Problems, Methods, and Challenges in Comprehensive Data Cleansing

Heiko Müller, Johann-Christoph Freytag

Humboldt-Universität zu Berlin zu Berlin,
10099 Berlin, Germany

{hmueller, freytag}@dbis.informatik.hu-berlin.de

Abstract

Cleansing data from impurities is an integral part of data processing and maintenance. This has lead to the development of a broad range of methods intending to enhance the accuracy and thereby the usability of existing data. This paper presents a survey of data cleansing problems, approaches, and methods. We classify the various types of anomalies occurring in data that have to be eliminated, and we define a set of quality criteria that comprehensively cleansed data has to accomplish. Based on this classification we evaluate and compare existing approaches for data cleansing with respect to the types of anomalies handled and eliminated by them. We also describe in general the different steps in data cleansing and specify the methods used within the cleansing process and give an outlook to research directions that complement the existing systems.

Contents

1 Introduction

The application and exploitation of huge amounts of data takes an ever increasing role in modern economy, government, and research. Anomalies and impurities in data cause irritations and avert its effective utilization, disabling high performance processing and confirmation of the results and conclusions gained by data interpretation and analysis. This leads to higher cost and less benefit for data processing, possibly making its application worthless for the consumer. In [Eng00] several problems caused by data anomalies are listed, among others, that incorrect price data in retail databases costs American consumers $2.5 billion in annual overcharges. In this context, data cleansing has received a strong and growing interest over the years. We rely on error-free data and the ability to deploy it to our needs. This deployment includes the efficient integration, application, interpretation, and analysis of data to support business processes, administrative tasks, decision making, and scientific reasoning.

The existence of anomalies and impurities in real-world data is well known. In [Orr98, Red98] their typical rates are estimated to be at 5%. This has lead to the development of a broad range of methods intending to identify and eliminate them in existing data. We subsume all these under the term data cleansing; other names are data cleaning, scrubbing, or reconciliation. There is no common description about the objectives and extend of comprehensive data cleansing. Data cleansing is applied with varying comprehension and demands in the different areas of data processing and maintenance. The original aim of data cleansing was to eliminate duplicates in a data collection, a problem occurring already in single database applications and gets worse when integrating data from different sources. Data cleansing is therefore often regarded as integral part of the data integration process. Besides elimination of duplicates, the integration process contains the transformation of data into a form desired by the intended application and the enforcement of domain dependent constraints on the data.

Usually the process of data cleansing cannot be performed without the involvement of a domain expert, because the detection and correction of anomalies requires detailed domain knowledge. Data cleansing is therefore described as semi-automatic but it should be as automatic as possible because of the large amount of data that usually is be processed and because of the time required for an expert to cleanse it manually. The ability for comprehensive and successful data cleansing is limited by the available knowledge and information necessary to detect and correct anomalies in data.

Data cleansing is a term without a clear or settled definition. The reason is that data cleansing targets errors in data, where the definition of what is an error and what not is highly application specific. Consequently, many methods concern only a small part of a comprehensive data cleansing process using highly domain specific heuristics and algorithms. This hinders transfer and reuse of the findings for other sources and domains, and considerably complicates their comparison

This paper presents a survey of data cleansing approaches and methods. Starting from a motivation of data cleansing, existing errors in data are classified and a set of criteria is defined that comprehensive data cleansing has to address. This enables the evaluation and comparison of existing approaches for data cleansing regarding the types of errors handled and eliminated by them. Comparability is achieved by the classification of errors in data. Existing approaches can now be evaluated regarding the classes of errors handled by them. We also describe in general the different steps in data cleansing, specify the methods used within the cleansing process, and outline remaining problems and challenges for data cleansing research.

The structure of the paper is as follows. In the next section we motivate the need to cleanse data by describing the various intensions for data management, the occurring problems, and resulting consequences from anomalies in the data. In Section 3 we classify the anomalies in data in more detail. In Section 4 our perception of data cleansing is defined and a set of quality criteria is specified that represent measures for the status of data – before and after executing a cleansing process. Hence, these criteria can be used to measure the necessity and success of data cleansing applications. In Section 5 we outline the different steps in data cleansing. In Section 6 the methods used in data cleansing are listed and described in short. Section 7 gives an overview and comparison of existing projects for data cleansing according to our specification of comprehensive data cleansing. Section 8 outlines existing challenges and open problems within the research field of data cleansing. We conclude in Section 9.

2 Motivation

The processing and analysis of data is motivated by different objectives. Data are symbolic representations of information, i.e., facts or entities from the world, depicted by symbolic values. Data is collected to form a representation of a certain part of the world, called the mini-world (M) or Universe of Discourse (UoD) [EN00]. For the rest of this paper we consider data items to be tuples, i.e., sets of related discrete values from a finite set of domains. Each tuple represents an entity from the mini-world with each of its values in turn representing one of the entity's properties, e.g., persons with their properties name, date of birth, height, and weight.

The main objective for collecting and processing data is to fulfil different tasks in (i) administration, e.g., to keep track of the employees in a company, the customers of our company, or the sales volume of our companies branches, (ii) supporting business process, e.g., using the customers address list for direct mailing to advertise new products, and (iii) analysis and interpretation to support decision making and to generate new information and knowledge. Decision making in business and government is based on analysis of data to discover trends and keep tab on developments and investments. Scientists analyse data to gain new information and knowledge about the partition of the world they are investigating.

Anomaly is a property of data values that renders them a wrong representation of the mini-world. They may result from erroneous measurements, lazy input habits, omissions occurring while collecting and maintaining data, etc. They might also stem from misinterpretations in data analysis or due to changes in the mini-world that are unnoticed or not reflected by changes to the representing data. A special type of anomaly is redundancy, i.e., multiple tuples representing the same fact or overlapping parts of it. Throughout the reminder of this paper we will use the term anomaly and error synonymously.

The efficient and effective use of data is hampered by data anomalies. By efficient we mean the predominantly automatic, high performance processing and by effective the attainment reliable and profitable outcomes. We call data containing anomalies erroneous or dirty. There are various examples and scenarios illustrating the problems caused by erroneous data. Here we give a short example for each of the objectives mentioned above.

Administration: The government analyses data gained by population census to decide, which regions of the country require further investments in the infrastructure, like schools and other educational facilities, because of expected future trends. If the rate of birth in one region has increased over the last couple of years the existing schools and teachers employed might not be sufficient the handle the amount of students expected. Thus, additional schools or employment of teachers will be needed. Inaccuracies in analysed data can lead to false conclusions and misdirected investments. In this example, erroneous data could result in an over- or

undersupply of educational facilities if the information about the birth rate was inaccurate, or if many people have moved to other regions.

