.1) in such a way, that prediction of next update considers only recent updates (of current window) rather than older ones, unless current window has a few of them, as we will show later.

In this work, we focus on implementation of this method with window of fixed size, rather than on analysis of window size itself (which may also require from a data analyst manual processing of history of updates of entities of the same type). Hence, we will take a shifting window of fixed length $|t_{win}|$, instances of which will sequentially cover entire lifetime of each data element under staleness measurement, as in Figure 35.

From the definition of this method it is obvious, that in most cases with web data even a few windows per lifecycle of an entity should improve accuracy of staleness measurement (as we mentioned, we target entities with update rate of order at least 1000 per lifecycle). From the other hand, computational resources and intervals with sparse updates dictate upper limit on number of windows for each application.

Assigning time periods $t = \{0,1, \dots\}$ that correspond to each of these windows, we will operate with time stamps indicating start of each of such window $t_{win\_t} = \{t_{win\_0}, t_{win\_1}, \dots\}$.

For each window of current period t, we keep tracking number of updates committed since instant of start of window t_{win_t} and until current (or test) instant t_{cur}. This gives us current frequency of updates in current window of period *t*:

$$\lambda^e_{cur\_t} = \frac{N^e_t}{t_{cur} - t_{win\_t}} \tag{5.7}$$

However, as we mentioned before, until there are enough updates in each new window, we cannot realistically estimate data element's expected update frequency to predict each new update of an element inside that window. Hence, we set a threshold of having at least three updates for each new window before we start calculating element's update rate inside the new window. Before we will get sufficient updates in current window (while $N^e_t < 3$), we consider previous window's update frequency also as current one; otherwise, current update frequency is calculated as ratio of committed updates in the current window to elapsed time from instant of start of the window to current instant:

$$\lambda^e_{cur\_t} = \begin{cases} \lambda^e_{t-1}, & N^e_t < 3 \\ \dfrac{N^e_t}{t_{cur} - t_{win\_t}}, & N^e_t \geq 3 \end{cases} \tag{5.8}$$

Combining eq. 5.1 for case $t_{cur} > t^e_{pr}$ with eq. 5.4 and eq. 5.8 (given that $\lambda^e_{cur\_t} = \lambda^e$), we have:

$$S^e_{t_{cur}} = \begin{cases} t_{cur} - \left(t^e_{last} + \dfrac{t_{cur} - t_{win\_t}}{N^e_t} \right), & N^e_t \geq 3 \\ t_{cur} - \left(t^e_{last} + \dfrac{1}{\lambda^e_{t-1}} \right), & N^e_t < 3 \end{cases} \tag{5.9}$$

As one can see, this method requires the following variables: 1) number of updates in current window N^e_t, 2) previous window's update rate λ^e_{t-1} and 3) time of last update t^e_{last}. Hence, this methods turns out to be as efficient in terms of required variables, as enhanced averaged-based method.

Note, that since we operate with entities of the same high order of update frequency (as we mentioned in the introduction of this chapter), we can adjust boundaries of all shifting windows $t_{win\_0}, t_{win\_1}, \dots$ for all data elements, with precision of order t. In fact, such an adjustment can reduce only length of t_{win_0} that we treat as time of entity initialization. Thus, we can store and track all boundaries for shifting windows for entire data repository, instead of individual elements.

For the entities with high update rate, we may not necessarily need a precise calculation of predicted instant of update. Instead, we can try to operate with small intervals, within which a method will predict presence or absence of update(s). The next section demonstrates implementation of this idea via exponential smoothing method.

4.3 EXPONENTIAL SMOOTHING

As we mentioned before, staleness measurement with help of shifting window uniformly considers all updates within a window of size $|t_{win}|$. In this section, we will show how one can exponentially account a wider (up to entire lifecycle) scope of updates, reducing potentially unnecessary precision of prediction for each next update instant for elements of high order of updates that we deal with.

Among a wide variety of exponential smoothing methods [113], we focus on the N-N methods, which do not consider trend and seasonality of time-series. We adjusted one of those methods for prediction of presence of an update during time period. This method is similar to shifting window method (Section 4.2), but with much finer granularity with respect to update frequency.

One of key elements in measuring data staleness with exponential smoothing method is a time interval that for better reflection of its semantics we called Predictable Time Interval (PTI). Such an interval is an equivalent of shifting window of granularity of order of updates. This means that, on average, for each update event there should be a few PTI's.

For all updates in each PTI we assign exponentially growing weights, indicating importance of those in prediction of future updates (see Figure 36), and hence, measuring current staleness value (we will give more details on the weights later on in this section).

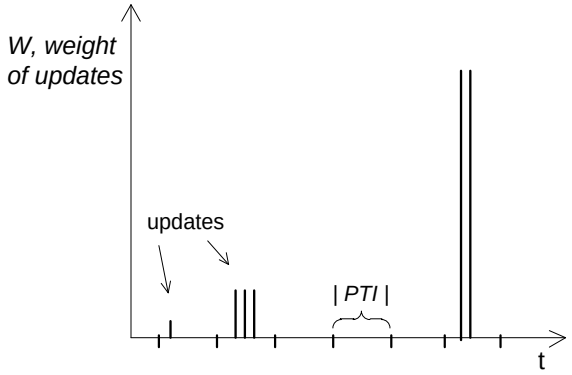


Figure 36. Example of exponential growth of weights associated with updates in each PTI.

The choice of length of those intervals depends on foreseen update rate of observed data element, which can be also estimated based on historical analysis of entities of same type. Hence, one can approach to calculation of Predictable Time Interval length (in hours) as follows:

$$|PTI| = \frac{\frac{\sum t^e}{|e|}}{\frac{\sum N^e}{|e|} \cdot PTI_{PC}} \cdot 24 = \frac{1}{\lambda \cdot PTI_{PC}} \cdot 24 \qquad (5.10)$$

where $\frac{\sum t^e}{|e|}$ is average lifetime (expressed in days in our case) of elements in a set; $\frac{\sum N^e}{|e|}$ is expected average number of updates of elements in a set during their lifetime; λ is an average update rate of elements of a set; PTI_{PC} is a precision coefficient, indicating required granularity of a predictable time interval. This coefficient depends on application, and to a higher extent, on expected update frequency and nature of data – we explain this below.

Since our ultimate goal is measurement of data staleness, which is a data characteristic that can be useful in a decision making, we impose the following empirical granularity limits on rate of PTIs (or "tick size") to frequency of updates. From one side, a frequency of updates higher than about 2 events per tick interval will deprecate value of prediction result of the presented algorithms since in most predictions the algorithms will foresee an updatable tick providing little value for staleness assessment. On the other hand, accountability of the recent updates (a few recent ticks range) imposes the lower limit – for the input data we want to have no less than 1 update event per about 3 ticks.

To evaluate more precisely the upper and lower boundaries and get for the measurement of data staleness a range of best values of $|PTI|$ length, our experiments (Section 7) suggested the precision coefficient PTI_{PC} to be 1.2 – for the nature of data updates, as well as the frequencies of the updates we experimented with. Hence, staleness of similarly updateable data ($0.4 \leq \lambda \leq 2.5$, or 1000 to 7500

updates during about 6 years) and of similar update nature, can be measured by using PTI intervals from 8 to 49 hours, as formula 10 suggests.

Now, we want to assign a weight to each PTI that will depend on number and recency of all previous updates: for prediction of each next update we consider more recent updates (closer to instant of measurement *t*) to a higher extent by gradually assigning higher weights to them. In this way, we define averaged weighted linear combination of all updates of a data element *e*:

$$Savg_t^e = \alpha \cdot N_t^e + \alpha \cdot (1-\alpha) \cdot N_{t-1}^e + \alpha \cdot (1-\alpha)^2 \cdot N_{t-2}^e + \ldots \quad (5.11)$$

where N_t^e is number of updates that element *e* had for time interval between *t-1* and *t* (the most recent); α is a smoothing constant such that $0 \leq \alpha \leq 1$ (this constant drives scale of growth of weights, associated with each PTI).

From eq. 5.11 we have the following recursive formula:

$$Savg_t^e = (1-\alpha) \cdot Savg_{t-1}^e + \alpha \cdot N_t^e \quad (5.12)$$

We consider weighted linear combination of each PTI $Savg_t^e$ as a probability that the next PTI will have an update. Whenever $0.5 \leq Savg_t^e < 1$, it shows us that the next time interval should have at least one update. The same prediction applies to $Savg_t^e \geq 1$, which shows us that previous PTI accommodated more than one update. $Savg_t^e < 0.5$ indicates that next PTI most probably will not have an update.

While measuring data staleness with this method, we also need for each PTI interval a staleness coefficient PTI_t that will indicate how stale a data element is (or it is not stale but fresh). This makes this method a perfect fit as a solution to our motivating example (Section 1 of this chapter), where we imposed a requirement for a measurement method that it should result in a corresponding element's staleness value soon after last update. From the other hand, it should consider that element fresh if last update took place long time ago, compared to expected update rate of element.

Staleness coefficient PTI_t has the following logic. As soon as predicted presence or absence of updates during the next PTI corresponds to real case, PTI_t equals to 0; in case of inconsistency between prediction and real case, PTI_t is increased by 1 if $Savg_t^e$ suggests an update, unless it is not reset to 0 by presence of a real update event during past PTI – see Table 4.

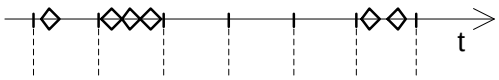
Table 4. Logical table for PTI_t staleness coefficient

Presence of updates (prediction by $Savg_t^e$ / real case for interval *t*)	
No / No	$PTI_t = 0$
No / Yes	$PTI_t = 0$
Yes / No	$PTI_t = PTI_t + 1$
Yes / Yes	$PTI_t = 0$

Eventually, one can get staleness of element *e* at current time by multiplying PTI_t staleness coefficient by the corresponding length of PTI:

$$S_{t_{cur}}^e = PTI_t \cdot |PTI| \quad (5.13)$$

Figure 37 demonstrates on a sample history of updates, an instantiation of the exponential smoothing method, with $\alpha = 0.8$. In this figure, we can see initial 6 intervals (PTI's) delimited on the timeline by instants $t = 0..6$. Each rhombus in the figure denotes an update to an element. According to eq. 5.12, Table 4 and eq. 5.13, we calculate values of $Savg_t^e$, PTI_t and eventually, staleness value S_t^e, at each instant of time $t = 1..6$ (setting 0 as corresponding initial values at the first step).



t	0	1	2	3	4	5	6
$Savg^e_t$	0	**0.8**	**2.56**	**0.512**	0.1	0.02	**1.6**
PTI_t	0	0	0	1	0	0	0
S^e_t	0	0	0	1*|PTI|	0	0	0

$Savg^e_t$ averaged weighted linear combination of history of updates

PTI_t staleness coefficient of Predictable Time Interval (PTI)

Figure 37. Example of staleness estimation with exponential smoothing method.

As one can see, in terms of stored variables, this method is as efficient as those presented in this section before: it requires 3 individual variables for measurement of data staleness, namely 1) weighted linear combination for previous time interval $Savg^e_{t-1}$; 2) number of updates for current interval N^e_t and 3) staleness coefficient PTI_t. The rest of required variables ($|PTI|$, α, boundaries of PTI's) one can set common for entire data repository, or individual for some of data elements, depending on availability of resources and a given task.

In our experiments, we will demonstrate that exponential smoothing method gives the most accurate update prediction results (and hence, more accurate staleness measurement at equal other conditions) with fewer errors for entities with aperiodic updates.

5 MEASUREMENT TRIGGERS PARAMETRICAL EVALUATION OF THE PRESENTED PREDICTION METHODS

Staleness measurement assumes calculation of such key element as t^e_{pr}, predicted instant of potential update of element e. Having such instant, calculation of current staleness value $S^e_{t_{cur}}$ will become a trivial task (one may simply evaluate difference of current time and t^e_{pr}) with well predictable resources required for that.

Undoubtedly, resources for calculation of t^e_{pr} will depend on prediction method used. However, leaving out a prediction method selected for a particular task, in this section we focus on studies of possible options for suitable measurement of data staleness considering consumption aspects of those results. In particular, when one measure time-related quality dimension, it is equally essential to understand not only how to measure (an approach/method), but also *when* to measure and *how* to deliver the results. To answer these questions, below we will consider possible staleness measurement triggers which are events at such instants when staleness of monitored data elements is measured, and then we will compare space requirements for various trigger options.

The best answer to the question "when to measure staleness of a certain data element" is "whenever it will be needed". This means that whether our data will be updated very frequently or will not be updated at all, it may be stored in our system with no staleness measures as long as it is not queried. As soon as user or system component queries our data, we should have its staleness measurement complete. Hence, we have the first trigger event as *query* for data. In this case, we do not need to store any measured value (like instant of next update t^e_{pr} for the averaging methods presented before) for each data element since we can directly convey measured staleness value $S^e_{t_{cur}}$ to the stakeholder.

However, not always can we do measurement at each query instant – frequent ("heavy[7]") queries can do this task infeasible. Another type of trigger can mitigate this problem – at *update* time. Using this

[7] we consider those loads making it impossible for a system to provide data staleness measurements in time as per specified obligations

trigger, though, we may have (for those methods using it) an additional parameter to store for each monitored data element – t_{pr}^{e} – before one will query it and we can calculate current staleness of an element. This trigger one can use if query load will not allow our system to compute t_{pr}^{e} points for each monitored data elements on the fly, but pre-compute those values in advance.

Table 5 summarizes parametrical analysis of trigger-aware usage of the proposed three staleness measurement methods presented before – enhanced averaging (EA), shifting window (SW) and exponential smoothing (ES). In this table, the final result ($S_{t_{cur}}^{e}$) in all the three cases is computed on each query by using corresponding stored or pre-computed parameters.

Note, that if a data element will experience "heaviness" for both queries and updates, one should re-consider application of staleness measurement methodology for such elements. For example, while monitoring thousands of stock market indicators which may change their values every second, is there need to calculate their staleness provided that trivial system-driven data propagation delay may be normally detected and properly handled is such systems?

