Data Warehousing & Data Mining Seminar SS 2007

Fachsprache Englisch Seminar SS 2007

Data Quality and Data Cleaning in Data Warehouses

Institut für Wirtschaftsinformatik – Data & Knowledge Engineering

Johannes Kepler Universität Linz

&

Institut für Fachsprachen

Johannes Kepler Universität Linz

Supervision:

o. Univ.-Prof. Dr. Michael Schrefl

&

BSc (Hons) Alastair Long

Group 3

Mario	**Hofer**	**0255509**
Tobias	**Oberascher**	**0455334**
Alexander	**Sgardelli**	**0156384**
Markus	**Teufl**	**0255273**

Contents

Abstract

In times of huge data warehouses, both data cleaning and data quality considerably gain in importance. Therefore, the authors first discuss general problems and the problems related to poor data quality, and then continue by pointing out the relevant techniques to manage data quality, which is interrelated to the information provided on the data cleaning process. Poor data quality may implicate the existence of certain anomalies that need to be discovered, analysed and eliminated. Hence the authors give information on a selection of possible anomalies, including indicators and metrics, appropriate data cleaning methods and frameworks (that can be applied to remove the quality defect), as well as a comparison of important criteria. To start a conclusion, open questions are discussed and important information highlighted to retain facts in the reader's memory.

1 Introduction

The data quality and data cleaning is a major problem in data warehouses. Nowadays, the quality of data has become a main criteria for efficient databases. Data cleaning is an important approach to make sure that the data quality is at a high level.

At the beginning of this paper, an overview is given of what data quality means and how it can be managed. Therefore, we also show what impact poor data quality has on the data structure and on the data warehouse itself. In order to continue this thought, the motivation and effort to improve data quality is explained.

Anomalies in the data are the main reason for poor data quality. The data cleaning process has the goal to clean up the data of these anomalies.

The authors give a few examples of data anomalies affecting data quality. In detail, the following anomalies were chosen: lexical error, domain format error, irregularities, integrity constraint violation, contradiction, duplicates, invalid tuples, missing value and missing tuple. The anomalies will be explained and cleaning methods for each anomaly are presented and used in an example that is introduced in section 2.5.

In the section 5, data cleaning frameworks, a short overview of current approaches to data cleaning is presented and a comparison of five approaches is made. These cleaning approaches are, e.g., AJAX, FraQL or IntelliClean.

Finally, the authors are concerned with the open questions that could not be answered within this paper. Not for every anomaly exists a method to repair the records, or too little literature was released on this context so far.

2 Data Quality Management

2.1 Introduction

The importance of data quality has increased considerably during the last decade. More and more institutions, both private and public, as well as governmental use data warehouses to facilitate data processing at the various levels: operative, administrative and strategic.

On the one hand, data needs to be stored in databases that compose of a number of tables, views, etc. Information (describing parts of the real-world, called mini-world or universe of discourse) is represented as symbolic values that are subsumed as data throughout the remainder of this paper. By its nature, a table stores a number of tuples representing the mini-world's entities, including all the properties stored as single values that, altogether, represent one tuple, vice versa.

On the other hand, data warehouses compose of a number of instances of databases, which facilitates decision-support at all organisational levels. Data needs to be collected and processed both effectively and efficiently in order to support the data consumers. According to [22] "data consumers are people, groups, or systems that use data". Since data collection is often incumbent to humans (amongst other data producers, such as systems), data quality may suffer. According to [29], reasons for poor or dirty data are: "erroneous measurements, lazy input habits, omissions occurring while collecting and maintaining data", as well as "misinterpretations in data analysis or due to changes in the mini-world that are unnoticed or not reflected by changes to the representing data". However, data needs to be "fit for use by data consumers" [22].

In order to understand the importance of high quality data, it is necessary to provide some figures related to erroneous data: [35] claims that "unless an enterprise has made extraordinary efforts, it should expect data (field) error rates of approximately 1-5%. (Error rate = number of erred fields/number of total fields.)".

2.2 Impact of Poor Data Quality

As mentioned before, poor data quality may impede operations at various organisational levels and implies the existence of an anomaly (that "is a property of data values that renders them a wrong representation of the mini-world" [29]). [35] points out that the first obstacle is

to create awareness of this important issue. In addition, the author describes some typical impacts that result from reluctant data quality management.

2.2.1 Operative Impacts

Operations usually involve customers, cost related issues, and employees. First, customers may suffer by being named and/or addressed improperly, being sent the wrong products, and/or even being billed improperly. In turn, these circumstances directly lead to increased dissatisfaction because customers "are especially unforgiving of data errors" [35]. Second, estimates claim that 8-12% of the revenue is spent to discover and correct errors, or to put it into other relations, "40-60% of a service organization's expense may be consumed as a result of poor data" [35]. Hence operational costs are increasing considerably. Third, managers must also consider the employees' job satisfaction, which is directly influenced by poor data quality; e.g., one cannot expect data consumers to check the correct domain-format for all attributes every time they generate a query in order to receive customer details.

2.2.2 Administrative Impacts

Since the quality of a decision depends on the quality of the data used to obtain the decision, it is necessary to provide the quality needed in a cost-effective manner (higher quality data implies higher costs involved creating it, vice versa). Poor data quality, or even little uncertainty of its quality impedes the decision-making process, or, to put it into other words, hinders the data consumer to take advantage of the data provided by a data warehouse. In addition to the issues mentioned so far, administrative tasks (related to the provision of data) also "aim to put the right data in the right place at the right time to better serve a customer". But "one simply cannot serve customers when the data is not correct" [35]. Finally, and analogically to customer and employee dissatisfaction, poor data quality may lead to (perceived) mistrust within the organisation (e.g., misconceptions between departments).