Supporting business processes: Erroneous data leads to unnecessary costs and probably loss of reputation when used to support business processes. Consider a company using a list of consumers with their addresses and buying habits and preferences to advertise a new product by direct mailing. Invalid addresses cause the letters to be returned as undeliverable. People duplicated in the mailing list account for multiple letters being send to the same person, leading to unnecessary expenses and frustration. Inaccurate information about consumer buying habits and preferences contaminate and falsify the target group, resulting in advertisement of products that do not correspond to consumers needs. Companies trading such data face the possibility of an additional loss of reputation in case of erroneous data.

Analysis: In genome research, functional annotation of proteins is used to estimate the function of uncharacterised proteins by comparing their amino acid sequences and transferring functional annotation in case of high sequence similarity. Existing errors within annotations are therefore propagated to the newly discovered protein. In [Bre99] the error rate for functional annotation of proteins is estimated to be over 8% by comparing analysis results of three independent research groups annotating the proteome of *Mycoplasma genitalium*. With the increased dependency on automatic annotation methods because of the high data volume this rate of errors can only be expected to rise.

3 Data Anomalies

Before describing and classifying data anomalies we define terms and concepts regarding the schema and instances of a data collection. Here, we closely follow the definitions in [EN00] and extend them to our needs.

3.1 Data Model

We assume the existence of a non-empty set $D = \{D_1, \ldots, D_m\}$ of domains whose values are sequences of symbols from an finite, non-empty alphabet $\sum_D$. For each domain D_i, $1 \le i \le m$, there exists an arbitrary grammar describing the syntax of the domain values. Domains therefore represent regular languages, i.e., words from $\sum_D^*$ generated by a grammar $G(D_i)$, called the domain format.

A relation schema R is a name and a list of attributes, $A_1, \ldots, A_n$, denoted by $R(A_1, \ldots, A_n)$. The degree #R of a relation schema R is the number of attributes n. Each attribute A is the name of a role played by some domain from D, denoted by *dom*(A), in the relation schema. A relation (or relation instance) r of the relation schema R is a set of n-tuples $r = \{t_1, \ldots, t_n\}$. Each t_i is a list of n values $t_i = \langle v_{i1}, \ldots, v_{in} \rangle$, where each value v_{ij}, $1 \le j \le n$, is an element of $dom(A_j)$. The degree #t of a tuple is the number of values n. The set of values for attribute A_j, in instance r of $R(A_1, \ldots, A_n)$ is denoted by *val*(r, A_j).

A database schema S contains a set of relational schemas $\{R_1, \ldots, R_k\}$ and a set of integrity constraints Γ that must hold for each instance of $S = (\{R_1, \ldots, R_k\}, \Gamma)$. The instance s of a database schema is a list of relation instances, denoted by $s = \langle r_1, \ldots, r_k \rangle$, where each instance r_i, $1 \le i \le k$, is an instance of relational schema R_i, satisfying the constraints in Γ. Each integrity constraint $\gamma \in \Gamma$ is a function that associates a Boolean value with an instance s. A database instance s satisfies γ if $\gamma(s) = \text{true}$. We call a database instance a data collection if the schema and the integrity constraints are not fully enforced by software components, called the database management system.

3.2 Classification of Data Anomalies

We roughly classify data anomalies into syntactical, semantic, and coverage anomalies. Syntactical anomalies describe characteristics concerning the format and values used for representation of the entities. Semantic anomalies hinder the data collection from being a comprehensive and non-redundant representation the mini-world. Coverage anomalies decrease the amount of entities and entity properties from the mini-world that are represented in the data collection.

3.2.1 Syntactical Anomalies

Lexical errors name discrepancies between the structure of the data items and the specified format. This is the case if the number of values are unexpected low/high for a tuple t, i.e., the degree of the tuple #t is different from #R, the degree of the anticipated relation schema for the tuple. For example, assume the data to be stored in table form with each row representing an tuple and each column an attribute (Figure 1). If we expect the table to have five columns because each tuple has five attributes but some or all of the rows contain only four columns then the actual structure of the data does not conform to the specified format. Data which is lexically correct can be parsed into a specified token structure deducible from the relational schema.

Name	Age	Gender	Size
Peter	23	M	7’1
Tom	34	M	
Sue	21	5’8	

Figure 1: Data table with lexical errors

Domain format errors specify errors where the given value for an attribute A does not conform with the anticipated domain format G(*dom*(A)). For example, an attribute NAME may have a domain specified to be $\sum_D^*$, $\sum_D^*$. Given a concrete value ‘*John Smith*’ while this is a possibly correct name it does not satisfy the defined format of the domain values because of the missing comma between the words.

The anomaly classes lexical errors and domain format errors often are subsumed by the term **syntactical error** because they represent violations of the overall format.

Irregularities are concerned with the non-uniform use of values, units and abbreviations. This can happen for example if one use different currencies to specify an employees salary. This is especially profound if the currency is not explicitly listed with each value, and is assumed to be uniform. This results in values being correct representations of facts if we have the necessary knowledge about their representation needed to interpret them. Another example is the use or the different use of abbreviations. Inconsistent use of values harden the comparability of tuples which is influencing clustering for duplicate elimination or other analysis tasks.

3.2.2 Semantic Anomalies

Integrity constraint violations describe tuples (or sets of tuples) that do not satisfy one or more of the integrity constraints in Γ. Integrity constraints are used to describe our understanding of the mini-world by restricting the set of valid instances. Each constraint is a rule representing knowledge about the domain and the values allowed for representing certain facts, e.g., AGE ≥ 0.

Contradictions are values within one tuple or between tuples that violate some kind of dependency between the values. An example for the first case could be a contradiction between the attribute AGE and DATE_OF_BIRTH for a tuple representing persons. Contradictions are either violations of functional dependencies that can be represented as integrity constraints or duplicates with inexact values. They are therefore not regarded as separate data anomaly throughout the reminder of this paper.

Duplicates are two or more tuples representing the same entity from the mini-world. The values of these tuples do not need to be complete identical. Inexact duplicates are specific cases of contradiction between two or more tuples. They represent the same entity but with different values for all or some of its properties. This hardens the detection of duplicates and there mergence.

Invalid tuples represent by far the most complicated class of anomaly found in data collections. By invalid we mean tuples that do not display anomalies of the classes defined above but still do not represent valid entities from the mini-world. They result from our inability to describe reality within a formal model by integrity constraints. They are extremely hard to find and even more complicated to correct because there are no rules (or constraints) which are violated by these tuples and on the other hand we only have incomplete knowledge about every entity in the mini-world.

3.2.3 Coverage Anomalies

Missing values are the result of omissions while collecting the data. This is to some degree a constraint violation if we have null values for attributes where there exists a NOT NULL constraint for them. In other cases we might not have such a constraint thus allowing null values for an attribute. In these cases we have to decide whether the value exists in the mini-world and has to be deduced here or not. Only those missing values that should exist in our data collection, because the entity has an according property with a measurable value, but are not contained, are regarded as anomalies.