Table 5. Parameters to store per data element for the proposed staleness measurement methods considering existing loads of queries and updates.

<table>
<tr><td colspan="2"></td><td colspan="2">Updates</td></tr>
<tr><td colspan="2"></td><td>moderate</td><td>heavy</td></tr>
<tr><td rowspan="2">Queries</td><td>moderate</td><td>Measurement trigger type: on query
EA: $N^e + t_{last}^e + t_0^e$
SW: $N_t^e + t_{last}^e + \lambda_{t-1}^e$
ES: $N_t^e + Savg_{t-1}^e + PTI_t$</td><td></td></tr>
<tr><td>heavy</td><td>Measurement trigger type: on update
EA: $N^e + t_{last}^e + t_0^e + t_{pr}^e$
SW: $N_t^e + t_{last}^e + \lambda_{t-1}^e + t_{pr}^e$
ES: $N_t^e + Savg_{t-1}^e + PTI_t$</td><td>N/A</td></tr>
</table>

Hence, while keeping in mind space optimization criteria, one may use query-based trigger whenever system resources will allow that or update-based trigger otherwise, unless such method as exponential smoothing (requiring constant number of stored parameters) is used.

6 EFFICIENCY EVALUATION OF STALENESS MEASUREMENT ALGORITHMS

While having staleness prediction methods, we want to understand how good are they for the task they are designed, considering specifics of the notion of data-driven staleness we provided before. For that, we will consider those methods in a test environment where we evaluate prediction of update points from within time series at hand.

6.1 EVALUATION OF ALGORITHM WITH EXACT PREDICTION

6.1.1 TYPES OF PREDICTION ERRORS

Between two types of possible errors in prediction of update points we differentiate between a) "positive" prediction – when an algorithm predicts an update that succeeds the real one and b) "negative" prediction – when an algorithm predicts an update that precedes the real one.

In the "positive" prediction case (Figure 38, a) as soon as we reach a real update point we dismiss predicted update and recalculate the next one at time when a measurement policy will require it (e.g., on update or on query event). In this case, in spite of a possible error, data element is considered fresh up to its real update. Whether or not accuracy of the prediction algorithm is low, there is no implication of prediction error on reported staleness value.

In the case of "negative" prediction (Figure 38, b) data is considered stale from instant of predicted update to the corresponding real one. If a suggested update model is incorrect, implications of

the results of prediction have a negative impact: the algorithm indicates that data element is stale while in reality it is fresh. Knowing about level of algorithm's accuracy, a strategy on the prediction results usage can be worked out for the following two possible cases. Output of the algorithms with high accuracy can trigger an update request from the corresponding data sources (probably possessing fresh version of stale data of our repository). Output of the algorithms with low accuracy can be used to communicate the result on measurement to data operators (or other software modules of our system) of *possible* staleness problems, no automatic actions should be triggered without additional metainformation about reported data.

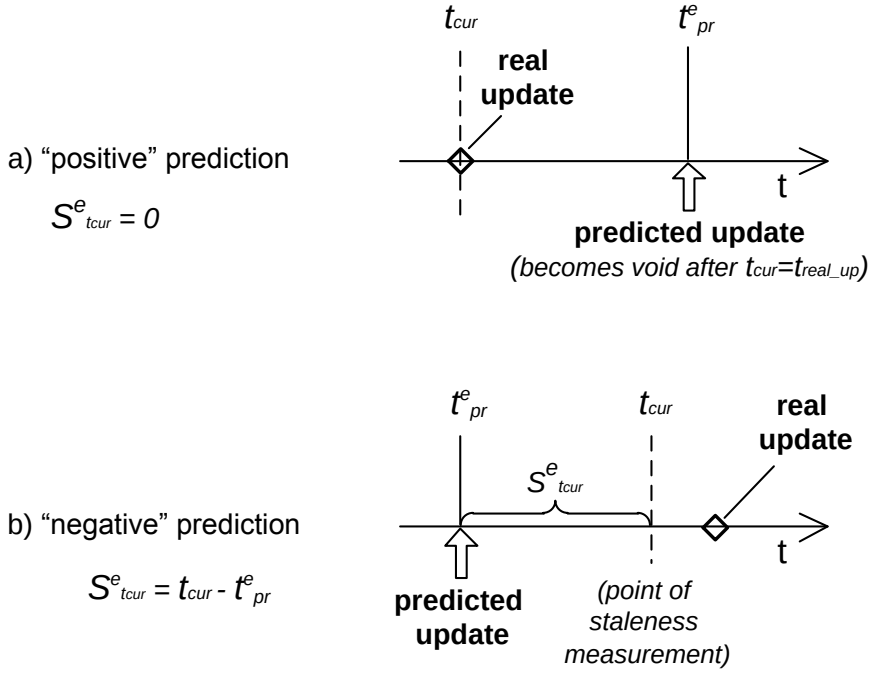


Figure 38. The possible outcomes of data staleness prediction.

The rate of "positive" predictions among all of them we call *prediction outstrip factor* (POF), which we will use with other indicators to evaluate accuracy and "prediction safety" we consider below.

6.1.2 EVALUATIVE INDICATORS FOR ALGORITHMS WITH EXACT PREDICTION

To evaluate efficiency of instantiated algorithms for measurement of data staleness, we propose to use with a well-known efficiency metric like *precision* those indicators that are specific and also very important for staleness prediction: *prediction outstrip factor* (POF) and *mean error*.

POF shows the fraction of "positive" predictions of updates in total number of updates, that is:

$$POF = \frac{N_{P>R}}{N_{tot}} \tag{5.14}$$

where $N_{P>R}$ is number of predictions that succeed corresponding real updates; N_{tot} is total number of updates committed to a database.

As we have mentioned before, "positive" predictions are safer in sense of possible outcome of a prediction error. Thus, this factor shows us how safe prediction results are, or how much we can trust those results. This will influence the way algorithm will be integrated into entire system. In particular, algorithms with high POF will give results that may be further used (without human involvement) to request for update of possibly stale version of data element in our system; in case of low POF, additional indicators should be considered to take a decision whether to request for update of data element.

From the other side, this indicator can show us only how safe predictions of an algorithm are, but it does not show how good they are. For example, if an algorithm will always predict end of timeline for all updates, POF will be 100% (all decisions are safe, there are no false indications of data staleness) but failure of such prediction will be also 100%. Whether or not there will be update or data element will become stale, such an algorithm will not detect these issues.

To have an error indicator that will show extent to which a given algorithm is accurate, we need to measure its absolute prediction error – *mean error*. For some algorithms that predict exact time of

update, this indicator will reflect more accurate errors, while in those algorithms predicting presence of updates for time intervals, we can take mid points of those intervals to have averaged prediction errors.

$$MErr = \frac{\sum|P - R|}{N_{tot}} \tag{5.15}$$

where $\sum|P - R|$ is a sum of absolute values of differences between predicted updates and corresponding real ones; in case of interval-based update prediction, the midpoint of an interval should be taken as time *R*; N_{tot} is total number of updates committed to a database.

6.2 EVALUATION OF ALGORITHMS WITH INTERVAL PREDICTION

6.2.1 EVALUATIVE INDICATORS FOR ALGORITHMS WITH INTERVAL PREDICTION

For algorithms that predict updates for intervals, the prediction outstrip factor loses representativeness of algorithm's efficiency. If we consider the case when *POF = 100%,* this will mean that algorithm's predictions never falls into PTIs. Hence, the more representative indicator for efficiency of such algorithms would be rate of *prediction correctness (PCorr),* which indicates fraction of correct predictions in total number of predictions. We consider as "correct" those predictions, which coincide with real case of presence or absence of updates for a set of conventional intervals, which are bounded with limits of predictable time intervals (PTI) introduced in Section 4.3. Hence, prediction correctness rate can be computed as follows:

$$PCorr = \frac{N_{P_T \in R \pm |PTI|} + N_{P_F \notin R \pm |PTI|}}{N_{tot\_pr}} \tag{5.16}$$

where $N_{P_T \in R \pm |PTI|}$ is number of predictions of updates that fall into regions with centers as real updates (as we mentioned before, $|PTI| = \frac{1}{\lambda \cdot PTI_{PC}} \cdot 24$, and $PTI_{PC} = 1.2$); $N_{P_F \notin R \pm |PTI|}$ is number of predictions of absence of updates that fall outside regions with center as real updates; N_{tot_pr} is total number of predictions made for algorithm's precision evaluation in certain experiment.

7 EXPERIMENTAL EVALUATION OF THE PRESENTED METHODS

In our experiments, we compared the best predictive accuracy of the approaches proposed in Section 4, given the least error rates of each of the method with different values of the corresponding parameters (tick size, shifting window size and α coefficient). In particular, we measured how often a method can correctly predict every next update of each article, and how many errors it makes. By erroneous prediction we consider presence of non-predicted update and absence of a predicted one. Table 6 provides a summary of the cases of correct and erroneous predictions accountability in our analysis methodology.

Table 6. Correctness of update predictions with respect to real updates.

		Real update	
		present	absent
Predicted update	yes	OK	Error
	no	Error	n/a

As one can see from their description, enhanced averaged-based approach and shifting window can predict a precise instant of update, while exponential smoothing method predicts presence of an update in an interval. To compare predictive accuracy of those methods, in our experiments we have adopted the two former approaches for prediction of next update during a time interval (predictable time interval, PTI) of the same length that exponential smoothing method will use.

As a dataset, we took revision history of articles in English from Wikipedia[8], covering around 3.8 million items from January 16, 2001 to December 01, 2011 (we used for reference purposes a time-series[9] representing metadata of articles till May 12, 2009; in particular, it helped us to verify motivating example we gave in Section 1 of this chapter). As we intended to explore the algorithms targeted to deal with a data that is getting updates at high frequency (at least 1000 updates per lifecycle, which is at most 11 years for the Wikipedia's dump we analyzed), we selected groups of articles (Table 7) by number of their revisions.

Table 7. Articles revision groups

Articles group (number of revisions)	Initial number of articles in a group	After filtration number of articles in a group
1000 ± 10	601	495
1500 ± 10	191	157
2000 ± 10	95	83
2500 ± 10	54	47
3000 ± 10	19	15
3500 ± 10	15	12
4000 ± 10	16	13
4500 ± 10	14	14
5000 ± 10	2	1
7500 ± 150	5	3

Articles of these groups 1000±10 to 5000±10 we used as an input data for exploration of properties of the described algorithms. Due to the natural limitations of Wikipedia (among all articles, there is significant decline of number of those with a few thousands revisions), we also selected articles with revisions 7500±150 (no. 5) that we used for extrapolation and confirmation of the results at higher update rates.

For more competitive comparison of the results from the presented algorithms, we want to align articles revision histories lengths, i.e., eliminate recently (close to the final point of the revision metadata taken) created articles that fell into one of our groups, therefore, making update density of each group of articles more equal. For example, Figure 39 represents updates distribution of two sample articles of the same update rate – 5000. Among those two kind of articles we are interested the most to experiment with, is the one having wider distribution in time of the same number of revisions, and with less global peaks, which should potentially give more credit to those prediction methods that can adapt faster to varying update density (shifting window and exponential smoothing will demonstrate this in the following experiments).

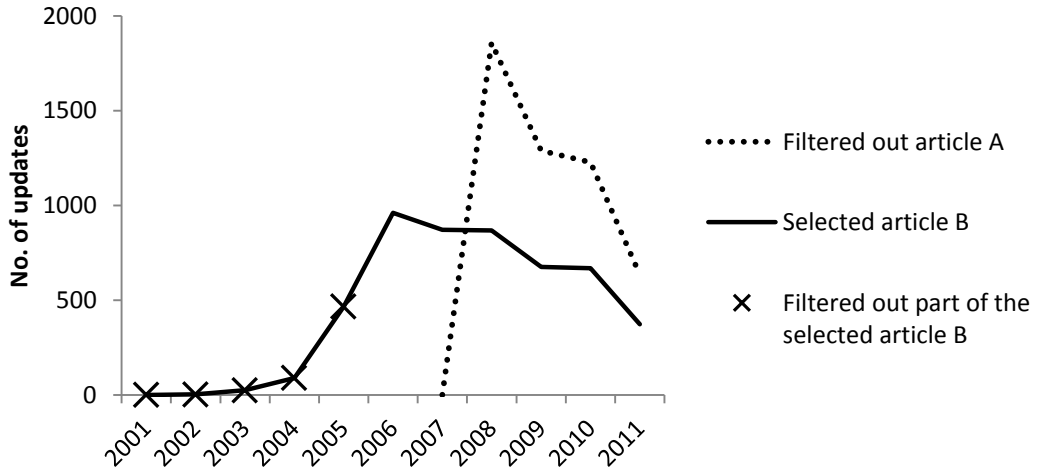


Figure 39. Yearly updates for two sample articles (A and B) of update rate 5000.

[8] http://dumps.wikimedia.org/enwiki/20111201/enwiki-20111201-stub-meta-history.xml.gz
[9] http://download.wikimedia.org/enwiki/20090512/enwiki-20090512-stub-meta-history.xml.gz

Hence, in all the initially selected groups, we filtered out articles with their first revision after 2006-01-01 (like article A in Figure 39)[10]. This allowed us to leave for the analysis the following number of items of their corresponding groups (Table 7): 495 (1000±10), 157, 83, 47, 15, 12, 13, 14, 1 and 3 (7500±150).

Taking into account the empirical granularity limits mentioned before (no more than 2 updates per tick interval, but no less than 1 update per 3 ticks, where "ticks" is a variable parameter taken as 1 day for preliminary data selection purposes), we want to take out of the experiments a period of time when the least updatable group of the selected articles (1000) got revisions less than 1 per 3 days on average during one year. As our preliminary data analysis showed (Figure 40), before 2006-01-01 average number of updates among all the selected articles of update rate 1000 was at most 79 per year, while average yearly revisions for 2006 became about 180 for this group.