2.2.3 Strategic Impacts

Since decisions at the strategic level have long-term impacts, many new external data sources need to be tapped. This increases uncertainty and hence lowers the quality of data. Furthermore, strategies need to be deployed and maintained in order to achieve corporate objectives. Maintenance is feasible only with the appropriate evaluation that involves new data or, to mention the flip side of the coin, new sources of possible errors. In addition, strategies need to be implemented, which involves operations and hence is directly related to effects of poor data quality management at the operational level.

2.3 Motivation and Effort to Improve Data Quality

So far, this paper has discussed general issues related to the impacts of poor data quality. It is about time to provide information concerning the benefits of subtle data quality management and its motivation.

"Data captures the enterprise's values and ideas" [35], and hence has to be managed conscientiously and with all the modern knowledge, methods and frameworks available. Any effort made directly leads to higher customer and employee satisfaction, lowered cost, improved decision-making (at all levels), increased organisational trust, and improved development and execution of strategies.

2.3.1 (Proactive) Data Quality Management

Generally, data quality management is incumbent to the data quality manager (DQM) whose activities address the issues concerning [35]:

- Data views, such as relevancy, granularity, and level of detail.
- Data values, such as accuracy, consistency, currency and completeness.
- Presentation of data, such as the appropriateness of the format, ease of interpretation, ease of understanding, etc.

Other issues are, for example, privacy, security, and ownership.

According to [16], the DQM is required to act proactive, applying the concept of total quality management to ensure high quality data in data warehouses. This implies to "focus on customer requirements, participation of all stakeholders as well as continuous improvement and a comprehensive management approach". Two major tasks that need to be coped with are quality planning and quality control.

On the one hand, quality planning involves collection of data consumer requirements and expectations, as well as transformation into the relevant processes and specifications. In addition, data quality dimensions (in regards to the next chapter) need to be established and prioritised.

On the other hand, quality control deals with verification of the data delivery processes that were identified during the planning phase. Quality control ensures compliance of all processes with the specifications that were agreed upon earlier. Furthermore, it is necessary to evaluate data quality in quantitative terms. Otherwise, continuous measurement of the current level of data quality is impossible and no quality measures can be taken.

Conducting these tasks may not be trivial. Therefore, it is necessary to utilise appropriate techniques and tools that support proactive data quality management. An appropriate (rule-based) concept that enables compliance with the requirements mentioned so far is suggested in [16] and illustrated in the following figure.