Missing tuples result from omissions of complete entities existent in the mini-world that are not represented by tuples in the data collection.

Anomalies could further be classified according to the amount of data accounting for the constraint violation. This can be single values, values within a tuple, values within one or more columns of the data collection or tuples and sets of tuples from different relations. This is in close relation to the classification of data cleansing problems in [RD00]. There, the authors also distinguish schema-related and instance-related problems. Schema-level problems can be addressed at the schema level by schema evolution, schema translation, and schema integration. Examples for schema-level problems are missing integrity constraints or domain format errors. The schema-level problems are always reflected by anomalies within the instances. In our definition the cleansing of data is performed on the relation instances and does not have to be reflected by changes within the schema. We therefore do not distinguish between schema and instance related problems.

4 Data Cleansing and Data Quality

The existence of anomalies in real-world data motivates the development and application of data cleansing methods. With the above definitions we are now able to define data cleansing and specify how to measure the success of cleansing erroneous data.

4.1 Data Quality

To be processable and interpretable in a effective and efficient manner, data has to satisfy a set of quality criteria. Data satisfying those quality criteria is said to be of high quality. In general, data quality[1] is defined as an aggregated value over a set of quality criteria [Nau02]. Starting with the quality criteria defined in [Nau02, Ton00] we describe the set of criteria that are affected by comprehensive data cleansing and define how to assess scores for each one of them for an existing data collection.

To measure the quality of a data collection, scores have to be assessed for each of the quality criteria. The assessment of scores for quality criteria can be used to quantify the necessity of data cleansing for a data collection as well as the success of a performed data cleansing process on a data collection. Quality criteria can also be used within optimization of data cleansing by specifying priorities for each of the criteria which in turn influences the execution of data cleansing methods affecting the specific criteria.

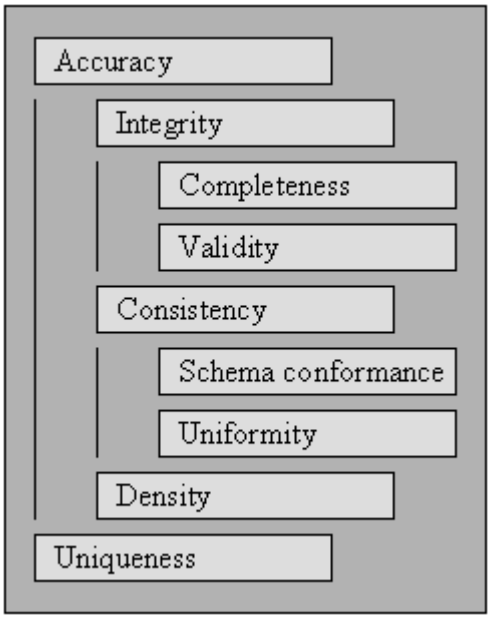

Figure 2: Hierarchy of data quality criteria

4.2 Quality Criteria

The quality criteria defined for comprehensive data cleansing form a hierarchy. This hierarchy results from quality criteria being sub-divided into finer grained quality criteria, i.e., criteria being used as short-cut for a set of quality criteria. Figure 2 shows the resulting hierarchy of the criteria defined below. For each of the criteria we describe how to assess the quality score for a given data collection. Here we assume each collection consisting of only one relational instance, i.e. $S = (\{R\}, \Gamma)$ and therefore $s \equiv r$. We do not assume the enforcement of integrity constraints on r, i.e., not every constraint $\gamma \in \Gamma$ is true for all tuples in r. The set of entities in the mini-world is denoted by M. The score for each of the quality criteria is influenced by one or more of the anomalies defined in section 3.2. In table 1 we show the quality criteria and the anomalies affecting them.

Accuracy in [Nau02] is defined as the quotient of the number of correct values in the data collection and the overall number of values. Here we use the term accuracy in a different way.

[1] We will use the term data quality instead of information quality because we disregard criteria that are concerned with the information consumer and the subjective usefulness of the presented information.

Accuracy is describe as an aggregated value over the quality criteria Integrity, Consistency, and Density. Intuitively this describes an exact, uniform and complete representation of the mini-world, i.e., a data collection not containing any of the defined anomalies except for Duplicates. We define the quality criteria Integrity, Consistency, and Density in the following.

Integrity is used as defined in [Mot89]. It is further divided into the criteria Completeness and Validity and therefore again an aggregated value over quality criteria. Intuitively, an integral data collection contains representations of all the entities in the mini-world and only of those, i.e., there are no invalid tuples or integrity constraint violations as well as no missing tuples.

Completeness is defined as the quotient of entities from M being represented by a tuple in r and the overall number of entities in M. Achieving this form of completeness is not a primary data cleansing concern but more of a data integration problem. We achieve completeness within data cleansing by correcting tuples containing anomalies and by not just deleting these tuples if they are representations of entities from M. Within integrity constraint enforcement (see section 6.3) it is also possible that additional tuples are generated, representing obviously existing entities from M that are currently unrepresented in r.

Validity is the quotient of entities from M being represented by tuples in r and the total amount of tuples in r, i.e., the percentage of tuples in r representing (valid) entities from M. As mentioned in section 3.2.2 the identification of invalid tuples is complicated and sometimes impossible because of the inability or high cost for repeating measurements to verify the correctness of a measured value. Validity can be approximated using the integrity constraints specified in Γ. Integrity constraints represent our understanding of the regularities in the mini-world. Tuples violating integrity constraints are therefore regarded as invalid tuples. Constraint violations arise within systems that do not enforce integrity constraints completely. This might be because of system limitations or on demand of the user, probably because of performance issues. We approximate validity as the quotient of tuples satisfying all integrity constraints in Γ and the overall number of tuples in r.

Consistency concerns the syntactical anomalies as well as contradictions. It is further divided into Schema conformance and Uniformity forming another aggregated value over quality criteria. Intuitively a consistent data collection is syntactically uniform and free of contradictions.

Schema conformance is the quotient of tuples in r conform to the syntactical structure defined by schema R and the overall number of tuples in r. This includes the domain formats $G(dom(A_i))$ of the attributes $A_1, \ldots A_n$ in R. Some systems do not enforce the complete syntactical structure thus allowing for tuples within the collection that are not absolutely format conform. This is especially true for the relational database systems where the adherence of domain formats is incumbent to the user.

Uniformity is directly related to irregularities, i.e., the proper use of values within each attribute. The semantic and usage of a values should be uniform within $val(r, A_i)$ for each A_i in R. Uniformity is the quotient of attributes not containing irregularities in their values and n, the total number of attributes in r.

Density is quotient of missing values in the tuples of r and the number of total values that ought be known because they exist for a represented entity. There still can be values or properties non-existent that have to be represented by null values having the exact meaning of not being known. These are no downgrades of data quality. It would be a downgrade if we try to estimate a value for them.