Hence, for all the selected articles, we set the starting point to begin accounting the updates as January 01, 2006, when there were near 1 million articles in the Wikipedia (English part).

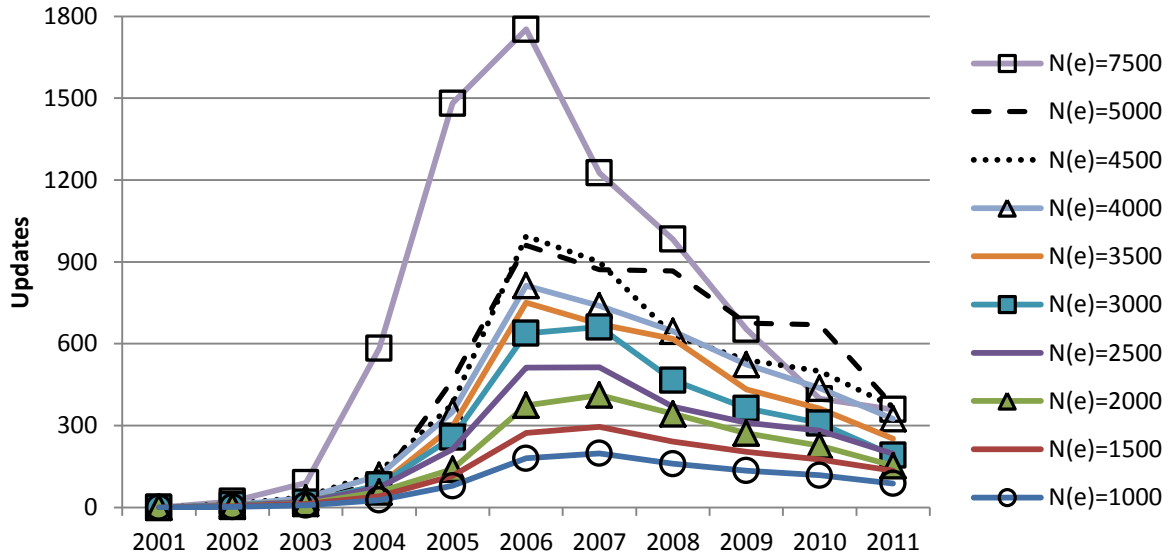


Figure 40. Average yearly updates of selected articles with first revision ≤ 2006-01-01.

For the selected articles, leaving out updates prior to 2006-01-01 lets us having items with average number of revisions decreased by at most 12% of the corresponding groups' size (882 for 1000±10, etc.), except for the control group 7500±150 which got 5371 accountable updates on average.

Hence, we got a period of revisions committed for the selected articles from 2006-01-01 to 2011-12-01, which is about 71 months, or 2160 days, meaning that for the groups of articles 1000 to 7500 revisions we have on average $0.4 \leq \lambda \leq 2.5$ updates per day. This drives us in getting corresponding tick sizes we can explore for the algorithms, coupled with their valid update ranges (see below).

For the given range of frequencies of updates (λ), eq. 5.10 suggests use of tick sizes from about 8 to 49 hours. However, for more extensive studies of prediction accuracy of the algorithms presented in this work, we used an increased range in our experiments – from 6 to 72 hours. More specifically, we used the following values for tick sizes of 6, 12, 24, 48 and 72 hours.

For each tick size, we want to account only *valid* range of updates. As we mentioned before, such validity is driven by the empirical granularity limits. For example, for a tick size 24 hours and lifetime of about 3000 days the valid range of updates would be from 1000 to 6000. Table 8 represents all the tick sizes we used in our work, with the corresponding valid ranges of revisions and the ranges applicable to the groups of selected articles (1000 to 7500).

[10] For real examples of potentially accepted and declined graphs of articles with update rate equals to 1000 see Appendix 4

Table 8. Validity periods for corresponding tick sizes.

Tick size, hours	Valid range of revisions	Valid range of revisions in the selected groups
6	4000 - 24000	4000 - 7500
12	2000 - 12000	2000 - 7500
24	1000 - 6000	1000 - 5000
48	500 - 3000	1000 - 3000
72	350 - 2000	1000 - 2000

7.1 AVERAGING METHOD

Having adopted implementation of the enhanced averaging prediction method (Section 4.1) for an interval prediction (ticks), we tested accuracy of the prediction on the entire dataset (articles with revisions ranging from N(e) = 1000 to N(e) = 7500) with tick sizes from 6 to 72 hours. Leaving only valid ranges of update rates for each tick (Table 8), we got results presented in Figure 41.

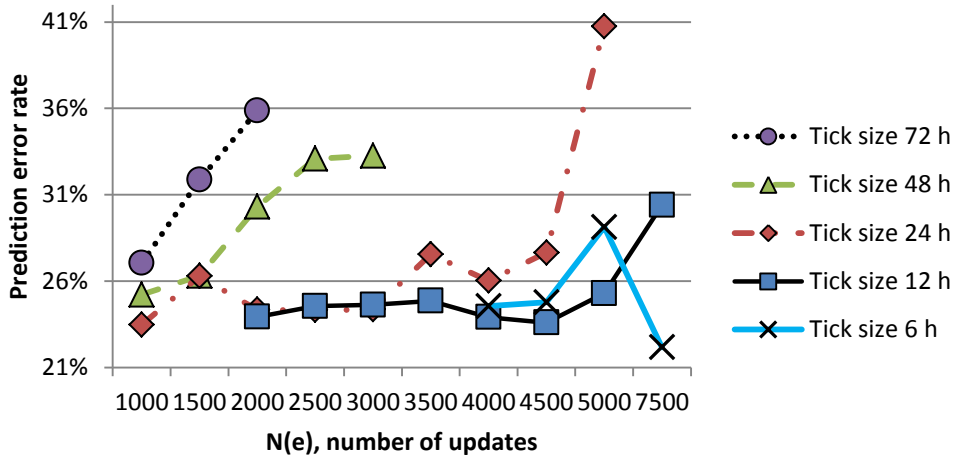


Figure 41. Error rates of enhanced averaging method.

As one can see from the figure, there is a common trend of the averaging method to have more errors as update rate increases, which is valid for almost all the tick sizes we used in our experiments. However, there is a noticeable drop of error rate in prediction of updates for the articles of group N(e) = 7500 while using tick size of 6 hours, compared to the articles of group N(e) = 5000. This can be explained by a finer granularity of time interval used for the prediction, compared to tick size of 12 hours. Nevertheless, this only factor will not explain such an outstanding drop in the graph. We should also consider an increased number of aperiodic updates during the same time interval. The latter factor, in fact, influences prediction accuracy higher as number of updates increases: impact of this factor on the trends of error rates in interval of update groups 2000-4500 is noticeably less than in the groups 5000-7500.

Also, one can see that throughout the entire range of updates, the best accuracy of this method is achievable for tick sizes from 6 to 48 hours. To compare predictive accuracy of this method with the other two, we will take the best accuracy curve which includes error rate at tick size of 24 hours for articles with revisions N(e) = 1000, 48 hours for N(e) = 1500, 12 hours for N(e) = 2000, etc.

7.2 SHIFTING WINDOW

We have explored shifting window method (Section 4.2) over a wide range of values of shifting window parameter, coupled with the set of defined above tick sizes. As the range of values of shifting window parameter we took the following values: 0.5, 1, 2, 3, 6, 12 and 24 months. As we will show later, this range and granularity of step is sufficient enough to find the best prediction performance of this method to compare it with performance of others.

Initially, for each value of shifting window, we obtained prediction error rate while accounting corresponding validity intervals of each tick size (Table 8). Figure 42 represents one of such graphs for window size of 1 month.

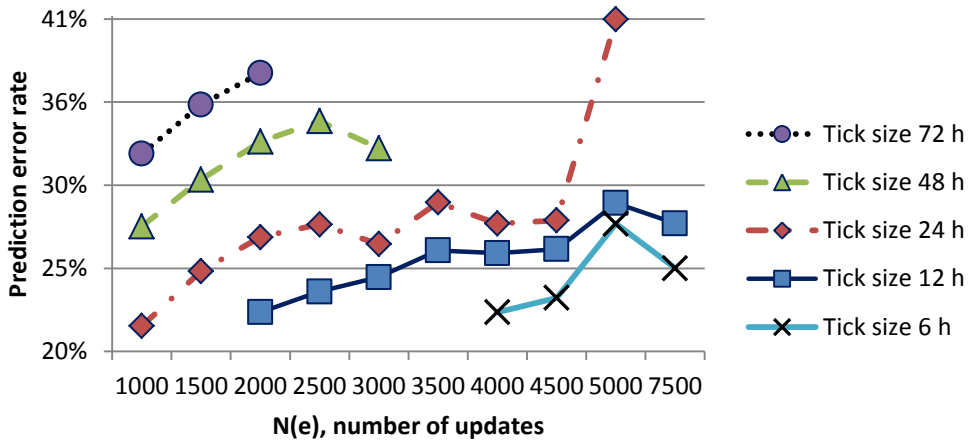


Figure 42. Error rates of shifting window method with window size of 1 month.

For further analysis, we need to see which tick size values give the best prediction results for this method throughout entire range of update rate groups 1000 to 7500. In fact, for all window sizes from 0.5 to 24 months, only ticks from 6 to 24 hours drove this method to the least prediction errors, as on Figure 42. Now, to compare the best accuracy among entire range of shifting window sizes for these 3 tick sizes, we will take each of those ticks and explore behavior of the method in the corresponding update rate groups.

Figure 43 represents error rates for tick size of 6 hours, for all shifting windows from 0.5 to 24 months.

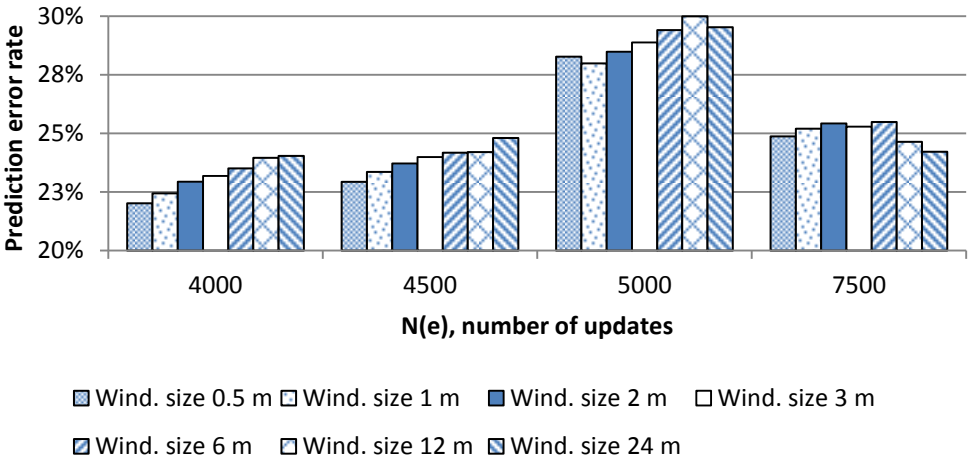


Figure 43. Error rates of shifting window method for tick size of 6 hrs.

As one can see, the best performance of this method for tick size of 6 hours was achieved with shifting window of sizes from 0.5 month (update rate 4000 and 4500) to 1 month (rate 5000) and 24 months (rate 7500).

Similarly, for the other two tick size parameters (12 and 24 hours) error rate is represented on Figure 44 and Figure 45 for the corresponding valid intervals of update rate groups.

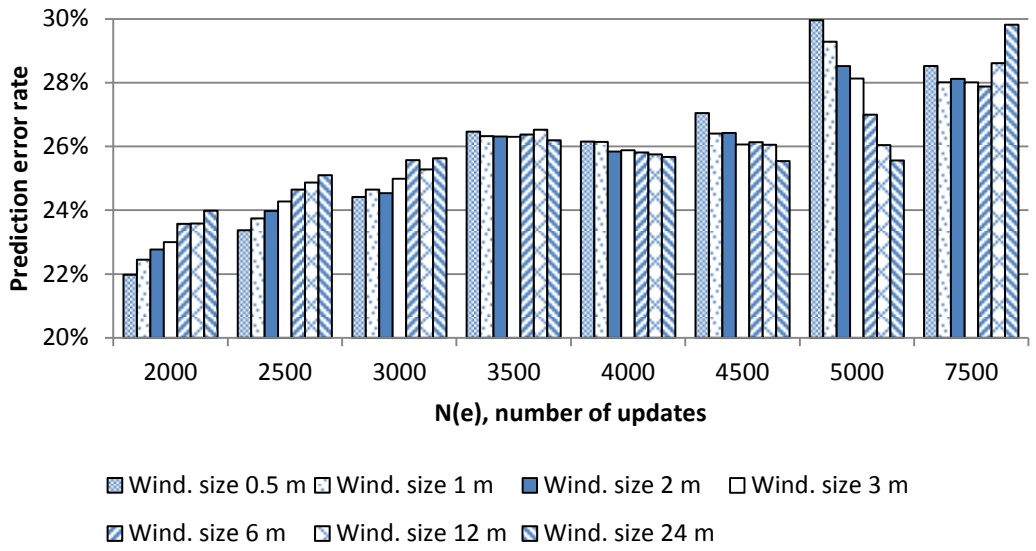


Figure 44. Error rates of shifting window method for tick size of 12 hrs.

For both tick sizes of 12 and 24 hours this method demonstrates the best prediction for window sizes of 0.5, 6 and 24 months.