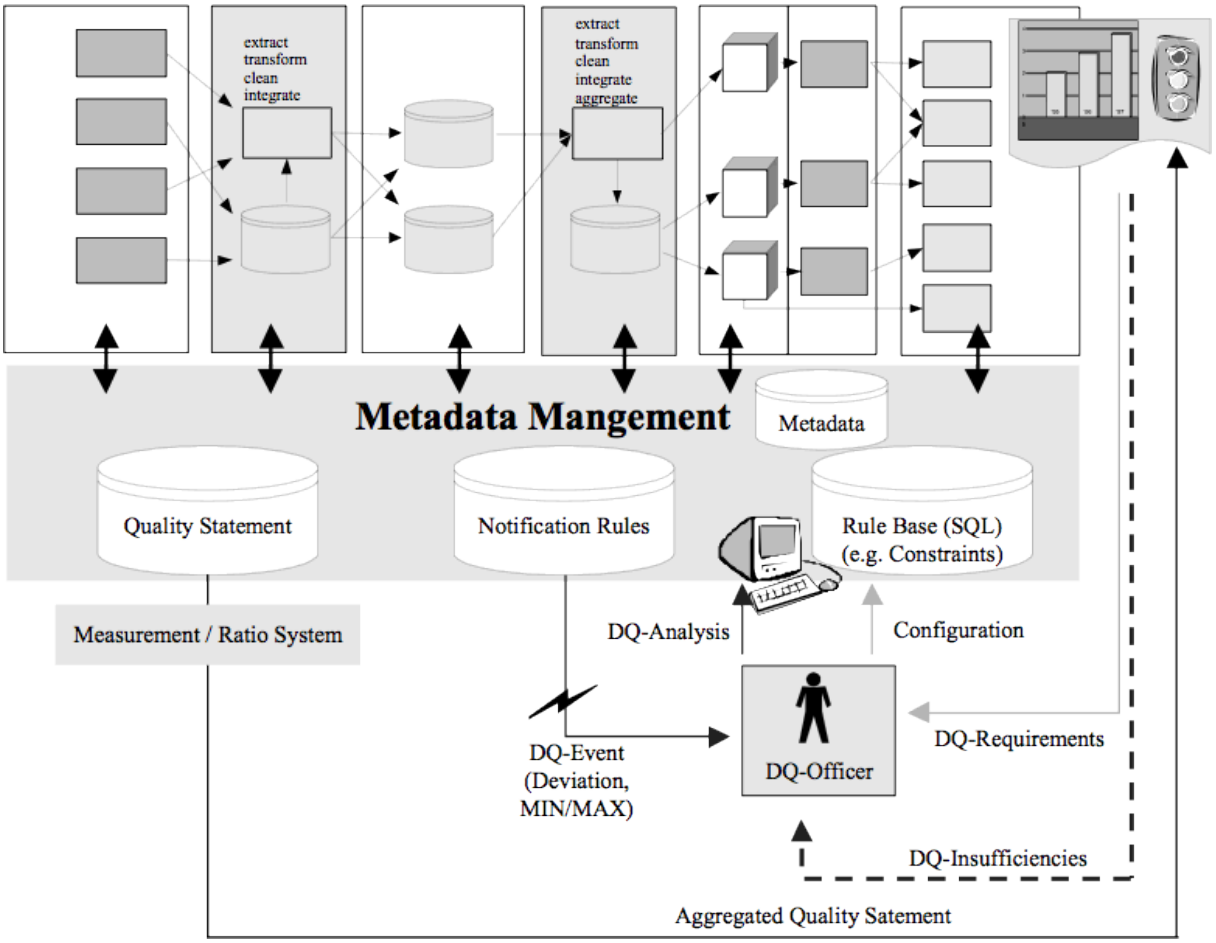

Figure 1: Architecture of the metadata based data quality system [16]

In order to work, a rule base for the evaluation of the current level of data quality in a data warehouse needs to be established. Depending on the notification rules, the DQM (or DQ-Officer) will be informed in case of irregularities (e.g., a constraint violation), and, therefore, plays a central role in this concept. Together with a quality statement base and all metadata defined for the data warehouse, these elements build up an "integrated metadata management component, which contains all important information about data quality" [16].

2.3.2 Dimensions of Data Quality

In order to deploy data quality improvement strategies successfully it is necessary to obtain an understanding of the dimensions involved when discussing data quality (or "'fitness of use',

which implies the concept of data quality is relative" [41]). The literature provides several dimensions that are widely accepted.

For example, [4] suggest the following four dimensions:

- Accuracy: having recorded the values correctly.
- Completeness: having recorded all relevant values.
- Consistency: having recorded the values according to the domain format.
- Timeliness: having recorded the values immediately after disposition.

A more recent classification of data quality dimensions was suggested in [46]. The authors conducted a survey to identify perceived data quality dimensions. As a result of this empirical approach, over 100 items had to be grouped in 20 dimensions, which were then further grouped into four categories:

Category	Dimensions
Intrinsic	Accuracy, Objectivity, Believability, Reputation
Accessibility	Access, Security
Contextual	Relevancy, Value-Added, Timeliness, Completeness, Amount of Data
Representational	Interpretability, Ease of Understanding, Concise Representation, Consistent Representation

Figure 2: Perceived Data Quality Dimensions [46]

An additional approach that will be referred to throughout the remainder of this paper is the definition of quality criteria in [27], [28], [42], which was also discussed in [29]. The authors suggest the following hierarchy:

Accuracy		
	Integrity	
		Completeness
		Validity
	Consistency	
		Schema conformance
		Uniformity
	Density	
Uniqueness		

Figure 3: Hierarchy of data quality criteria [29]

In order to describe each criterion, [29] introduced the following relational scheme:

Collection: *$S = (\{R\}, \Gamma)$, where $s \equiv r$; collection consists of only one instance*
Integrity: *$\gamma \square \Gamma$, true for all tuples in r; no enforcement of integrity constraints*
Mini-World: *M*

Accuracy: Aggregated value over integrity, consistency, and density

- Integrity: Aggregated value over completeness and validity
  - Completeness: Quotient of entities from M, represented by a tuple in r and total number of entities in M
  - Validity: Quotient of entities from M, validly represented by tuples in r and the total number of tuples in r
- Consistency: Aggregated value over schema conformance and uniformity
  - Schema conformance: Quotient of tuples in r conform to syntactical structure defined by R and total number of tuples in r (e.g., domain format $G(dom(A_i))$ of attributes A_1 to A_n in R
  - Uniformity: Quotient of attributes not containing irregularities (violation of val(r, Ai) for each Ai in R) and total number of attributes in r
- Density: Quotient of missing values in tuples of r and the number of total values (that are expected to be known)

Uniqueness: Quotient of tuples the same entity in M and total number of tuples in r

2.3.3 Model-Supported Data Quality Management

So far, the reader possesses the necessary understanding of the issues involved in the data quality process, knowledge of the impacts of poor data quality, as well as the dimensions of data quality criteria, which will facilitate discovery and elimination of anomalies. The following chapter will discuss two models that provide step-by-step instructions to successfully manage data quality in data warehouses.

The first model under review was introduced in [5], providing a systematical six-step guideline. In addition, the model pays attention to the well-known trade-off between the value of data availability and costs to obtain and store the data. The model is applied as follows:

1. Determination of the supported activities within an organisation.
2. Identification of all relevant and required data for each activity specified in step 1.
3. Evaluation of the data quality (using the dimensions introduced earlier) for each set of data specified in step 2.
4. Identification of the managerial actions (e.g., projects, surveys, etc.) that can be taken in order to improve data quality, and the associated costs.
5. Estimation of the possible impacts of actions specified in step 4 on data quality.
6. Definition of the changes in warehouse utility in case action is taken.

This model has been formulated in [5], but has been hardly used in real-world situations.

The second model under review was suggested in [18] and complies with ISO 9001:2000 [20]. It provides a concept called CLIQ that can be applied to data integration projects in data warehouses. In order to merely obtain an idea of CLIQ, it is not necessary to provide more detail concerning the standard or the concept itself. The ten steps suggested are illustrated in the figure below.