Uniqueness is the quotient of tuples representing the same entity in the mini-world and the total number of tuples in r. A collection that is unique does not contain duplicates. Recalling

the definition of accuracy as a collection not containing any anomalies except duplicates, a data collection being accurate and unique does not contain any of the anomalies defined in section 3. This describes a data collection not being in need for data cleansing.

Table 1 lists the defined quality criteria and anomalies affecting them. Included are only those criteria that are not further divided into sub-criteria. Each • indicates direct downgrading of the quality criteria while – indicates that the occurrence of this anomaly hampers the detection of other anomalies downgrading the quality criteria.

	Completeness	Validity	Schema conform.	Uniformity	Density	Uniqueness
Lexical error		–	•	–	–	–
Domain format error		–	•	–		–
Irregularities		–		•		–
Constraint Violation		•				
Missing Value					•	–
Missing Tuple	•					
Duplicates						•
Invalid Tuple		•				

Table 1 : Data anomalies affecting data quality criteria

Each • indicates direct downgrading of the quality criteria while – indicates that the occurrence of this anomaly hampers the detection of other anomalies downgrading the quality criteria.

5 A Process Perspective on Data Cleansing

Comprehensive data cleansing is defined as the entirety of operations performed on existing data to remove anomalies and receive a data collection being an accurate and unique representation of the mini-world. It is a (semi-)automatic process of operations performed on data that perform, preferable in this order, (i) format adaptation for tuples and values, (ii) integrity constraint enforcement, (iii) derivation of missing values from existing ones, (iv) removing contradictions within or between tuples, (v) merging and eliminating duplicates, and (vi) detection of outliers, i.e., tuples and values having a high potential of being invalid. Data cleansing may include structural transformation, i.e. transforming the data into a format that is better manageable or better fitting the mini-world. The quality of schema though is not a direct concern of data cleansing and therefore not listed with the quality criteria defined above. We will list the methods for data cleansing in more detail in section 6.

According to [MM00, RH01], the process of data cleansing comprises the three major steps (i) auditing data to identify the types of anomalies reducing the data quality, (ii) choosing appropriate methods to automatically detect and remove them, and (iii) applying the methods to

the tuples in the data collection. Steps (ii) and (iii) can be seen as specification and execution of a data cleansing workflow. We add another task (iv), the post-processing or control step where we exam the results and perform exception handling for the tuples not corrected within the actual processing. Figure 3 shows the steps within the data cleansing process. The specification of the data cleansing process and the control of its execution is done by one or more domain experts, i.e., experts with knowledge about the mini-world and its regularities and peculiarities.

The process of data cleansing normally never finishes, because anomalies like invalid tuples are very hard to find and eliminate. Depending on the intended application of the data it has to be decided how much effort is required to spend for data cleansing.

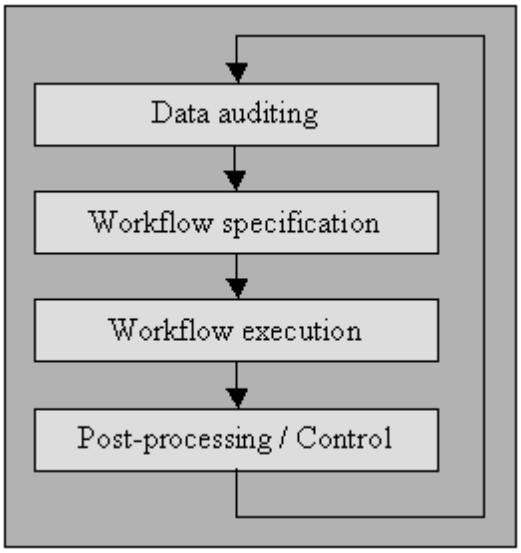

Figure 3: Data cleansing process

5.1 Data Auditing

The first step in data cleansing is auditing the data to find the types of anomalies contained within it. The data is audited using statistical methods and parsing the data to detect syntactical anomalies. The instance analysis of individual attributes (data profiling) and the whole data collection (data mining) derives information such as minimal and maximal length, value range, frequency of values, variance, uniqueness, occurrence of null values, typical string patterns as well as patterns specific in the complete data collection (functional dependencies and association rules) [RD00].

The results of auditing the data support the specification of integrity constraints and domain formats. Integrity constraints are depending on the application domain and are specified by domain expert. Each constraint is checked to identify possible violating tuples. For one-time data cleansing only those constraints that are violated within the given data collection have to be further regarded within the cleansing process. Auditing data also includes the search for characteristics in data that can later be used for the correction of anomalies.

As a result of the first step in the data cleansing process there should be an indication for each of the possible anomalies to whether it occurs within the data collection and with which kind of characteristics. For each of these occurrences a function, called tuple partitioner, for detecting all of its instances in the collection should be available or directly inferable.

5.2 Workflow Specification

Detection and elimination of anomalies is performed by a sequence of operations on the data. This is called the data cleansing workflow. It is specified after auditing the data to gain information about the existing anomalies in the data collection at hand. One of the main challenges in data cleansing insists in the specification of an cleansing workflow that is to be applied to the dirty data automatically eliminating all anomalies in the data.

For the specification of the operations intending to modify erroneous data the cause of anomalies have to be known and closely considered. The causes for anomalies are manifold. Typical causes for anomalies are impreciseness in measurement or systematic errors in experimental setup, false statements or lazy input habits, inconsistent use of abbreviations, misuse or misinterpretation of data input fields incorrect or careless interpretation of the analysis results, or even be a consequence of anomalies in the data analysed, leading to invalid tuples results and to a propagation of errors. For the specification of correcting methods the cause of error has to be considered. If for example we assume an anomaly to result from typing errors at data input the layout of the keyboard can help in specifying and assessing the generation of possible solutions. The knowledge about the experiments performed also helps identify and correct systematic errors.

Syntax errors are normally handled first because the data has to be automatically process to detect and remove the other types of anomalies which is additionally hindered by syntax errors. Otherwise there is not specific order in eliminating anomalies by the data cleansing workflow.

In [RD00] another step is defined after specifying the cleansing workflow and before its execution, the verification. Here, the correctness and effectiveness of the workflow is tested and evaluated. We assume this verification step to be an integral part of the workflow specification.

5.3 Workflow Execution

The data cleansing workflow is executed after specification and verification of its correctness. The implementation should enable an efficient performance even on large sets of data. This is often a trade-off because the execution of a data cleansing operation can be quite computing intensive, especially if a comprehensive and 100% complete elimination of anomalies is desired. So we need a heuristics to can achieve the best accuracy while still having an acceptable execution speed [GFSSS01b].

There is a great demand for interaction with domain experts during the execution of the data cleansing workflow. In difficult cases the expert has to decide whether a tuple is erroneous or not and specify or select the correct modification for erroneous tuples from a set of solutions. The interaction with the expert is expensive and time consuming. Tuples that cannot not be corrected immediately are often logged for manual inspection after executing the cleansing workflow.