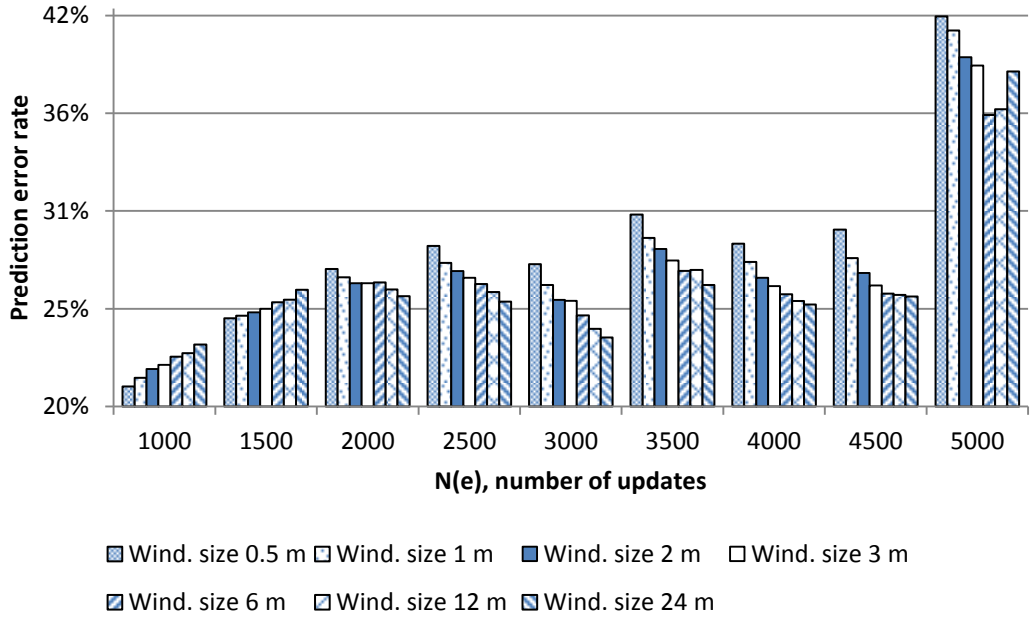


Figure 45. Error rates of shifting window method for tick size of 24 hrs.

We should remind here, that for a group N(e) = 5000 we have only 1 article, which may cause the outlier behavior on Figure 43 and Figure 45. However, since this is the best approximation we could have from the Wikipedia, we accounted this group for our analysis, keeping in mind its nature.

As one can see from the graphs, depending on a tick size and density of updates, this method gives the best results with a wide range of shifting window sizes. However, there is a common trend of better prediction accuracy with window size of 0.5 month for lower update rates (up to N(e) of 4500, 3000, 1500 for tick sizes of 6, 12 and 24 hours correspondingly), and window size of 24 months for higher update rates.

To compare the best prediction performance of this method with the other two, we will take minimal error at each group of update rate, combining graphs for each tick size shown above, from Figure 43 to Figure 45. The results of such a comparison are given in Section 8.

7.3 EXPONENTIAL SMOOTHING

For the exponential smoothing method (Section 4.3) we will first explore a wide range of α parameter (which defines speed of gradually "forgetting" about recent updates). For this experiment, we took α = 0.01, 0.05 and 0.1 to 0.9 with increment of 0.1.

As a result, the best performance of this method was achieved with either some of the selected tick sizes (Figure 46) or all of them (Figure 47).

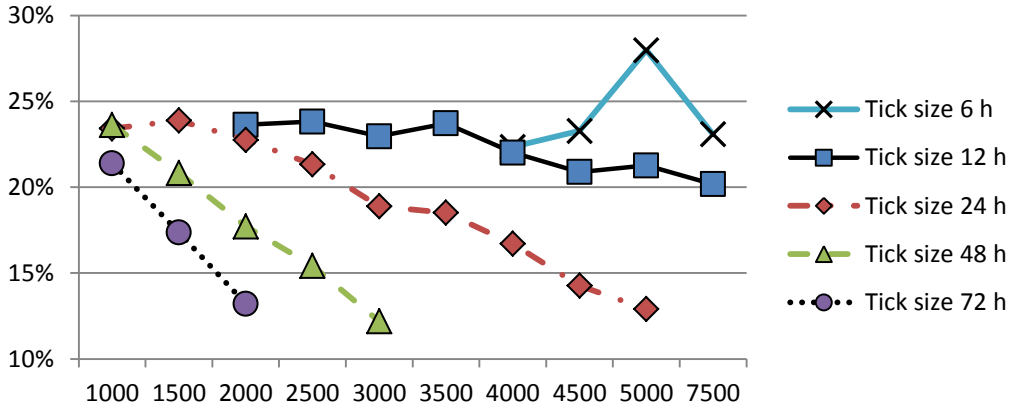


Figure 46. Error rates of exponential smoothing method with parameter α=0.05.

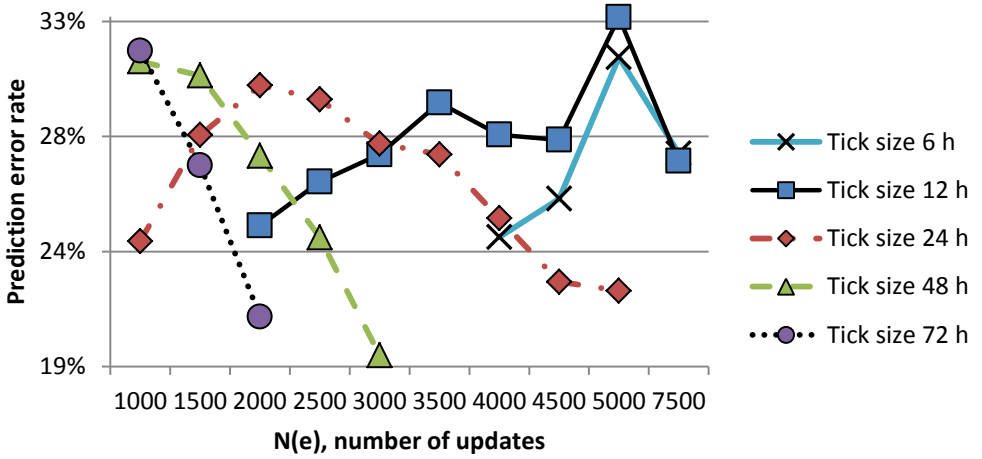


Figure 47. Error rates of exponential smoothing method with parameter α=0.9.

As study of the predictive accuracy of this method for each tick size revealed, only α from 0.01 to 0.5 led to the least error rates. In the following graphs we omitted error rates for α = 0.6 and higher for a better visualization of the results.

On the corresponding valid update rate interval for a tick size of 6 hours exponential smoothing demonstrated the best performance with α = 0.01 and 0.05 (Figure 48).

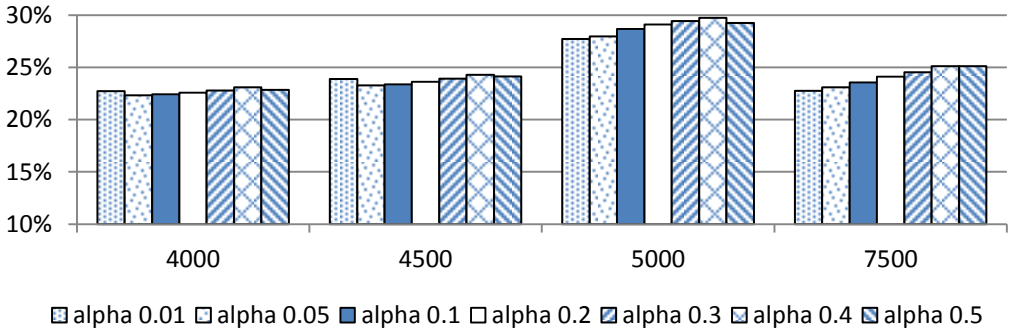


Figure 48. Error rates of exponential smoothing method for tick size of 6 hrs.

For the other tick sizes (from 24 to 72 hours) the best performance of this method was normally achieved for α = 0.01, 0.05 and 0.5 (Figure 49, Figure 50, Figure 51 and Figure 52). For tick size of 24 hours usage of α = 0.1 gave the least errors for group of rate 5000 as well.

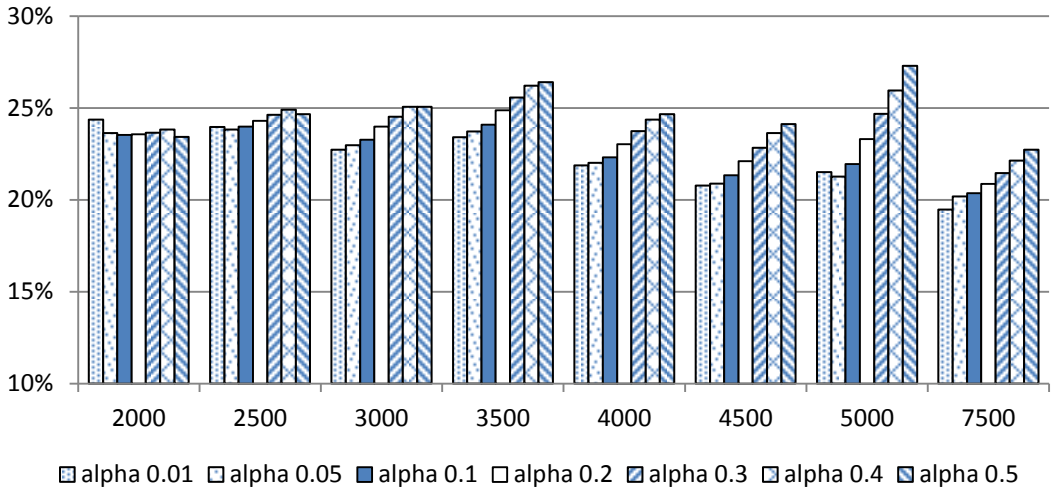


Figure 49. Error rates of exponential smoothing method for tick size of 12 hrs.

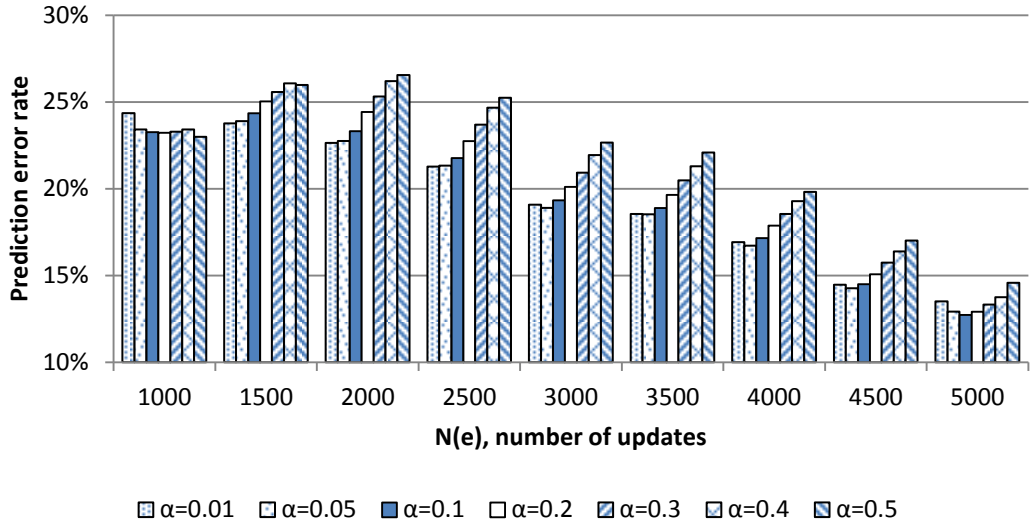


Figure 50. Error rates of exponential smoothing method for tick size of 24 hrs.

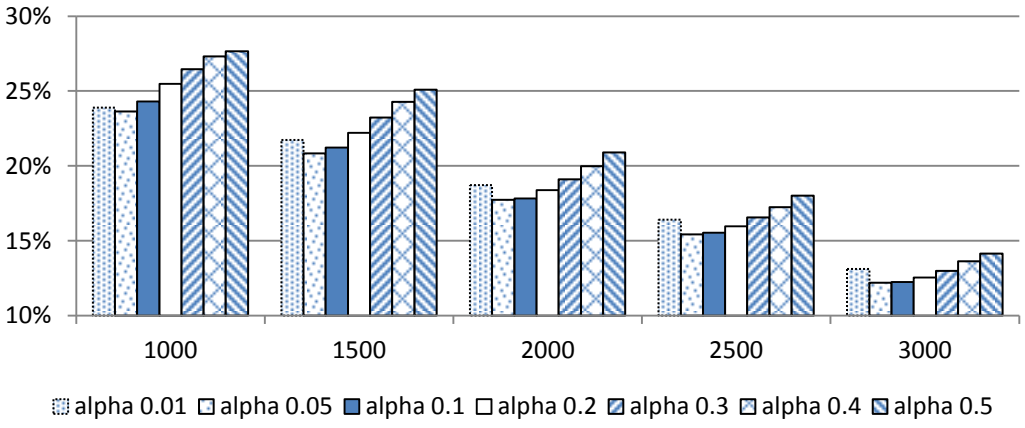


Figure 51. Error rates of exponential smoothing method for tick size of 48 hrs.

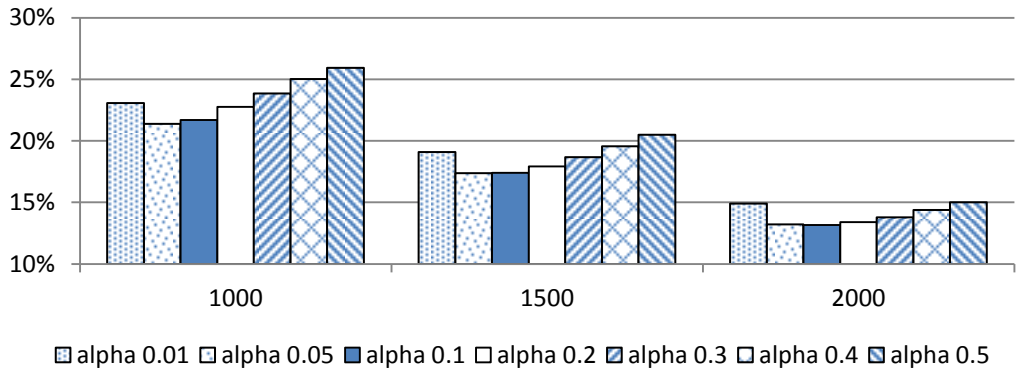


Figure 52. Error rates of exponential smoothing method for tick size of 72 hrs.