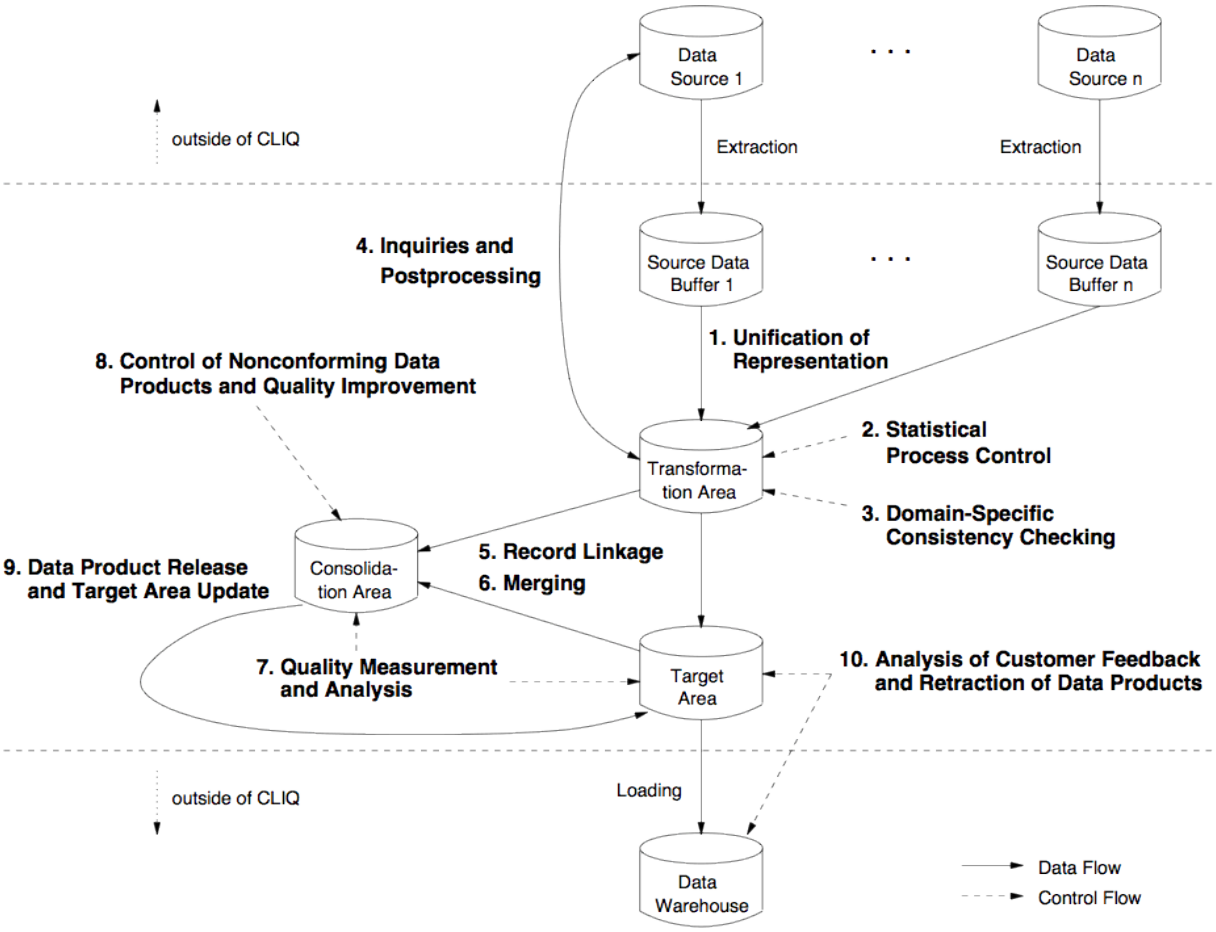

Figure 4: Operational Steps of Data Integration in CLIQ [18]

Due to compliance with ISO 9001:2000 this model should be widely accepted and applied to real-world projects.

2.4 Conclusion

In order to ensure high quality data it is necessary to raise awareness of the importance of all the issues involved in this topic. This task, among other duties, such as the definition of all relevant data quality dimensions, as well as the appliance of a preferred model to put plans into action is incumbent to the DQM who, in the best case, manages proactively. Altogether, a sophisticated DQM system will prevent negative impacts owing to poor data quality on the various organisational levels, as discussed in this chapter's introductory part.

2.5 Introduction of a Uniform Database

In order to facilitate the understanding of all processes involved in the data quality management and cleaning processes, it is necessary to introduce a uniform database "Mail-Order" that will be used throughout the remainder of this paper.

Employee (ENO, ENAME, *ZIP*, HDATE)
Part (PNO, PNAME, QOH, PRICE, LEVEL)
Customer (CNO, CNAME, STREET, *ZIP*, PHONE)
Order (ONO, *CNO*, *ENO*, RECEIVED, SHIPPED)
ODetail (*ONO*, *PNO*, QTY)
ZipCode (ZIP, CITY)

Employee

ENO	ENAME	ZIP	HDATE
1000	Jones	67226	12-DEC-95
1001	Smith	60606	01-JAN-92
1002	Brown	50302	01-SEP-94

Figure 5: Mail-Order Database – Table Employee

Part

PNO	PNAME	QOH	PRICE	LEVEL
10506	Land Before Time I	200	19.99	20
10507	Land Before Time II	156	19.99	20
10800	Dirty Harry	140	14.99	30

Figure 6: Mail-Order Database – Table Part

Customer

CNO	CNAME	STREET	ZIP	PHONE
1111	Charles	123 Main St.	67226	316-636-5555
2222	Bertram	237 Ash Ave.	67226	316-689-5555
3333	Barbara	111 Inwood St.	60606	316-111-1234