5.4 Post-Processing and Controlling

After executing the cleansing workflow, the results are inspected to again verify the correctness of the specified operations. Within the controlling step the tuples that could not be corrected initially are inspected intending to correct them manually. This results in a new cycle in the data cleansing process, starting by auditing the data and searching for characteristics in exceptional data that allow us to specify an additional workflow to cleanse the data further by automatic processing. This might be supported by learning sequences of cleansing operations

for certain anomalies by example, i.e., the expert cleanses one tuple by example and the system learns from this to perform the cleansing of other occurrences of the anomaly automatically.

6 Methods used for Data Cleansing

There exists a multitude of different methods used within the data cleansing process. In this section we intend to give a short overview for the most popular of them.

6.1 Parsing

Parsing in data cleansing is performed for the detection of syntax errors. A parser for a grammar G is a program that decides for a given string whether it is an element of the language defined by the grammar G. In the context of compilers for programming languages the strings represent computer programs [AU79]. In data cleansing the strings are either complete tuples of a relational instance or attribute values from as domain. Those strings that represent syntax errors have to be corrected. This can be done for example using edit distance functions choosing the possible correction with the minimal edit distance.

The existence and amount of syntax errors in a data collection depends on the extend of schema enforcement in the environment where the data is kept. If the data is contained in flat files there exists the possibility of lexical errors and domain errors. In this case, a grammar derived from the file structure is used and the stings represent complete tuples. Data being managed by database management systems is not expected to contain lexical or domain errors. Still, domain format errors can exists for each of the attributes. The grammar used for parsing is the domain format G(A). The specification of domain formats can be supported by pattern learning techniques like they are described in [RH01]. They use a sample set of values to deduce the format of the domain. They also generate a discrepancy detector which is used for anomaly detection.

6.2 Data Transformation

Data transformation intends to map the data from their given format into the format expected by the application [ACMM+99]. The transformations affect the schema of the tuples as well as the domains of their values. Schema transformation is often performed in close conjunction with data cleansing. The data from various sources is mapped into a common schema better fitting the needs of the intended application. The correction of values have to be performed only in cases where the input data does not conform to its schema leading to failures in the transformation process. This makes data cleansing and schema transformation supplemental tasks.

Standardization and normalization are transformations on the instance level used with the intension of removing irregularities in data. This includes simple value conversion or translating functions as well as normalizing numeric values to lie in a fixed interval given by the minimum and maximum values [SS01].

6.3 Integrity Constraint Enforcement

The elimination of integrity constraints violations is closely related to techniques from integrity constraint enforcement. In general, integrity constraint enforcement ensures the satisfaction of integrity constraints after transactions modifying a data collection by inserting, delet-

ing, or updating tuples [MT99] have been performed. The two different approaches are integrity constraint checking and integrity constraint maintenance. Integrity constraint checking rejects transactions that, if applied, would violate some integrity constraint. Integrity constraint maintenance is concerned with identifying additional updates (i.e. repair) to be added to the original transaction to guarantee that the resulting data collection does not violate any integrity constraint.

In [EBRS+01] it is shown that integrity constraint enforcement is applicable to some degree in data cleansing but it also has some limitations. The basic idea is to automatically identify from the set of integrity constraints a series of modifications and apply them on the data collection so that afterwards the collection does not contain any further integrity constraint violations. The authors outline some limitations of the application of these techniques for data cleansing (or data reconciliation as they call it). The main limitation is based on the fact that integrity enforcement techniques operate on valid database states and single transactions violating some of the constraints. When cleansing data, we have a large amount of data and constraint violations and therefore a large search space resp. an even larger amount of repairs (because we could first repair tuple 1 and then based upon this tuple 2 or the other way around leading to different results). In addition, the generated repairs primarily focus on the insertion or deletion of tuples rather than the correction of single values to eliminate constraint violations. Another problem is based upon the fact that existing approaches use strategies to limit the set of possible repairs. These strategies are shown to remove important repairs from their applications domain point of view.

The application of integrity enforcement as seen by the authors is limited to a supportive rather than a dominant role. Control of the process must remain with the user all the time. The tool presents possible repairs to the user, makes the state of repairs resistant and supports safe experimentation.

6.4 Duplicate elimination

There are several approaches for duplicate elimination or record linkage which is a part of data cleansing. Every duplicate detection method proposed requires an algorithm for determine whether two or more tuples are duplicate representations of the same entity. For efficient duplicate detection every tuple has to be compared to every other tuple using this duplicate detection method. In [HS95] a fast method (sorted neighbourhood method) is developed to reduce the number of required comparisons. The tuples are sorted by a key, possibly constructed from the attributes of the relation, that hopefully brings duplicate tuples close to another. Then only the tuples within a small window (floating over the relation) are compared with each other to find duplicates. The identification whether two tuples are duplicates is done using rules based on domain specific knowledge. In order to improve accuracy, the results of several passes of duplicate detection can be combined by explicitly computing transitive closure of all discovered pairwise duplicate tuples. Not much is said about how the duplicates are merged. In [LLLK99] this approach is further extended on tuples with syntactically structured attributes, i.e., not conform with the domain format. Because of format errors, tuples that are duplicates might not be close together after sorting. Therefore, the attribute values are tokenized into lexical units and the tokens are then sorted within each attribute before the whole relation is sorted. The pairwise tuple matching algorithms used in most previous work have been application-specific. In [ME97] a domain independent matching-algorithm is used in that it can be used without any modification in different applications. Therefore an edit-distance algorithm based on the Smith-Waterman algorithm is used [SM81]. In [ACG02] the authors propose an approach that avoids the problems of the sorting methods. They rely on the dimensional hierarchies typically associated with dimensional tables in a data warehouse

[ACG02]. Those are hierarchies of tables typically in 1:n-relationships expressed by key-foreign key relationships. Each tuple in the 1-relation is associated with a set of tuples in the n-relation. The degree of overlap between sets associated with two tuples from the 1-relation is a measure of co-occurrence between them, and can be used to detect duplicates.

6.5 Statistical Methods

In [MM00] the application of statistical methods in data cleansing is described. These methods can be used for the auditing of data as well as the correction of anomalies. Detection and elimination of complex errors representing invalid tuples go beyond the checking and enforcement of integrity constraints. They often involve relationships between two or more attributes that are very difficult to uncover and describe by integrity constraints. This can be viewed as a problem in outlier detection, i.e., minorities of tuples and values that do not conform to the general characteristics of a given data collection.

By analysing the data using the values of mean, standard deviation, range, or clustering algorithms a domain expert may find values that are unexpected indicating possible invalid tuples. They can then be analysed more detail. The correction of such errors is often impossible (besides simply deleting them) because the true values or unknown. Possible solution include statistical methods like setting the values to some average or other statistical value. Outliers can also be detected as violations of association rules [AIS93] or other existing patterns in the data.