Having identified parameters for the best prediction accuracy of exponential smoothing (depending on update rate group and a tick size used), we want to compare the best possible prediction accuracy of this method with the other two, combining the lowest error rate of this method at each update frequency group (Figure 48 to Figure 52).

8 EXPERIMENTAL COMPARISON OF PREDICTIVE ACCURACY OF THE PRESENTED METHODS

To compare predictive accuracy of the algorithms presented in this work, we took the lowest boundary of error rate of each of them, considering entire range of possible values of parameters specific for a corresponding algorithm (tick size, size of shifting window and α coefficient) we used in our experiments. As a result, for each algorithm we got the best predictive accuracy value for every update rate group from 1000 to 7500 (revisions per lifecycle) – see Figure 53. Comparing these results, one can see that throughout the entire range of revision frequencies, except for group N(e) = 1000, exponential smoothing method demonstrated the best predictive accuracy. However, outperformance of enhanced averaging methods over shifting window on some groups of articles (3500, 5000 and 7500) is an interesting observation which may result from exhaustive trend of umber of articles at higher update rate on Wikipedia, and as a result, intolerance for possible specifics of revision history trend of some articles in an update group. However, average error rates by a corresponding prediction methods – enhanced averaging, shifting window and exponential smoothing – were observed as 24.2%, 23.6% and 16.1% correspondingly, which is expected from the design of these algorithms.

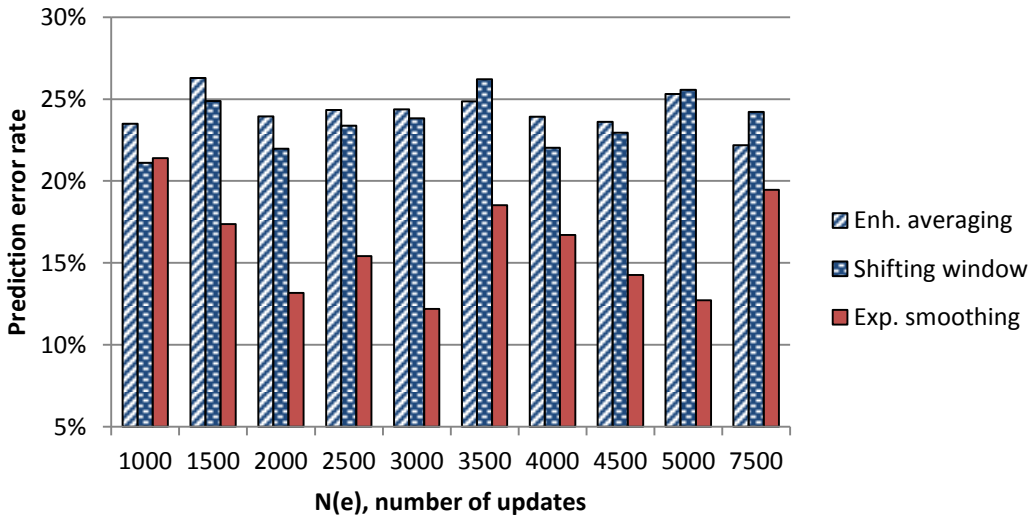


Figure 53. Algorithmic minimal error rates for selected articles.

As one can see, requiring same number of variables for representation of update history of each data element (article from Wikipedia), exponential smoothing method predicts each next update (for N(e)

> 1000 during about 6 years) more accurately than the averaging methods (enhanced averaging and shifting window) adopted for interval prediction. This fact is expected and can be explained by the recency of significant updates for the former method, which allows accounting both global trends and local outliers.

Evaluating efficiency of the presented algorithms according to the metrics proposed in Section 6, we got the following results. For the exponential smoothing method for articles with update rare 1000 we observed predictive correctness $PCorr_{1000} \approx 92.8\%$. For exact prediction methods, namely enhanced average-based and shifting window, we saw prediction outstrip factor (of the same group of articles) as $POF_{1000} \approx 80.8\%$ and $POF_{1000} \approx 46\%$ correspondingly. When those two algorithms are used for interval prediction, we can also get their mean error metrics: $MErr_{1000} \approx 98$ hrs for enhanced average-based vs. $MErr_{1000} \approx 53.6$ hrs for shifting window. Hence, among these two methods, shifting window predicts with higher accuracy but with less "safe" predictions, while enhanced average-based predicts "safer" but less accurately. In a real world application, choice of the prediction methods can mainly be driven either by safety of prediction or by accuracy, depending on given business needs and available resources.

The overall trend of exponential smoothing method to predict updates more accurately among the three approaches is due to its ability to gradually account them; from the other side, this is because of memoryless nature of the two averaging methods – they neither store distribution of previous updates nor learn how they evolve.

9 CONCLUSIONS

In this chapter, we studied one of the most important time-related quality dimension of staleness, starting from demonstration of existing gap in understanding and measurement of such a dimension in current works. After giving a system context, we explored the notion of staleness against major requirements a time-related metric should satisfy. After that, we instantiated the notion of staleness by few statistical approaches to time series analysis (averaging, shifting window and exponential smoothing), comparing their efficiency in our experiments with metadata from Wikipedia.

In our experiments, we also demonstrated predictive efficiency limits of the proposed staleness measurement methods which may further serve as a baseline for development of other staleness measurement methods based on the proposed notion of data staleness.

Chapter 6
Conclusions and Future Work

1 CONCLUSIONS

In this thesis, we presented our studies of a complex and compound notion of data quality which may not be well understood in a research community, and it is definitely underestimated due to its misunderstanding in most IT companies. With our use cases, we tried to provide a vivid demonstration of place of data quality in a real life business intelligence application (Chapter 3) and in a web-oriented information system (0); we also provided deep exploration of one of key time-related quality dimensions – staleness (Chapter 5).

In particular, in Chapter 3 we based on a health care scenario, which in today's world normally lack systemic approach for identification and proper handling of data quality issues, that may come from external data sources, up to stakeholders responsible for strategic decision making. For study of the relevant quality issues, we started with definition of quality-aware reports for a BI environment, and the corresponding consequences of having quality issues from bottom of a data pipeline, up to the reporting level. We broke down a compound notion of data quality into quality dimensions, interrelation and aggregation effects of which we further demonstrated. We proposed an architecture that should effectively address quality aspects highlighted in our scenario.

In 0, we studied a system that was developed within one of FP7 projects (OKKAM). A very important feature of such system for research in data quality is its entire data pipeline transparency: we demonstrated place of data quality from data originators up to data consumers. We identified relevant quality dimensions in each step of the data pipeline, and demonstrated how one can measure quality of data in such systems at different aggregation levels. The corresponding use cases for each of those quality dimensions demonstrated satisfiability of an ultimate goal of data quality – "fitness for customer's use".

In Chapter 5, we studied nature of a data-driven notion of staleness, based on current state of the art and existing requirements for data quality metrics. After setting a proper notion, we described algorithms for data staleness measurement that are suitable for an information system acquiring data with aperiodic updates of high frequency. In particular, we presented an efficient (in terms of memory and CPU resources, and compared to naïve averaging approaches) exponential smoothing method to measure data staleness. Depending on accuracy of required prediction and available resources, one can expand this approach to consider trends, outliers, seasonality and other factors required for a given task.

In our experiments, we have explored prediction accuracy of exponential smoothing method, comparing it with the averaging algorithms presented in this work as well. The results demonstrate the best achievable prediction accuracy for articles of various update frequency and the corresponding parameters required for that. As a consequence, these results can serve as a basis for selection of a proper method for prediction of updates, and hence, measuring time related data-driven characteristic of staleness. In particular, for stochastically changing and frequently updatable web data, represented by revision history of articles from Wikipedia, exponential smoothing method presented in this work gives a basis for development of a staleness measurement algorithm for a custom application. Alternatively, it can serve as a ready-to-go approach.

With data staleness measurement mechanism set, one can either communicate staleness of data elements, or set necessary synchronization techniques with external data sources in such a way that own data would satisfy requirements imposed by a time-related quality dimension identified for an application.

2 FUTURE WORK

Data quality assessment problems have always took their place since appearance of information systems. Price for ignorance of those problems in modern society may cost a noticeable portion of a company turnover. In spite of a solid background and existing data quality assessment approaches, it is a challenge to apply the data quality body of knowledge and available techniques to solve DQ problems in a real world operational environment. The main reasons for that are numerous real world factors influencing organizational data quality. For example, a proper data quality management program even in a mid-size modern Internet company (100-500 people) should account not only internal technical and external user factors, but also social collaboration aspects between programmers, analysts, managers and other persons producing or contributing, processing and consuming data assets. In fact, in such companies, data quality tools and techniques play secondary role, as they only support an entire cross-organizational data quality programs. While leaving the managerial aspects of such programs to the concerned people, from the research point of view we would like to highlight the possible future directions that we foresee for the technological and systematic quality assessment approaches behind those programs, mainly concerned with the problems we studied in this work.

Behind interactivity of the reports in the (BI) Application context (Chapter 3), the solution designer should also account a challenge of real time data presentation in such reports: there are no techniques that will allow to have those reports operating in near real time while processing huge volumes of raw data (that middle size IT companies have nowadays). There are numerous approaches to solve this problem, including efficient data warehouse architectures (like the ones presented by Inmon [114] and Kimball [115]) proposing to pre-process and aggregate data necessary for business needs (i.e., prospective reporting) in advance. The research question would be in approach of aggregation of probabilistic data in order to correctly represent quality issues of base data. Corporate (normally, static) monthly, quarterly and yearly reports raise this question even to a higher level – how to present probabilistic base data at such aggregation level in a static and easy understandable manner?

Besides the factual representation of quality metadata, it is often useful to see its provenance, that is how a certain quality measure is calculated, who and why defined a certain quality rule, etc. Such provenance data visualization is another research challenge one should solve while implementing an application for a BI scenario.

Elaboration of user quality feedback that we outlined in a scope of a BI architecture (Figure 16), constitute another research direction in these lines: how to account user knowledge of beliefs that may help to resolve certain quality issues, but may contradict with beliefs of other users on root causes and resolution processes of the same quality issues.

All of the reporting components outlined in Chapter 3, consider human interaction and consumption of the quality metadata enriched data. It is equally important to consider also automated data analysis and data mining techniques that may benefit from the quality metadata. Most of modern data mining algorithms suppose that data they operate with is not evaluated on its quality aspects, i.e., it is supposed to be complete, consistent, etc. Hence, usually we can get deterministic insights (like classification, clustering, association) instead of probabilistic quality-driven ones. In fact, some data mining techniques stay aside from the deterministically-driven approaches: privacy-preserving data mining [116] aims at data obfuscation by reducing its determinism in the output. However, approaches taken in those techniques cannot be used as is for non-deterministic data (which a quality-aware data can be seen as) in their input.

In spite of the fact that there are works for analysis of uncertain data in OLAP in general [92], as well as detection of outliers in such data [117], data mining algorithms lack pervasive approaches for conventional algorithms to account quality-aware data, as we mentioned. Understanding the need in elaboration of those approaches [118], researches proposed some approaches to data quality mining [119], including association rules mining [120]. However, this directions still lacks robust DQ-aware algorithms for diverse data mining applications which exist for conventional data with no DQ measures.

While in the System part of this work (0) we explored the data quality measurement approach for semi-structured data, we will see a bunch of even more intriguing data quality challenges when dealing with unstructured data in systems like ENS – starting from selection of a proper DQ methodology, to DQ dimensions assessment techniques, possibly uncertain data processing and presentation. Additional (to

those mentioned in this work) uncertainty factors in this case would originate from the nature of unstructured data and the corresponding structure mining techniques.

An approach demonstrated in Chapter 5 towards definition and measurement of one of key time-related data quality dimensions – staleness – lays foundation for further research in this direction. In particular, it would be very interesting to explore data types scope (including the best data type-specific predictable time interval precision coefficients PTI_{PC}) and limitations of the proposed algorithms by testing their prediction accuracy on data of different update nature (e.g., verification if the proposed or similar approaches to assessment of a time-related data characteristic would efficiently work in sensor networks which may have data transfer issues between the nodes).

Equally interesting would be to compare the proposed algorithms with other prediction methods, reasoning not only on their accuracy of prediction, but also on their computational and space efficiency, together with their applicability limits.

BIBLIOGRAPHY

[1] Solomon Negash, "Business intelligence," *Comm. of the Association for Information Systems*, vol. 13, pp. 177-195, 2004.

[2] H.J. Watson and B.H. Wixom, "The Current State of Business Intelligence," *Computer*, vol. 40, no. 9, pp. 96-99, 2007.

[3] Florian Daniel, Fabio Casati, Themis Palpanas, and Oleksiy Chayka, "Managing Data Quality in Business Intelligence Applications," in *VLDB*, Auckland, New Zealand, 2008.

[4] Ron Anderson-Lehman, Hugh J. Watson, Barbara H. Wixom, and Jeffrey A. Hoffer, "Continental Airlines Flies High with Real-time Business Intelligence," *MIS Quarterly Executive*, vol. 3, no. 4, p. 163–176, 2004.

[5] Sue Hay. (2009, January) SAP BusinessObjects Data Quality Management for SAP Solutions XI 3.1. [Online]. http://scn.sap.com/docs/DOC-19787

[6] Paolo Bouquet, Heiko Stoermer, and Barbara Bazzanella, "An Entity Name System (ENS) for the Semantic Web," in *ESWC*, 2008.

[7] Joseph M. Juran, *Juran's quality handbook*, 5th ed.: McGraw-Hill, 1998.

[8] Dominik Luebbers, Udo Grimmer, and Matthias Jarke, "Systematic Development of Data Mining-Based Data Quality Tools," in *VLDB*, 2003.

[9] R.Y. Wang and D.M. Strong, "Beyond accuracy: what data quality means to data consumers," *Journal of Management Information Systems*, vol. 12, no. 4, pp. 5-33, 1996.