Figure 7: Mail-Order Database – Table Customer

Order

ONO	CNO	ENO	RECEIVED	SHIPPED
1020	1111	1000	10-DEC-94	12-DEC-94
1021	1111	1000	12-JAN-95	15-JAN-95
1022	2222	1001	13-FEB-95	20-FEB-95
1023	3333	1000	20-JUN-97	Null

Figure 8: Mail-Order Database – Table Order

Odetails

ONO	PNO	QTY
1020	10506	1
1020	10507	1
1023	10800	2

Figure 9: Mail-Order Database – Table Odetails

ZipCode

ZIP	CITY
67226	Wichita
60606	Fort Dodge
50302	Kansas City
54444	Columbia
66002	Liberal
61111	Fort Hays

Figure 10: Mail-Order Database – Table ZipCode

3 Data Cleaning Process

The overall goal of the data cleaning process is to clean-up the data of anomalies. The process itself cannot be fully automatic. There are always decisions which have to be made by human experts who overview the process. In general the tasks can be combined with the data acquisition for the data warehouse. [29] says that the data cleaning process never ends because there can always occur new anomalies when you correct the found ones. But it is a question of how much money a company wants to spend to ensure the data quality. The better the data quality should be, the more cycles of the data cleaning process will be needed.

[29] and [31] describe the four main parts of the data cleaning process as shown in the figure below.

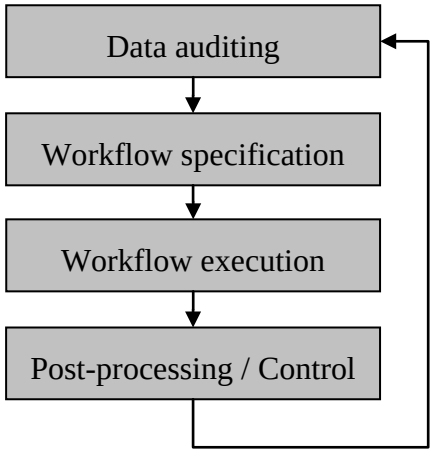

Figure 11: Data Cleaning Process [29]

3.1 Data auditing

This first part of the data cleaning process has to analyse the data to identify which kinds of anomalies occur in the data set. Data auditing is mostly a semi-automatic process and contains

- data profiling (analysis of single attributes of a tuple) and
- data mining (analysis of the whole data collection with the goal to identify anomalies)

As a result of this first part the data cleaning experts should know which anomalies are hidden in the data provided and which kind of form they have. Integrity constraints and domain formats can be specified after the data auditing process is finished. Furthermore, this part of the data cleaning process should identify characteristics within the data set to help correcting

the anomalies later [31]. For example, data mining algorithms can show that there exists the following rule within the Mail-Order Database.

"RECEIVED-Date < SHIPPED-Date"

A confidence of 99% of the rule above means that 1% of the orders in the mail-order database could be anomalies and need to be checked.

3.2 Workflow specification

The second part of the data cleaning process specifies step by step which operations have to be performed to clean the data. Together, these steps form the data cleaning workflow. The goal of the workflow specification is to describe the workflow as good as possible in a declarative way and ensure that the anomalies can be corrected automatically.

One way to define operations that can help eliminate anomalies is User Defined Functions (UDF). Supported by the standard query language SQL:99 [31].

Let's assume that there is a domain format anomaly in the Mail-Order Database in the Table Employee (Figure 12: Employee table with a data format anomaly). The attribute "HDate" contains the hire date of the employee but is not saved with the expected date format. A user defined function "TransformDate" can be used to fix this anomaly.

Employee

ENO	ENAME	ZIP	HDATE
1000	Jones	67226	12/DEC/95

Figure 12: Employee table with a data format anomaly

E.g. Update Employee Set HDATE = TransformDate(HDATE);

Employee

ENO	ENAME	ZIP	HDATE
1000	Jones	67226	12-DECEMBER-1995

Figure 13: Employee table without an anomaly

The simple example of an UDF above should show that for each individual problem experts or the system have to choose the right method to solve it and this difficult task is performed during the workflow specification. [31] recommend testing and evaluating the specified workflow because not only must the workflow be correct it should be effective too.

3.3 Workflow execution

Having designed and tested the workflow, it will be executed. During this part of the data cleaning process it is often necessary that the cleaning experts make decisions on the fly. There will always be tuples that cannot be checked by the cleaning process automatically. Thus an expert must decide if such a tuple should be corrected, deleted or even left unchanged. Expert knowledge is expensive and each tuple which has to be rechecked costs time and a lot of money [29].

3.4 Post-processing / Control

The last part of the data cleaning process verifies the executed workflow. Tuples which could not be corrected are checked manually again. Good systems have learned from the decisions made by cleaning experts and may adjust the intern knowledge base now. A new cycle of the data cleaning process begins. The data is audited again and typical characteristics are searched. A new workflow will be specified, tested and executed [29].

4 Classification of Anomalies

4.1 Introduction

In order to illustrate the anomalies that may occur in a data warehouse, the authors use the following examples, based on the database introduced in 2, and in alignment with the data quality dimensions presented in the same chapter. The following figure illustrates the interrelation between these quality criteria and the anomalies that impede achievement of a certain level of quality.

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

Figure 14: Data anomalies affecting data quality criteria [29]

„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“ [29].