Another anomaly handled by statistical methods are missing values. Missing values are handled based on filling-in (imputing) one or more plausible values [Win99]. The generation of imputations require computationally intensive data augmentation algorithms as described in [Sch97, Tan96].

7 Existing Approaches for Data Cleansing

In this section we describe existing data cleansing projects. With the definition of anomalies occurring in data, the quality criteria affected by them, and a description of the data cleansing process and the methods used within it, we are now able to compare existing data cleansing approaches. The comparison will be done regarding the data anomalies eliminated within each of the approaches and the methods used by them.

7.1 AJAX

AJAX [GFSS00, GFSSS01b] is an extensible and flexible framework attempting to separate the logical and physical levels of data cleansing. The logical level supports the design of the data cleansing workflow and specification of cleansing operations performed, while the physical level regards their implementation. AJAX major concern is transforming existing data from one or more data collections into a target schema and eliminating duplicates within this process. For this purpose a declarative language based on a set of five transformation operations is defined. The transformations are mapping, view, matching, clustering, and merging.

The mapping operator expresses arbitrary one-to-many mappings between a single input relation and one or more output relations. The view operator is an equivalent to the view operator in SQL simply expressing limited many-to-one mappings with some additional integrity checking. The matching operator computes an approximate join between two relations assigning a distance value to each pair in the Cartesian product using an arbitrary distance function.

This operator is fundamental for duplicate detection. The last operator is the merge taking a single relation as input, partitioning it according to some grouping attributes and collapsing each partition into a single tuple using an arbitrary aggregation function. This enhances the SQL-99 standard with user defined aggregation functions. The semantics of the five operators involves the generation of exceptions that provide the foundation for interacting with the expert user.

The mapping operator performs the mapping between different formats as well as format standardisation, i.e., unification affecting the quality criteria format correspondence and uniformity. The integrity constraints checked by the view operator are those expressible in SQL. In case of constraint violation an exception is generated. This enables interacting with the user. It is not further mentioned how integrity violating tuples are corrected. The match, cluster and merge operators enable duplicate elimination. How the tuples are merged is implemented within a function specified in the merge operation. This cannot be done declaratively.

The data cleansing process is specified by arranging the transformation operations as a linear data flow graph with each operation taking the output of one or more preceding operations as its input. A data lineage mechanism enables the users to inspect exceptions, analyze their provenance in the data cleansing process and afterwards correct the tuples that contributed to its generation. Corrected data can then be re-integrated into the data cleansing process.

7.2 FraQL

FraQL [SCS00, SS01] is another declarative language supporting the specification of a data cleansing process. The language is an extension to SQL based on an object-relational data model. It supports the specification of schema transformations as well as data transformations. at the instance level, i.e., standardization and normalization of values. This can be done using user-defined functions. The implementation of the user defined function has to be done for the domain specific requirements within the individual data cleansing process.

With its extended join and union operators in conjunction with user-defined reconciliation functions FraQL supports identification and elimination of duplicates. Similar to SQL, the union and join operators can be refined by an `on` clause specifying the comparison attributes (for the union operator) resp. the comparison expression (for join operators). Both the union and join operators can be applied with an additional `reconciled by` clause which denotes a used-defined function for resolution of contradictions between tuples fulfilling the comparison clause.

Also supported by FraQL is filling in missing values, and eliminating invalid tuples by detection and removal of outliers, and noise in data. Missing values can be filled-in with values computed from others, e.g. the average or median of the attribute or the attribute value of all tuples in a relation. Noise in data is handled by partitioning the data into bins or buckets based on a certain attribute and smoothing the data by different criteria like bucket mean, bucket median or bucket boundaries.

7.3 Potter’s Wheel

Potter’s Wheel [RH01] is an interactive data cleansing system that integrates data transformation and error detection using spreadsheet-like interface. The effects of the performed operations are shown immediately on tuples visible on screen. Error detection for the whole data collection is done automatically in the background. A set of operations, called transforms, are specified that support common schema transformations without explicit programming. These are value translations, that apply a function to every value in a column, One-to-one mappings that are column operations transforming individual rows, and Many-to-many mappings of

rows solving schematic heterogeneities where information is stored partly in data values, and partly in the schema. The anomalies handled by this approach are syntax errors and irregularities.

Potter's Wheel allows users to define custom domains, and corresponding algorithms to enforce domain format constraints. The arbitrary domains are closely related to our domain formats. Potter's Wheel lets users specify the desired results on example values, and automatically infers regular expressions describing the domain format. Therefore, the user does not have to specify them in advance. The inferred domain format specification can afterwards be used to detect discrepancies.

Specification of the data cleansing process is done interactively. The immediate feedback of performed transformations and error detection enables the users to gradually develop and refine the process as further discrepancies are found. This enables individual reaction on exceptions. The complete data cleansing process is not further documented.

7.4 ARKTOS

ARKTOS [VVSKS01] is a framework capable of modelling and executing the Extraction-Transformation-Load process (ETL process) for data warehouse creation. The authors consider data cleansing as an integral part of this ETL process which consists of single steps that extract relevant data from the sources, transform it to the target format and cleanse it, then loading it into the data warehouse. A meta-model is specified allowing the modelling of the complete ETL process. The single steps (cleansing operations) within the process are called activities. Each activity is linked to input and output relations. The logic performed by an activity is declaratively described by a SQL-statement. Each statement is associated with a particular error type and a policy which specifies the behaviour (the action to be performed) in case of error occurrence.

Six types of errors can be considered within an ETL process specified and executed in the ARKTOS framework. PRIMARY KEY VIOLATION, UNIQUENESS VIOLATION and REFERENCE VIOLATION are special cases of integrity constraint violations. The error type NULL EXISTENCE is concerned with the elimination of missing values. The remaining error types are DOMAIN MISMATCH and FORMAT MISMATCH referring to lexical and domain format errors.

The policies for error correction simply are IGNORE, but without explicitly marking the erroneous tuple, DELETE as well as WRITE TO FILE and INSERT TO TABLE with the expected semantics. The later two provide the only possibility for interaction with the user.

The success of data cleansing can be measured for each activity by executing a similar SQL-statement counting the matching/violating tuples. The authors define two languages for declaratively specifying of the ETL process. This is accompanied with a graphical scenario builder.

7.5 IntelliClean

IntelliClean [LLL00, LLL01] is a rule based approach to data cleansing with the main focus on duplicate elimination. The proposed framework consists of three stages. In the Pre-Processing stage syntactical errors are eliminated and the values are standardized in format and consistency of used abbreviations. It its not specified in detail, how this is accomplished. The Processing stage represents the evaluation of cleansing rules on the conditioned data items that specify actions to be taken under certain circumstances. There are four different classes of rules. Duplicate identification rules specify the conditions under which tuples are

classified as duplicates. Merge/Purge rules specify how duplicate tuples are to be handled. It is not specified how the merging is to be performed or how its functionality can be declared. If no merge/purge rule has been specified, duplicate tuples can also manually be merged at the next stage. Update rules specify the way data is to be updated in a particular situation. This enables the specification of integrity constraint enforcing rules. For each integrity constraint an Update rule defines how to modify the tuple in order to satisfy the constraint. Update rules can also be used to specify how missing values ought to be filled-in. Alert rules specify conditions under which the user is notified allowing for certain actions.