[10] Matthew Bovee, Rajendra P. Srivastava, and Brenda Mak, "A conceptual framework and belief-function approach to assessing overall information quality," *International Journal of Intelligent Systems*, vol. 18, no. 1, pp. 51-74, 2003.

[11] Thomas C. Redman, *Data Quality for the Information Age*.: Artech House, 1997.

[12] L. Pipino, Y.W. Lee, and R.Y. Wang, "Data quality assessment," *Comm. ACM*, vol. 45, no. 4, pp. 211-218, 2002.

[13] Y.W. Lee, D.M. Strong, B.K. Kahn, and R.Y. Wang, "AIMQ: a methodology for information quality assessment," *Information and Management*, no. 40, pp. 133-146, 2002.

[14] C. Cappiello, C. Francalanci, and B. Pernici, "Data quality assessment from the user's perspective," in *IQIS*, 2004.

[15] M. Scannapieco and T. Catarci, "Data Quality under the Computer Science perspective," 2002.

[16] Y. Wand and R.Y. Wang, "Anchoring Data Quality Dimensions in Ontological Foundations," *Communications of the ACM*, no. 39, 1996.

[17] M. Jarke, M.A. Jeusfeld, C. Quix, and P. Vassiliadis, "Architecture and quality in data warehouses: An extended repository approach," *Information Systems*, vol. 24, no. 3, 1999.

[18] Larry P. English, *Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits*.: Wiley, 1999.

[19] Felix Naumann, Ulf Leser, and Johann Christoph Freytag, "Quality-driven Integration of Heterogeneous Information Sources," in *VLDB*, 1999.

[20] A. Even and G. Shankaranarayanan, "Utility-driven assessment of data quality," *ACM SIGMIS Database*, vol. 38, no. 2, pp. 75-93, 2007.

[21] Felix Naumann and Johann Christoph Freytag, "Completeness of Information Sources," Technical Report HUB-IB-135 2000.

[22] Carlo Batini and Monica Scannapieco, *Data Quality: Concepts, Methodologies and Techniques*.: Springer, 2006.

[23] Diane M. Strong, Yang W. Lee, and Richard Y. Wang, "Data quality in context," *Comm of the ACM*, vol. 40, no. 5, pp. 103-110, 1997.

[24] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino, "Methodologies for data quality assessment and improvement," *ACM Computing Surveys*, vol. 41, no. 3, 2009.

[25] T. C. Redman, *Data Quality The Field Guide*.: The Digital Press, 2001.

[26] S. Mitra, S. K. Pal, and P. Mitra, "Data mining in soft computing framework: A survey," *Data mining in soft computing framework*, vol. 13, no. 1, pp. 3-14, 2002.

[27] Laure Berti-Équille, "Quality Awareness for Managing and Mining Data," 2007.

[28] Gerhard Joos, "Data Quality Standards," in *XXIII FIG Congress*, Munich, Germany, 2006.

[29] Suparna Bhattacharya et al., "Coordinating backup/recovery and data consistency between database and file systems," in *SIGMOD*, 2002, pp. 500 - 511.

[30] Danette McGilvray, *Executing Data Quality Projects - 10 Steps to Data Quality and Trusted Information*.: Morgan Kaufmann, 2008.

[31] Andrea Domenici, Flavia Donno, Gianni Pucciani, Heinz Stockinger, and Kurt Stockinger, "Replica consistency in a Data Grid," *Nuclear Instruments and Methods in Physics Research*, pp. 24-28, 2004.

[32] R.A. DeKoning and T.L. Langford, "Method of verifying data consistency between local and remote mirrored data storage systems," US6480970, November 12, 2002.

[33] Mark Mosley, *DAMA Dictionary of Data Management*.: Technics Publications, LLC, 2008.

[34] A. Levitin and T. Redman, "Quality dimensions of a conceptual view," *Information Processing & Management*, vol. 31, no. 1, pp. 81-88, 1995.

[35] David Loshin, *The Practitioner's Guide to Data Quality Improvement*.: Morgan Kaufmann, 2010.

[36] Erhard Rahm and Hong Hai Do, "Data Cleaning: Problems and Current Approaches," *IEEE Data Engineering Bulletin*, 2000.

[37] Gao Cong, Wenfei Fan, Floris Geerts, Xibei Jia, and Shuai Ma, "Improving data quality: Consistency and accuracy," in *VLDB*, 2007, pp. 23-28.

[38] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for data cleaning," in *ICDE*, 2007.

[39] Kai-Uwe Sattler, "Data Quality Dimensions," in *Encyclopedia of Database Systems*.: Springer, 2009, pp. 612-615.

[40] Isabelle Comyn-Wattiau et al., "Multidimensional Management and Analysis of Quality Measures for CRM Applications at EDF," in *ICIQ*, 2009.

[41] Richard Y. Wang, "A product perspective on total data quality management," *Comm. of the ACM*, vol. 41, no. 2, 1998.

[42] Carlos Ordonez and Javier García-García, "Referential integrity quality metrics," *Decision Support Systems*, vol. 44, no. 2, p. 495–508, 2008.

[43] A. Segev and W. Fang, "Currency-based updates to distributed materialized views," in *ICDE*, 1990.

[44] Felix Naumann and Claudia Rolker, "Assessment Methods for information quality criteria," in *Proceedings of the International Conference on Information Quality*, 2000.

[45] Mokrane Bouzeghoub and Verónika Peralta, "A framework for analysis of data freshness," in *IQIS*,

2004.

[46] B. Heinrich, M. Klier, and M. Kaiser, "A Procedure to Develop Metrics for Currency and its Application in CRM," *ACM Journal of Data and Information Quality*, vol. 1, no. 1, June 2009.

[47] F. Naumann, U. Leser, and J.C. Freytag, "Quality-driven integration of heterogeneous information systems," in *VLDB*, 1999.

[48] U. Röhm, K. Böhm, H.J. Schek, and H. Schuldt, "FAS: a freshness-sensitive coordination middleware for a cluster of OLAP components," in *VLDB*, 2002.

[49] Hongfei Guo, Per-Åke Larson, Raghu Ramakrishnan, and Jonathan Goldstein, "Relaxed Currency and Consistency: How to Say "Good Enough" in SQL," in *SIGMOD*, 2004.

[50] A. Labrinidis and N. Roussopoulos, "Exploring the tradeoff between performance and data freshness in database-driven web servers," *VLDB Journal*, vol. 13, no. 4, pp. 240-255, 2004.

[51] Huiming Qu and Alexandros Labrinidis, "Preference-Aware Query and Update Scheduling in Web-databases," in *ICDE*, 2007.

[52] P.A. Bernstein, A. Fekete, H. Guo, R. Ramakrishnan, and P. Tamma, "Relaxed Currency Serializability for Middle-Tier Caching and Replication," in *SIGMOD*, 2006.

[53] L. Golab, T. Johnson, and V. Shkapenyuk, "Scheduling updates in a real-time stream warehouse," in *ICDE*, 2009.

[54] Hongfei Guo, Per-Åke Larson, and Raghu Ramakrishnan, "Caching with "Good Enough" Currency, Consistency, and Completeness," in *VLDB*, 2005.

[55] Junghoo Cho and Hector Garcia-Molina, "The evolution of the web and implications for an incremental crawler," in *VLDB*, 2000.

[56] Dimitar Denev, Arturas Mazeika, Marc Spaniol, and Gerhard Weikum, "The SHARC framework for data quality in Web archiving," *VLDB J*, vol. 20, no. 2, pp. 183-207, 2011.

[57] M. Xiong, S. Han, K.-Y. Lam, and D. Chen, "Deferrable Scheduling for Maintaining Real-Time Data Freshness: Algorithms, Analysis, and Results," in *26th IEEE International Real-Time Systems Symposium*, 2005.

[58] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez, "Data Currency in Replicated DHTs," in *SIGMOD*, 2007.

[59] Fuat Akal et al., "Fine-grained replication and scheduling with freshness and correctness guarantees," in *VLDB*, 2005.

[60] Min Xiey, Haixun Wangz, Jian Yinz, and Xiaofeng Meng, "Providing Freshness Guarantees for Outsourced Databases," in *EDBT*, 2008.

[61] D. Ballou, R. Wang, H. Pazer, and G. Tayi, "Modelling Information Manufacturing Systems to Determine Information Product Quality," *Management Science*, vol. 44, no. 4, 1998.

[62] Junghoo Cho and Hector Garcia-Molina, "Synchronizing a database to Improve Freshness," in *SIGMOD*, 2000.

[63] Junghoo Cho and Hector Garcia-Molina, "Effective page refresh policies for Web crawlers," *ACM Transactions on Database Systems (TODS)*, vol. 28, no. 4, December 2003.

[64] C. Cappiello, C. Francalanci, and B. Pernici, "Time-related factors of data quality in multichannel information systems," *Journal of Management Information Systems*, vol. 20, no. 3, 2003.

[65] P. Missier, S. Embury, M. Greenwood, A. Preece, and B. Jin, "Managing Information Quality in e-science: the Qurator work-bench," in *SIGMOD*, 2007.

[66] P. Missier, S. Embury, M. Greenwood, A. Preece, and B. Jin, "Quality views: capturing and exploiting the user perspective on data quality," in *VLDB*, 2006.

[67] J. Rothenberg, "A Discussion of Data Quality for Verification, Validation, and Certification (VV&C) of Data to be Used in Modeling," 1997.

[68] R.L. Leitheiser, "Data Quality in Health Care Data Warehouse Environments," in *HICSS*, 2001.

[69] Wang Chiew Tan, "Provenance in Databases: Past, Current, and Future," *IEEE Data Engineering Bulletin*, vol. 30, no. 4, pp. 3-12, 2007.

[70] P. Buneman and S. Khanna, "On Propagation of Deletions and Annotations through Views," in *PODS*, 2002.

[71] T.J. Green, G. Karvounaraki, and V. Tannen, "Provenance Semirings," in *PODS*, 2007.

[72] P. Missier, S. Embury, and R. Stapenhurst, "Exploiting provenance to make sense of automated data acceptance decisions in scientific workflows," in *IPAW*, 2008.

[73] D. Bhagwat, L. Chiticariu, C. W. Tan, and G. Vijayvargiya, "An annotation management system for relational databases," *VLDBJ*, vol. 14, no. 4, pp. 373-396, 2005.

[74] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1-16, 2007.

[75] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita, "Declarative Data Cleaning: Language, Model, and Algorithms," in *VLDB*, 2001, pp. 371-380.

[76] E. Lim, J. Srivastava, S. Prabhakar, and J. Richardson, "Entity Identification in Database Integration," in *ICDE*, 1993, pp. 294-301.

[77] S. Guha, N. Koudas, A. Marathe, and D. Srivastava, "Merging the Results of Approximate Match Operations," in *VLDB*, 2004, pp. 636-647.

[78] M. Jarke and Y. Vassiliou, "Data warehouse quality: a review of the DWQ project," in *IQ*, 1997, pp. 299-313.

[79] P. Vassiliadis, M. Bouzeghoub, and C. Quix, "Towards Quality-oriented Data Warehouse Usage and Evolution," *Information Systems*, vol. 25, no. 2, pp. 89-115, 2000.

[80] D. Theodoratos and M. Bouzeghoub, "Data Currency Quality Factors in Data Warehouse Design," in *DMDW*, 1999.

[81] L. Antova, C. Koch, and D. Olteanu, "10^10^6 Worlds and Beyond: Efficient Representation and Processing of Incomplete Information," in *ICDE*, 2007, pp. 606-615.

[82] O. Benjelloun, A.D. Sarma, A.Y. Halevy, M. Theobald, and J. Widom, "Databases with uncertainty and lineage," *VLDBJ*, vol. 17, no. 2, p. 243–264, 2008.

[83] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," *VLDBJ*, vol. 16, pp. 523-544, 2007.

[84] S. Singh, C. Mayfield, R. Shah, S. Prabhakar, and S. E. Hambrusch, "Database Support for Probabilistic Attributes and Tuples," in *ICDE*, 2008, pp. 1053-1061.

[85] A. Motro, "Management of uncertainty in database system," in *Modern database system: The object model, interoperability*.: Addison-Wesley, 1995.

[86] F. Naumann, J. C. Freytag, and U. Leser, "Completeness of integrated information sources," *Information Systems*, vol. 29, pp. 583-615, 2004.

[87] M. Scannapieco and C. Batini, "Completeness in the Relational Model: a Comprehensive Framework," in *IQ*, 2004, pp. 333-345.

[88] S. Sarawagi, "Special Issue on Data Cleaning," *Bulletin of the IEEE Technical Committee on Data Engineering*, vol. 23, no. 4, 2000.

[89] T. Palpanas, N. Koudas, and A. Mendelzon, "Using Datacube Aggregates for Approximate

Querying and Deviation Detection," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 11, pp. 1465-1477, 2005.

[90] Themis Palpanas and Nick Koudas, "Entropy Based Approximate Querying and Exploration of Datacubes," in *SSDBM* , 2001, pp. 81-90.

[91] D. Srivastava and Y. Velegrakis, "Intensional associations between data and metadata," in *SIGMOD*, 2007, pp. 401-412.

[92] Doug Burdick, Prasad M. Deshpande, T.S. Jayram, Raghu Ramakrishnan, and Shivakumar Vaithyanathan, "OLAP over uncertain and imprecise data," *VLDB Journal*, pp. 123-144, 2007.

[93] Nicolas Bonvin, Paolo Bouquet, Michele Catasta, Daniele Cordioli, Peter Fankhauser, Julien Gaugaz, Ekaterini Ioannou, Hristo Koshutanski, Antonio Maña, Claudia Niederée, Themis Palpanas, Heiko Stoermer Zoltán Miklós, "From web data to entities and back," *Advanced Information Systems Engineering*, pp. 302--316, 2010.

[94] T. Palpanas, J. Chaudhry, P. Andritsos, and Y. Velegrakis, "Entity data management in Okkam," in *Database and Expert Systems Application*, 2008, pp. 729--733.