4.2 Lexical Error

4.2.1 Introduction

One of the most common anomalies that occur in data warehouses is represented by the so-called lexical error. From time to time, values may not be available when data is transferred to the data warehouse. As a result, some attributes may not be completely defined. Therefore, tuples within a table do not entirely describe the real-world object with all its attributes.

According to [29] lexical errors are defined as “discrepancies between the structure of data items and the specified format”. This means that the expected number of values for a tuple is violated (i.e., the degree of a tuple #t diverges from the expected degree of the relational schema #R).

According to the paper’s uniform database, table Employee will be used to illustrate this anomaly.

Employee

ENO	ENAME	ZIP	HDATE
1000	Jones	67226	12-DEC-95
1001	Smith	01-JAN-92	

Figure 15: Table Employee – Lexical errors

Every tuple is expected to consist of 4 values, one for each attribute. In this case, a missing ZIP value caused a shift of the HDATE value to the ZIP column. This may occur in case an employee’s ZIP code is unavailable when data is transferred to the data warehouse (e.g., a new employee was hired recently and not all personal data was collected by the Human Resource Department so far).

4.2.2 Data Cleaning Methods

Due to the fact that lexical and domain format errors are very similar and subsumed as syntactical errors, methods to discover and eliminate lexical errors are integrated in those that can be applied to domain format error cleaning. Therefore, this chapter directly refers to the methods and tools presented in the next chapter.

4.2.3 Conclusion

The first anomaly under review is expected to occur frequently for several reasons that were discussed earlier in this paper. Methods to discover and eliminate this error are highlighted later due to the similarity to domain format errors.

4.3 Domain Format Error

4.3.1 Introduction

All data stored in a table needs to conform to the domain specified for each attribute G(dom(A)) in a tuple. Domain format errors occur frequently and are caused by several reasons, e.g., negligent data entry by an employee. Again, table Employee will be used to discuss this issue.

Employee

ENO	ENAME	ZIP	HDATE
1000	Jones	67226	12/DEC/95

Figure 16: Table Employee – Domain Format Error

At first sight, one cannot tell the existence of an anomaly in the table populated above. Therefore, it is necessary to define the domain of each attribute as follows:

ENO: $\sum_D{}^4$
ENAME: $\sum_D{}^*$
ZIP: $\sum_D{}^5$
HDATE: $\sum_D{}^2\text{-}\sum_D{}^3\text{-}\sum_D{}^*$

ENO, ENAME, and ZIP satisfy the anticipated domain format, whereas HDATE violates the expected format for this attribute. It is meant to be stored as a hyphen-separated token.

4.3.2 Data Cleaning Methods

As discussed in [37] common cleaning operations for this type of anomaly comprise format & domain transformation, standardisation, normalisation, and dictionary look-up. These operations are performed by both individual parsers and separated cleaning modules. The latter usually combine a number of different approaches and sometimes provide data profiling mechanisms.

4.3.2.1 Metadata

Metadata plays an important role in application of the methods (and the appropriate tools) discussed in this section. Without the definition of the domain in the metadata (or design repository), it is impossible to evaluate sets of data regarding its correctness. Tools need "schema descriptions, libraries of transformation rules, descriptions of transformation scripts, etc." [6]. Tools that use dictionary look-up, for example, investigate if a certain value is in line with the expected domain range defined in the design repository.

A proprietary facility to support (meta)data transformation is Microsoft Repository that is shipped with Microsoft SQL Server. This allows easier management of metadata and offers an XML-option to facilitate storage and exchange of highly structured data. To work, "rules must be defined and implemented to identify and resolve inconsistent data formats, missing data, invalid data, semantic inconsistencies, and other sources of poor data quality" [6].

4.3.2.2 Data Profiling

As mentioned before, data profiling is a possible technique to reveal and eliminate syntactical errors. It describes the process of examining all data stored in the database in order to produce statistics and information about the purpose, quality, risks, metadata and other issues related to the examined set of data.

A modern tool to profile and automatically clean data is shipped with Oracle Warehouse Builder 10g Release 2. Three important features allow easy elimination of a bunch of anomalies, including syntactical errors [36]:

1. Graphical Data Profiler: helps to understand the structure, semantics, content, anomalies, and outliers present in the data, and derives rules that will later be used within the data warehouse.

2. Correction Wizard: takes the data rules defined during the profiling process, applies them to the data and automatically generates correction mappings to clean and transform the data.

3. Data Auditor: takes data rules and monitors quality of subsequent loads.

First, a data profile needs to be created using the Graphical Data Profiler. Users can easily profile the stored data and identify anomalies. The domain tab incorporated in the Data Profile Editor, for example, "shows for each column in the profiled object the suggested domain, and the degree to which the data is compliant with this domain" [36]:

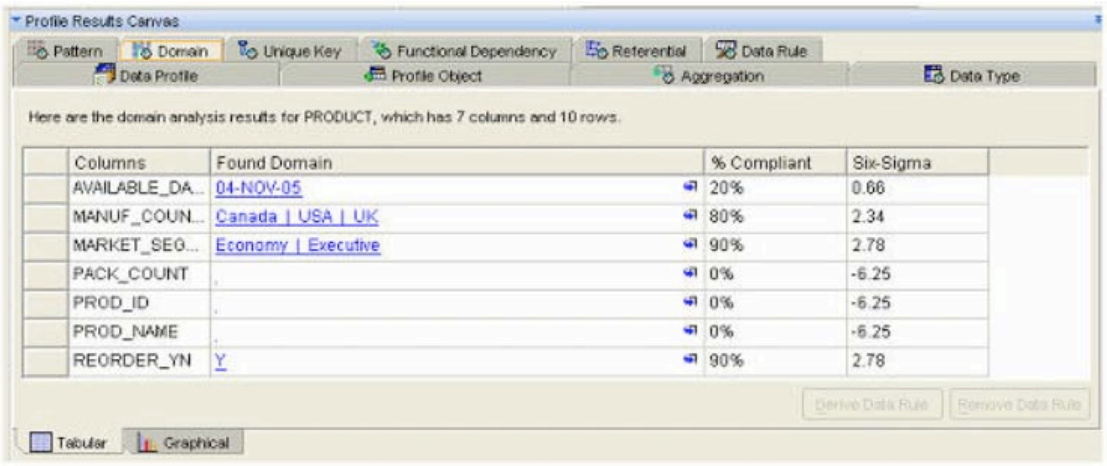

Figure 17: Oracle Warehouse Builder 10g Release 2 – Domain Tab [36]

The tool automatically identifies possible domains and derives rules that are later applied to the data (using the Correction Wizard). “Oracle Warehouse Builder will then implement this rule as a check constraint on the column, enforcing the data rule at the database level to ensure it is never broken”. All metadata that is generated about the expected “data quality is stored alongside the other metadata” in the design repository [36].

Second, once all rules are derived and stored in the design repository, it is about time to eliminate the anomalies. Therefore, the Correction Wizard offers three options that can be selected in case a rule is violated. The user may choose between “Ignore” (no action taken), “Report” (log entry of the one instance that violated a constraint), and “Cleanse”. The latter option accommodates four cleaning strategies: “Remove” (domain correction by excluding the affected rows), “Similarity Match” (domain correction using the built-in Match-Merge functionality), “Soundex Match” (domain correction using the built-in SOUNDEX function [26] to “change erroneous values to the one within the domain that is the closest phonetic match to it”), and “Custom” (that is “used where the logic to clean is more complex”) [36].

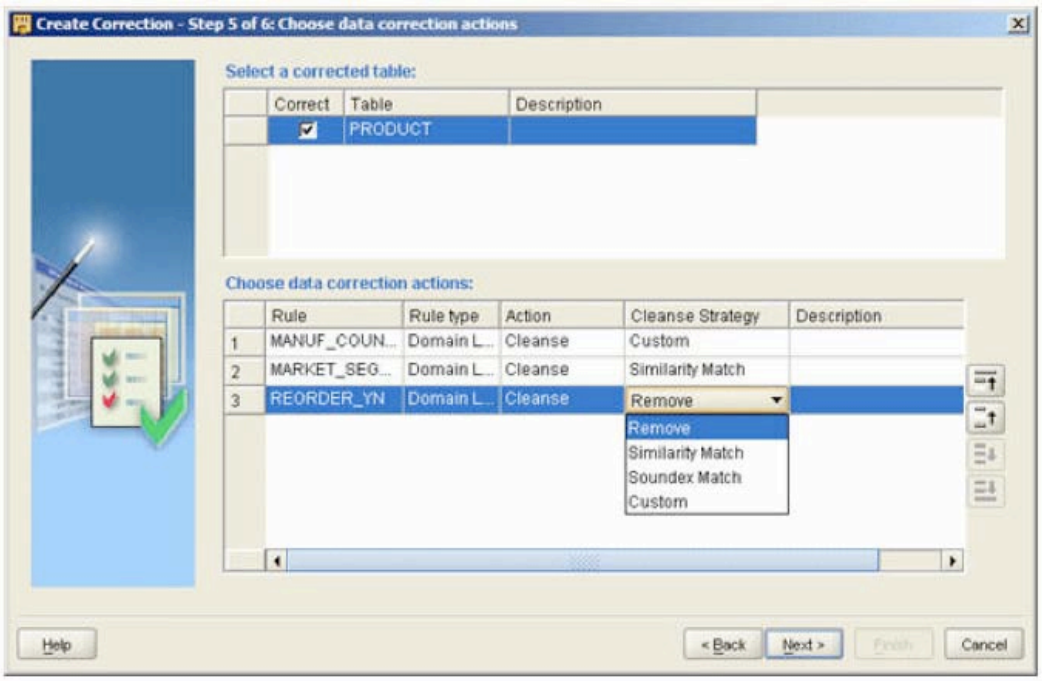

Figure 18: Oracle Warehouse Builder 10g Release 2 – Correction Wizard [36]

4.3.3 Conclusion

Similar to lexical errors, domain format errors occur for several reasons (e.g., heterogenic databases are merged and transformation causes format conflicts). Due to the fact that data warehousing is a growing segment of the database business, many frameworks (that apply certain methods) are available. This section discussed two proprietary systems that facilitate data cleaning (by using metadata that provides a rule-base which allows implementation of the appropriate constraints). These systems apply different methods that may also be used to clean other types of anomalies (e.g., SOUNDEX [26] allows elimination of both domain format errors and duplicates).

4.4 Irregularities

4.4.1 Introduction

According to [29], *"irregularities are concerned with the non-uniform use of values, units and abbreviations"*. This anomaly often arises from the process of merging data from multiple sources, e.g. if a multinational enterprise uses operational data from local stores to fill a data warehouse. It may be the case that there are shops in Austria where the local prices are denoted in Euros, whereas the Japanese stores denote the prices in Yen. Further examples of irregularities, described by [38], would be, if there are some shops which store prices

including the VAT and some shops which exclude the VAT or if an international automaker stores mileage data both in kilometres and miles.