During the first two stages of the data cleansing process the actions taken are logged providing documentation of the performed operations. In the human verification and validation stage these logs are investigated to verify and possibly correct the performed actions.

7.6 Comparison

In table 2 the described data cleansing approaches are compared according to the types of anomalies handled by them and a short indication about the methods and techniques used as defined by each of them. The term *User Defined* indicates that detecting and eliminating the specific anomaly is possible but not specified in detail.

	AJAX	**FraQL**	**Potter's Wheel**	**ARKTOS**	**IntelliClean**
Lexical Error					
Domain Format Error	User Defined	User Defined	Pattern learning	User Defined	User Defined
Irregularities	User Defined	User Defined			User Defined
Constraint Violation	Filter violating tuples			Three types of constraints	Alert and Update rule
Missing Values	User Defined	Statistical Values			Update rule
Missing Tuples					
Duplicates	Match, Cluster and Merge	Union, Join, and Reconciliation			Merge/Purge rule
Invalid Tuple		Statistical Methods			

Table 2. Comparison of data cleansing approaches

8 Challenges and Open Problems

In this section we outline some open problems and challenges in data cleansing that are not satisfied until now by the existing approaches. This mainly concerns the management of multiple, alternative values as possible corrections, keeping track of the cleansing lineage for documentation efficient reaction to changes in the used data sources, and the specification and development of an appropriate framework supporting the data cleansing process.

8.1 Error Correction and Conflict Resolution

The most challenging problem within data cleansing remains the correction of values to eliminate domain format errors, constraint violations, duplicates and invalid tuples. In many cases the available information and knowledge is insufficient to determine the correct modification of tuples to remove these anomalies. This leaves deleting those tuples as the only practical solution. This deletion of tuples leads to a loss of information if the tuple is not invalid as a whole. This loss of information can be avoided by keeping the tuple in the data collection and mask the erroneous values until appropriate information for error correction is available. The data management system is then responsible for enabling the user to include and exclude erroneous tuples in processing and analysis where this is desired.

In other cases the proper correction is known only roughly. This leads to a set of alternative values. The same is true when dissolving contradictions and merging duplicates without exactly knowing which of the contradicting values is the correct one. The ability of managing alternative values allows to defer the error correction until one of the alternatives is selected as the right correction. Keeping alternative values has a major impact on managing and processing the data. Logically, each of the alternatives forms a distinct version of the data collection, because the alternatives are mutually exclusive. It is a technical challenge to manage the large amount of different logical versions and still enable high performance in accessing and processing them.

When performing data cleansing one has to keep track of the version of data used because the deduced values can depend on a certain value from the set of alternatives of being true. If this specific value later becomes invalid, maybe because another value is selected as the correct alternative, all deduced and corrected values based on the now invalid value have to be discarded. For this reason the cleansing lineage of corrected values has to maintained. By cleansing lineage we mean the entirety of values and tuples used within the cleansing of a certain tuple. If any value in the lineage becomes invalid or changes the performed operations have to be redone to verify the result is still valid. The management of cleansing lineage is also of interest for the cleansing challenges described in the following two sections.

8.2 Maintenance of Cleansed Data

Cleansing data is a time consuming and expensive task. After having performed data cleansing and achieved a data collection free of errors one does not want to perform the whole data cleansing process in its entirety after some of the values in data collection change. Only the part of the cleansing process should be re-performed that is affected by the changed value. This affection can be determined by analysing the cleansing lineage. Cleansing lineage therefore is kept not only for tuples that have been corrected, but also for those that have been verified within the cleansing process as being correct. After one of the values in the data collection has changed, the cleansing workflow has to be repeated for those tuples that contains the changed value as part of their cleansing lineage.

The broad definition of require the collection and management of a large amount of additional meta-data to keep track of cleansing lineage. Efficient ways of managing the cleansing lineage have to be developed. It is also of interest to determine which additional information resulting from the initial workflow execution has to be collected in order to be able to speed-up ensuing cleansing workflow executions.

8.3 Data Cleansing in Virtually Integrated Environments

The problems mentioned in the preceding section intensify when performing data cleansing in environments of virtually integrated sources, like IBM's DiscoveryLink [HSKK+01]. In these environments it is often impossible to propagate corrections to the sources because of their autonomy. Therefore, cleansing of data has to be performed every time the data is accessed. This considerably decreases the response time. By collecting and managing appropriate meta-data like cleansing lineage and performed operations in a data cleansing middleware the performance could be increased considerably. This could also prevent unnecessary data cleansing if the data in the sources does not change between accessing the sources. There still remains the trade-off between collecting huge amounts of meta-data and materializing the complete integrated data collection. The middleware should only collect as much data as necessary but still enable fast cleansing of data.

8.4 Data Cleansing Framework

In many cases it will not be possible to describe the whole data cleansing graph in advance. This makes data cleansing an iterative, interactive and explorative task. The whole data cleansing process is more the result of flexible workflow execution. Process specification, execution and documentation should be done within a data cleansing framework which in turn is closely coupled with other data processing activities like transformation, integration, and maintenance activities. The framework is a collection of methods for error detection and elimination as well as methods for auditing data and specifying the cleansing task using appropriate user interfaces.

Data cleansing should be tightly integrated with the data maintenance within the same framework. The requirements for a data cleansing framework are (this might overlap with requirements for other data integration and management tasks):

- The ability to uniformly describe the structure and access the contend of data sources possibly heterogeneous in the data model used. We also want to be able to be notified whenever data in the data sources are updated, deleted or added.
- Abstract definition of possible error types so the user can be guided within the data auditing task. For each error type there need to be methods that can be configured and used for error detection and elimination. Machine learning techniques can be applied within the data cleansing framework. By learning about certain error types, their instances, and their correction the framework is enabled to even better support the user in providing useful hints towards possible proceeding operations executing the data cleansing process.
- For data cleansing a convenient user interface is needed. There has to be a formal language or graphical assistance to describe the data cleansing process formally. This has to be closely coupled with languages used for the specification of other activities in maintaining and processing data.
- Te performed activities are logged to enable a later trace-back and the generation of common workflows for the correction of errors of the same type. The documentation of the performed operations as well as the collection of additional meta-data allows the

verification of correctness for each of the tuples. This include the cleansing lineage. By logging the operations performed in explorative cleansing of individual tuples an appropriate sequence of cleansing operations can automatically derived for cleansing all tuples showing the specific anomaly.