[95] G. Giannakopoulos and T. Palpanas, "Adaptivity in entity subscription services," in *Proceedings of the 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, 2009, pp. 61-66.

[96] G. Giannakopoulos and T. Palpanas, "Content and type as orthogonal modeling features: a study on user interest awareness in entity subscription services," *International Journal On Advances in Networks and Services*, vol. 3, no. 1, pp. 296--309, 2010.

[97] Barbara Bazzanella, Heiko Stoermer, and Paolo Bouquet, "Top Level Categories and Attributes for Entity Representation," University of Trento, Technical Report 2008.

[98] Kuang Chen, Harr Chen, Neil Conway, Joseph M. Hellerstein, and Tapan S. Parikh, "USHER: Improving Data Quality with Dynamic Forms," in *ICDE*, 2010.

[99] Arkady Maydanchik, *Data Quality Assessment*.: Technics Publications, LLC, 2007.

[100] W3C I18N FAQ: Date formats. [Online]. http://www.w3.org/International/questions/qa-date-format

[101] J. Klensin. (2004, February) Network Working Group. [Online]. http://tools.ietf.org/html/rfc3696

[102] P. Mockapetris. (1987) Network Working Group. [Online]. http://tools.ietf.org/html/rfc1035

[103] R. Braden. (1989) Network Working Group. [Online]. http://tools.ietf.org/html/rfc1123

[104] R. Elz and R. Bush. Network Working Group. [Online]. http://tools.ietf.org/html/rfc2181

[105] Internet Assigned Numbers Authority (IANA). [Online]. http://data.iana.org/TLD/tlds-alpha-by-domain.txt

[106] (2010, Jan.) E.123, ITU-T Recommendation "Notation for national and international telephone numbers, e-mail addresses and Web addresses".

[107] W3C. Qualified Names. [Online]. http://www.w3.org/TR/REC-xml-names/#NT-QName

[108] Oleksiy Chayka, Themis Palpanas, and Paolo Bouquet, "Defining and Measuring Data-Driven Quality Dimension of Staleness," University of Trento, TR DISI-12-016, 2012.

[109] Paolo Bouquet, Themis Palpanas, Heiko Stoermer, and Massimiliano Vignolo, "A Conceptual Model for a Web-scale Entity Name System," in *ASWC*, Shanghai, China, 2009.

[110] Barbara Bazzanella, Themis Palpanas, and Heiko Stoermer, "Towards a General Entity Representation Model," in *Information Reuse & Integration*, 2009.

[111] *Collins English Dictionary*, 8th ed., 2006.

[112] Maria Kontaki, Apostolos N. Papadopoulos, and Yannis Manolopoulos, "Adaptive similarity search in streaming time series with sliding windows," *Data & Knowledge Engineering*, vol. 63, p. 478–502, 2007.

[113] Everette Jr. Gardner, "Exponential smoothing: The state of the art — Part II," *International Journal of Forecasting*, no. 22, p. 637– 666, 2006.

[114] W. H. Inmon, Claudia Imhoff, and Ryan Sousa, *The Corporate Information Factory*, 2nd ed.: Wiley, 2000.

[115] Ralph Kimball and Margy Ross, *The Data Warehouse Toolkit*, 2nd ed.: Wiley, 2002.

[116] Vassilios S. Verykios et al., "State-of-the-Art in Privacy Preserving Data Mining," *ACM SIGMOD Record*, vol. 3, no. 1, pp. 50-57, 2004.

[117] C. C. Aggarwal and P. S. Yu, "Outlier Detection with Uncertain Data," in *SDM*, 2008, pp. 483-493.

[118] Roger H. Blake and Paul Mangiameli, "The effects and interactions of data quality and problem complexity on data mining," in *ICIQ*, 2008.

[119] J. Hipp, U. Guntzer, and U. Grimmer, "Data quality mining - making a virtue of necessity," in *SIGMOD DMKD Workshop*, 2001.

[120] Laure Berti-Équille, "Data quality awareness: a case study for cost optimal association rule mining," *Knowledge and Information Systems*, vol. 11, no. 2, pp. 191-215, 2007.

[121] R.Y. Wang, M.P. Reddy, and H.B. Kon, "Toward quality data: an attribute-based approach," *Decision Support Systems*, vol. 13, 1995.

[122] Guzmán Llambías, Regina Motz, Federico Toledo, and Simon de Uvarow, "Learning to Get the Value of Quality from Web Data," *Lecture Notes in Computer Science*, vol. 5333, no. 2008, 2008.

[123] H. Hinrichs, "Datenqualitätsmanagement in Data Warehouse-Systemen," Oldenburg, doctoral thesis 2002.

[124] B. Heinrich, M. Kaiser, and M. Klier, "How to measure data quality? – a metric based approach," in *International Conference on Information Systems (ICIS*, 2007.

[125] B. Heinrich and M. Klier, "Assessing data currency — a probabilistic approach," *Journal of Information Science*, vol. 37, no. 1, 2011.

[126] G. Shankaranarayan, M. Ziad, and R.Y. Wang, "Managing data quality in dynamic decision environments: An information product approach," *Journal of Database Management*, vol. 14, no. 4, 2003.

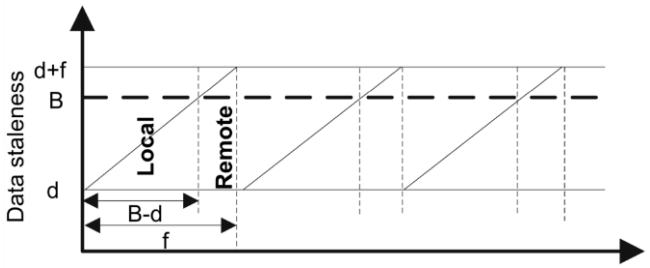
[127] Barbara Pernici and Monica Scannapieco, "Data Quality in Web Information Systems," *Journal on Data Semantics*, pp. 48-68, 2003.

APPENDICES

Appendix 1. TIME-RELATED NOTIONS, DEFINITIONS AND MEASUREMENT METHODS FROM THE RELATED RESEARCH WORKS

Notion, scope (System or/and Data)	Definition/measurement
Freshness (up-to-dateness) (S)	State of correspondence of data in replica (web cache) to data in its source [55]
	"freshness represents the fraction of up-to-date pages in the local collection"
Freshness (staleness, lateness, tardiness) (S)	"*freshness* of a data stream is the maximum timestamp of any of its tuples that have arrived at the warehouse by time t." "*data staleness* – the difference between the current time and the timestamp of the most recent tuple in a table." "(system) *lateness* (or *tardiness*) – the difference between the completion times of late tasks and their deadlines" [53]
	$S_T = t - freshness(T, t)$ where $freshness(T, t)$ is the maximum timestamp of any tuple in table *T* at time *t*
Freshness (up-to-dateness) (S)	Freshness (up-to-dateness) is state of correspondence of data in replica to data in its source table [52], [59]
	N/A: authors in [52] present a method of usage of existing freshness constraints in user queries; in [59] freshness requirements are specified by users
Freshness (up-to-dateness) (S)	Freshness: "the outsourced data is up-to-date, that is, all database update operations have been correctly carried out by the service provider" [60]
	N/A: authors presents a method to ensure correctness of update operations made by outsources database
Freshness (staleness, up-to-dateness) (S)	Availability of updates for replicated web database makes its fresh data stale (or out-of-date), until completion of application of corresponding updates [51]
	"staleness can be measured by the number of unapplied updates, as well as the time differential or value distance between the current and the most up-to-date data items"
Freshness (staleness) (S)	"(a data object) is stale when an update for it has arrived, but not yet executed" [50]
	"AWebView (a fragment of a web page) W_i is stale, if W_i is materialized and has been invalidated, or if W_i is not materialized and there exists a pending update for a parent relation of W_i. A WebView W_i is fresh, otherwise." $f(W_i, t_k) = \begin{cases} 1, & if\ W_i\ is\ fresh\ at\ time\ t_k \\ 0, & if\ W_i\ is\ stale\ at\ time\ t_k \end{cases}$
Freshness (temporal validity) (S)	"A real-time data object is *fresh* (or *temporally valid*) if its value truly reflects the current status of the corresponding entity in the system environment" [57]
	N/A: given data validity intervals (which are "defined based on the dynamic properties of the data object" and they are also "application-dependent"), authors demonstrate a deferrable algorithm for processor scheduling updates from sensors.

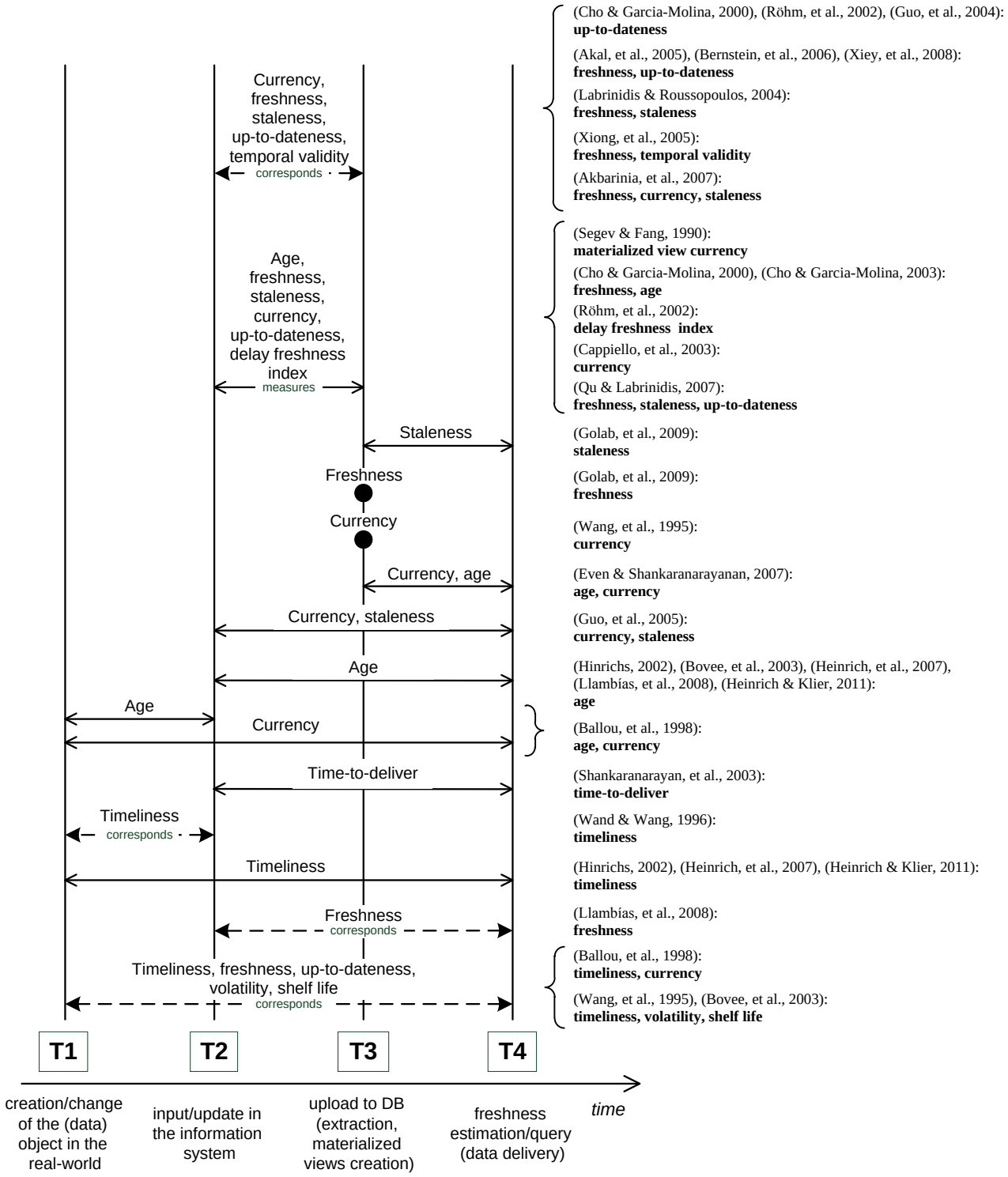
Freshness index (up-to-dateness) (S)	"reflects how much the data has deviated from the up-to-date version" [48]
	N/A: "freshness index reflects how much the data has deviated from the up-to-date version. Intuitively, a freshness index of 1 means that the data is up-to-date, while an index of 0 tells us that the data is 'infinitely' outdated"
Delay freshness index (S)	"reflects how late a certain cluster node is as compared to the up-to-date OLTP node" [48]
	$$f(c) = \frac{\tau(c)}{\tau(c_0)}$$ where $\tau(c)$ is commit time of last propagated update transaction on OLAP node c, $\tau(c_0)$ is commit time of the most recent update transaction on OLTP node
Freshness (age, up-to-dateness) (S)	*Freshness, age, up-to-dateness*: state of correspondence of data in replica to data in its source (real-world counterparts). *Freshness* is fraction of local database that is up-to-date [62], [63]
	Freshness of a database S at time t: $F(S;t) = \frac{M}{N}$ where M is up-to-date elements, and N is total number of element in the database. Freshness of a data element e_i at time t: $F(e_i,t) = \begin{cases} 1, & if\ e_i\ is\ fresh\ at\ time\ t \\ 0, & otherwise \end{cases}$ Age of a data element e_i at time t: $$A(e_i,t) = \begin{cases} 0, & if\ e_i\ is\ fresh\ at\ time\ t \\ t - modification\ time\ of\ e_i, & otherwise \end{cases}$$
Volatility/ shelf life (D)	"How long the item remains valid" [61], [121]
	N/A: "a quality parameter value [e.g., volatility] is the value determined by the user"
Web data Freshness (age, volatility) (S)	"*freshness* measures how much updated are data for a specific task" [122]
	Freshness = max{0, 1-(age/volatility)} "*age* suggests how old are data, captures the time interval between the creation or actualization of data and the time at what user receives data; *volatility* measures the frequency with which data change over time"
Age (D)	"the time difference between when the real-world event occurred and when the data was entered" [61]
	Age is measured as per its definition (see above).
Age (D)	"*Age* measures how long ago information was recorded" [10]
	Age is the time difference between when data was recorded and instant of measurement
Currency (age) (D/S)	"refers to the age of the primitive data units used to produce the information products" [61]
	"The currency measure is a function of several factors: when the information product is delivered to the customer (Delivery Time); when the data unit is obtained (Input Time); and how old the data unit is when received (Age)." Currency = (Delivery Time - Input Time) + Age (Delivery Time - Input Time) represents how long data have been in a system Age is the time difference between when a real-world event occurred and when the data was entered