9 Conclusion

Data cleansing is applied with different intensions and within different areas of the data integration and management process. It is defined it as the sequence of operations intending to enhance to overall data quality of a data collection. There is only a rough description of the procedure in data cleansing as it is highly domain dependent and explorative. Existing data cleansing approaches mostly focus on the transformation of data and the elimination of duplicates. Some approaches enable the declarative specification of a more comprehensive data cleansing processes still leaving most of the implementation details for the cleansing operation to the user. There still remain a lot of open problems and challenges in data cleansing. They mainly concern the management of multiple, alternative values, the management and documentation of performed cleansing operations and the cleansing lineage, as well as the specification and development of an appropriate framework supporting the data cleansing process.

Literature

[ACG02] R. Ananthakrishna, S. Chaudhuri, V. Ganti
Eliminating Fuzzy Duplicates in Data Warehouses.
Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002

[ACMM+99] S. Abiteboul, S. Cluet, T. Milo, P. Mogilevsky, J, Siméon, S. Zohar
Tools for Data Translation and Integration.
Bulletin of the Technical Committee on Data Engineering, March 1999, Vol. 22, No. 1, 3-8

[AIS93] R. Agrawal, T. Imielinski, A. Swami
Mining Association Rules between Sets of Items in Large Databases.
Proceedings of the ACM SIGMOD on Management of Data, 1993, 207-216

[AU79] A.V. Aho, J.D. Ullman
Principles of Compiler Design.
Addison-Wesley Publishing Company, 1979, ISBN 0-201-00022-9

[Bre99] S.E. Brenner
Errors in genome annotation
Trends in Genetics, 1999, 15, 4, 132-133

[EBRS+01] S. M. Embury, S. M. Brand, J. S. Robinson, I. Sutherland, F. A. Bisby, W. A. Gray, A. C. Jones, R. J. White
Adapting integrity enforcement techniques for data reconciliation
Information Systems, Vol. 26, 2001, 657-689

[EN00] R. Elmasri, S.B. Navathe,
Fundamentals of Database Systems.
Addison Wesley Pub Co., ISBN 0201542633

[Eng00] L. P. English
Plain English on Data Quality: Information Quality Management: The Next Frontier.
Published in DM Review in April 2000.

[GFSS00] H. Galhardas, D. Florescu, D. Shasha, E. Simon
AJAX: An extensible data cleaning tool.
Proceedings of the ACM SIGMOD on Management of data, Dallas, TX USA, May 2000

[GFSSS01a] H. Galhardas, D. Florescu, D. Shasha, E. Simon, C.-A. Saita
Improving data cleaning quality using a data lineage facility.
Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses, Interlaken, Switzerland, June 2001

[GFSSS01b] H. Galhardas, D. Florescu, D. Shasha, E. Simon, C.-A. Saita
Declarative data cleaning: Language, model, and algorithms.
Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

[HS95] M.A. Hernandez, S.J. Stolfo
The merge/purge problem for large databases.
Proceedings of the ACM SIGMOD Conference, 1995

[HSKK+01] L.M. Haas, P.M. Schwarz, P. Kodali, E. Kotlar, J.E. Rice, W.C. Swope
DiscoveryLink: A system for integrated access to life sciences data sources.
IBM Systems Journal, 2001, Vol. 40, No. 2, 489-511

[LLLK99] Mong Li Lee, Hongjun Lu, Tok Wang Ling, Yee Teng Ko
Cleansing data for mining and warehousing.
Proceedings of the 10th International Conference on Database and Expert Systems Applications, Florence, Italy, August 1999

[LLL00] Mong Li Lee, Tok Wang Ling, Wai Lup Low
IntelliClean: A knowledge-based intelligent data cleaner.
Proceedings of the ACM SIGKDD, Boston, USA, 2000

[LLL01] Wai Lup Low, Mong Li Lee and Tok Wang Ling
A knowledge-based approach for duplicate elimination in data cleaning
Information Systems, Vol. 26, 2001, 585-606

[ME97] A.E. Monge, C.P. Elkan
An efficient domain-independent algorithm for detecting approximately duplicate database tuples.
Proceedings of the SIGMOD 1997 workshop on data mining and knowledge discovery, 1997

[MM00] J.I. Maletic, A. Marcus
Data Cleansing: Beyond Integrity Analysis.
Proceedings of the Conference on Information Quality, October 2000

[Mot89] A. Motro
Integrity = Validity + Completeness.
ACM Transactions on Database Systems (TODS), Vol. 14, No. 4, 1989, 480-502

[MT99] Enric Mayol, Ernest Teniente
A Survey of Current Methods for Integrity Constraint Maintenance and View Updating.
ER Workshops 1999: 62-73

[Nau02] F. Naumann
Quality-Driven Query Answering for Integrated Information Systems
Lecture Notes in Computer Science, LNCS 2261, Springer, 2002

[Orr98] K. Orr
Data Quality and Systems Theory.
Communications of the ACM, Vol. 41, No. 2, 1998, 66-71

[RD00] E. Rahm, Hong Hai Do
Data Cleaning: Problems and current approaches.
IEEE Bulletin of the Technical Committee on Data Engineering, 2000, 24, 4

[Red98] T. Redman
The impact of Poor Data Quality on the Typical Enterprise.
Communications of the ACM, Vol. 41, No. 2, 1998, 79-82

[RH01] V. Raman, J.M. Hellerstein
Potter's Wheel: An Interactive Framework for Data Transformation and Cleaning.
Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

[Sch97] J.L. Schafer
Analysis of Incomplete Multivariate Data.
Chapmann & Hall, London, 1997

[SCS00] K.-U. Sattler, S. Conrad, G. Saake
Adding Conflict Resolution Features to a Query Language for Database Federations.
Proc. 3nd Int. Workshop on Engineering Federated Information Systems, Dublin, Ireland, 2000

[SS01] K.-U. Sattler, E. Schallehn
A Data Preparation Framework based on a Multidatabase Language.
International Database Engineering Applications Symposium (IDEAS), Grenoble, France, 2001

[SW81] T.F. Smith, M.S. Watermann
Identification of common molecular subsequences.
Journal of Molecular Biology, Vol. 147, 1981, 195-197

[Tan96] M.A. Tanner
Tools for Statistical Inference (third edition).
Springer, New York, 1996

[Ton00] D. Tonn
Data Cleansing Verfahren für Data Warehouses.
Diploma Thesis, Humboldt University Berlin, 2000

[VVSKS01] P. Vassiliadis, Z.a Vagena, S. Skiadopoulos, N. Karayannidis, T. Sellis
ARKTOS: towards the modeling, design, control and execution of ETL processes
Information Systems, Vol. 26, 2001, 537-561

[Win99] W.E. Winkler
State of Statistical Data Editing and Current Research Problems.
UN/ECE Work Session on Stat. Data Editing, Rome, Italy, 1999