Materialized view Currency (S)	Currency of materialized view is a degree of its correspondence to its base table [43]
	$$T_v^{(t_i)} = t_i - \max_{t \le t_i}\{t | State_v(t_i) = State_B(t)\}$$ $T_v^{(t_i)}$ is view currency at time t_i, $State_B(t)$ is base table state and $State_v(t_i)$ is state of materialized view at time t_i. Hence, this metric measures the time between data delivery (to a user) and data update (in materialized view from its base table or another view).
Currency level (S)	"Currency level is defined as the degree to which data are up-to-date in a given operational database" [64]
	$$currency\_level_{ij} = 1 - out\_of\_date_{ij}$$ is currency level of i-th functionality of j-th channel [in multichannel information system]
Currency (D)	"when the data item was stored in the database" [121]
	N/A: "a quality parameter value [e.g., currency] is the value determined by the user"
Currency (age) (S)	"Currency measures the degree to which the data is recent and up to date." More specifically, currency is: "The age of data items – the time lag between last update and present time" [*Data Characteristics Observed*]; "The extent to which the data items in the dataset are recent" [*Impartial Interpretation*]; "The extent to which outdated data damages utility" [*Contextual Interpretation*] [20]
	Currency measurement reflects *age*, the time lag between present time and instant of last update of data item.
Currency (staleness, up-to-dateness) (S)	State of correspondence of data in replica to its data in source table [49]
	Currency, staleness: linearly growing function with lower limit as *d*, update propagation delay, and upper limit as *d+f*, where *f* is update propagation interval. Authors show how boundary on currency *B* (given by user) one can include into SQL queries, enforcing execution of queries on local replica or remote source table. Figure below "Synch cycle and data currency" from the paper demonstrates this: 
Currency (up-to-dateness) (S)	State of correspondence of data in replica to its data in source table [54]
	Currency of a copy of database snapshot is measured by its staleness value, i.e., by the time elapsed from instant when source got update(s) which has not been propagated to copy instance, to instant of measurement.
Currency (freshness, staleness) (S)	Notions of *currency*, *freshness*, *staleness* relate to a state of correspondence of data in replica to data in its source; *current* replica is such replica with latest updates [58]
	N/A: scope of the paper is getting replicas with the latest updates rather than measuring their currency or freshness metrics

Timeliness (age) (D)	"The metric for timeliness shall deliver an indication (not a verified statement under certainty) whether an attribute value has changed in the real world since its acquisition and storage within the system or not." [123], [124], [125]
	$$Timeliness = \frac{1}{(mean\ attribute\ update\ time) \cdot (attribute\ age) + 1}$$ "This quotient serves as a metric, which quantifies if the current attribute value is outdated." "…if the *mean attribute update time* is 0 (i.e. the attribute value never becomes out of date), timeliness is 1 (attribute value is up-to-date). If on the other hand *attribute age* is 0 (i.e. the attribute value is acquired at the instant of quantifying DQ) we get the same result. For higher values of *mean attribute update time* or *attribute age* the result of the metric approaches 0. I.e., that the (positive) indication (the attribute value is still corresponding to its real world counterpart) decreases." [123] $$Q_{Time}(w, A) := \exp(-decline(A) \cdot age(w, A))$$ "$Q_{Time}(w, A)$ denotes the probability that the attribute value is still valid." [124], [125] *age*(*w*, *A*) denotes the age of the attribute [A] with value *w*, which is computed by means of two factors: the instant when DQ is quantified and the instant of data acquisition; *decline*(*A*) of attribute *A*'s values can be determined statistically
Timeliness (currency, volatility) (D/S)	"… timeliness of an information product is dependent upon when the information product is delivered to the customer" [61]
	Timeliness = $\{\max[(1\text{-currency/volatility}), 0]\}^S$ Exponent *s* is a parameter that allows us to control sensitivity of timeliness to currency-volatility ratio. Timeliness ranges from 0 ("the data is unacceptable from the timelines viewpoint") to 1 ("the data meets the most string timeliness standard").
Timeliness (currency, volatility) (D/S)	"can be characterized by *currency* (when the data item was stored in the database) and *volatility* (how long the item remains valid)" [121]
	N/A
Timeliness (relevancy, availability) (D/S)	"An additional [*to the notion of timeliness in* [121]] meaning of timeliness is whether information, relevant or not, was available in time to be useful" [10]
	N/A
Timeliness (D)	"In our model, timelines refers only to the delay between a change of the real-world state and the resulting modification of the information system state" [16]
	N/A: authors provide ontological foundations to further measure data quality dimensions
Time-to-deliver (S)	"The time-to-deliver an IP [information product] (or any component data) is defined as the time to completely generate the IP from any processing stage in the IPMAP [a set of modeling constructs to systematically represent the manufacture of an IP]" [126]
	Time-to-Delivery = $\sum_{i=1,n} t_i$ (a.1) Expected mean time at stage x $(ET_x) = (4 \cdot m_x + a_x + b_x)/6$ (a.2) Variance in time at stage x $(\sigma^2_{<x>}) = [(b_x - a_x)^2]/36$ (a.3) Probability (completing stage x) = $(ET_x - T)/(\sum \sigma_{<Critical\ Path\ to\ x>})$ (a.4) where m_x is mean time at stage x; a_x, b_x, are optimistic and pessimistic time estimates at stage *x*. These time estimates are assumed to follow Beta distribution. Expected mean time and variance at stage *x* are computed based on Beta distribution using equations a.2 and a.3. Normalized probability is specified by equation a.4.

Timeliness (age) Currency (up-to-dateness) (D/S)	Timeliness: “the extent to which the age of the data is appropriate for the task at hand.” Currency: "it is easy to tell if the data are updated" [9]
	N/A for notions of timeliness, age and currency; authors provide a list of data quality notions based on customer surveys.
Timeliness (S)	“extent to which the age of the data is appropriate for the task at hand” [47]
	N/A: timeliness is measured based on update information that is provided by information source

Appendix 2. KEY TIME-RELATED DATA QUALITY NOTIONS FROM THE RELATED RESEARCH WORKS



Appendix 3. DEFINITION OF TIME-DEPENDENT QUALITY DIMENSION OF COMPLETENESS

Authors in [127] proposed time-aware semantics of notion of completeness in the Web Information Systems which are characterized by heterogeneous data that continuously evolves and is usually a result of changeable data producing processes. They show data model and corresponding framework aiming not only at static data quality assessment, but also at dynamic association of expected data quality evolution regarding subsequent updates. Considering these aspects, authors have identified *completeness* as such a dimension that gives information about current degree of completeness, and *completability* as a dimension that shows how completeness measure will evolve over time.

Completability can be described through a function *C(t)* which aimed at forecasting of this metric in time interval from *t_curr* (instant at which metric is evaluated) to *t_max* (estimated instant after which measurements will not take place or will be unimportant). The corresponding forecasted metric can be then measured as $C = \int_{t\_curr}^{t\_max} C(t)dt$.

Suitable completability function *C(t)* for certain application context gives proper basis for estimation of completability metric at current time: in Figure 54 one can see a filled-in rectangular region *A* that is defined as $A = (t\_max - t\_curr) * \frac{c\_max - c\_pub}{2}$ where *c_max* corresponds to completability at instant *t_max*, and *c_pub* corresponds to instant of measurable data publishing. It can be also possible to evaluate degree of completability with respect to further evolvement at each moment of time *t_curr* as "high", ($C > A$); "medium", ($C = A$) or "low"($C < A$).

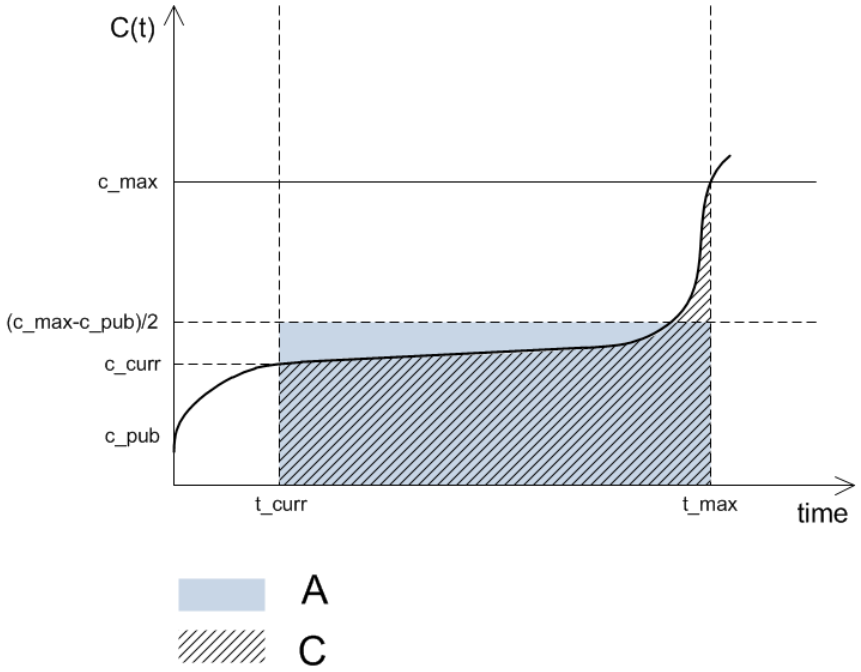


Figure 54. A graphical representation of completability, adapted from [127].

Appendix 4. DISTRIBUTION OF UPDATES OF ARTICLES FROM WIKIPEDIA WITH UPDATE RATE 1000

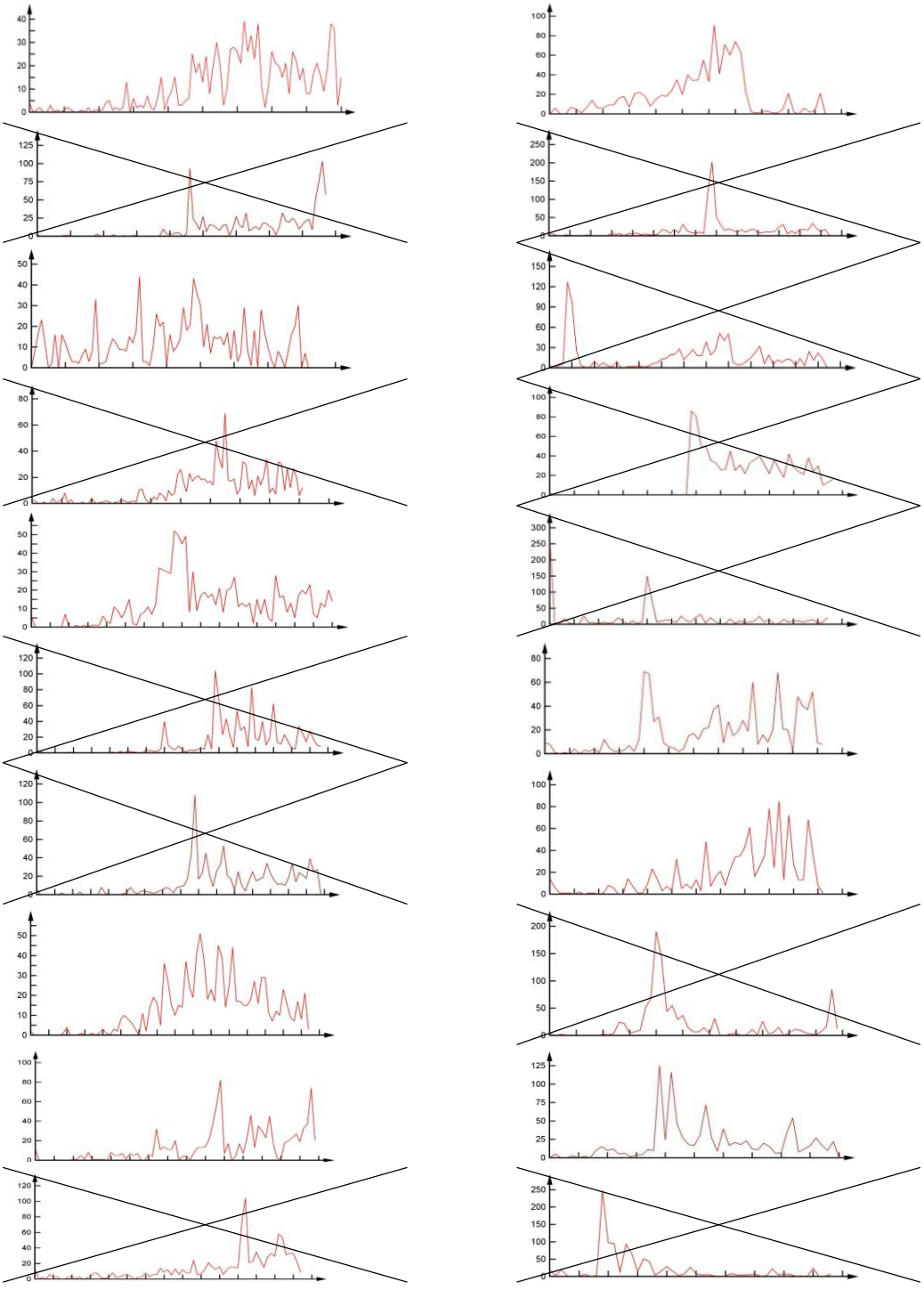


Figure 55. Distribution of updates of articles from Wikipedia with update rate λ=1000 (crossed out graphs are samples of filtered out articles for our experiments)