The examples from the previous section show, that it is very difficult to define the term "irregularity" in the context of data cleaning. This anomaly rather consists of a list of small problems where both, the source and the solution, can be quite different amongst the problems. Therefore, this paper will focus on the case of different currencies in different databases.

Figure 19: Table Part – Irregularity I shows two records from the "Part" table. Even though both records describe a DVD, it seems that the second DVD costs 3,269.11 Euros. An additional CURRENCY attribute may avoid misinterpretation of values, originally stored in a certain currency, e.g., the first value stored in PRICE is supposed to represent a Euro-value, whereas the second value stored in PRICE is supposed to represent a Yen-value.

Part

PNO	PNAME	QOH	PRICE	LEVEL
10506	Land Before Time I	200	19.99	20
10507	Land Before Time II	156	3,269.11	20

Figure 19: Table Part – Irregularity I

4.4.2 Data Cleaning Methods

Basically there are two ways to tackle the problems caused by irregularities. Firstly, one can try to prevent the occurrence of irregularities. In order to achieve this [31] and [38] suggest to incorporate data mining techniques like clustering, summarization, regression analysis and sequence discovery in the design process of a data warehouse. Secondly, methods like outlier analysis, as shown by [3], can be used to detect outliers after the process of joining data from multiple sources.

In order not to go beyond the scope of this paper, only the following two methods will be discussed:

- Regression Analysis
- Outlier Analysis

4.4.2.1 Regression Analysis

Like mentioned before, regression analysis can be applied in advance to the data joining process in order to prevent irregularities. [38] suggests to regress the prices from one database on the prices from the other database. If there are systematic differences in the prices, the regression analysis will yield statistically significant coefficients.

There exists a vast amount of literature on regression analysis like [9], [47], [48] and [33]. To demonstrate how regression analysis can be applied to the example database given in this paper, the method of OLS, as described in [9] suffices.

In order to test whether there are systematic differences in the prices from the two different databases one can use OLS to carry out the following regression:

$$p_E = \beta_0 + \beta_1 p_J$$

where p_E denotes the prices from the Austrian shops and p_J denotes the prices from the Japanese stores. If it is true that p_E represents Euro-values whereas p_J represents Yen-values, then β_0 is expected to be 0 and the coefficient β_1 would yield the Euro/Yen exchange rate.

The same procedure can be applied to the case where the prices of one database include VAT whilst the prices of the other database do not include the VAT.

4.4.2.2 Outlier Analysis

The following section explains an outlier detection algorithm introduced by [3]. The algorithm tries to mimic the human brain to detect outliers, by filtering elements which do not fit into recognized series of similar items. Compared to other methods of outlier detection [3] *"approach the problem from inside of the data, using the implicit redundancy of the data"*. That means that the algorithm does not rely on outside information, like integrity constraints, in order to detect deviations.

The following section will explain the algorithm by:

- introducing an exemplary dataset,
- explaining the main steps of the algorithm and finally
- comments and thoughts concerning the algorithm
- applying the algorithm to the exemplary dataset

The Exemplary Dataset

Figure 20: Exemplary Data for Outlier Detection shows a list of DVD prices retrieved from the "Part" table after joining the databases from Austria and Japan. The prices of DVDs are expected to be rather similar. Nevertheless there are two outliers in the sample, namely 3,269.11 and 3,761.33[1].

Prices	19.99	19.99	3,269.11	24.99	24.99	22.99	3,761.33	23.99

Figure 20: Exemplary Data for Outlier Detection

Main Steps Of The Algorithm

Step 1: Read the first element from the input stream. This element constitutes the subset I_1.

Step 2: Build I_j as the sum of I_{j-1} and the next element from the input stream and calculate the dissimilarity value $D(I_j)$[2].

Step 3: Calculate the difference in dissimilarity values. $d_j = D(I_j) – D(I_{j-1})$.

Step 4: Repeat Steps 2 and 3 until I_j includes all items from the input stream

Step 5: Find the element i_j which corresponds to the maximum positive difference in dissimilarities d_j and consider it to be a sequence exception. If there is no positive difference in dissimilarities, then there is no exception.

Step 6: For each i_k with $k>j$[3] calculate:

$d_{k0} = D(I_{j-1} \cup \{i_k\}) - D(I_{j-1})$

$d_{k1} = D(I_j \cup \{i_k\}) - D(I_j)$

[3] state that i_k should be added to the set of exceptions, if the condition $d_{k0} – d_{k1} > d_j$ holds.

[1] 3,269.11 and 3,761.33 are the equivalent Yen-values of €19.99 and respectively €22.99 at the current exchange rate.

[2] D(I), the so called dissimilarity function, can be any function that returns a low value for highly similar items and a high value for dissimilar items. An example would be the variance of the set items.

[3] With j being the index of the element which has been identified as the sequence exception in step 5.

This step was introduced to cover the degenerate case, where the item i_j has been identified as a candidate exception and assume that i_{j+1} is identical to i_j. [3] claim that *"in that case, $D(I_{j+1}) - D(I_j)$ will be 0 and i_{j+1} will not be considered an exception"*.

Comments and thoughts concerning the algorithm

In order to keep the example and the algorithm simple, a final step, proposed by [3], has been left out. They suggest, going through steps 1 to 6 *m* times to get *m* competing potential sets of exceptions. The final step would be to *"select the one with largest value of difference in dissimilarity dj, scaled with the cardinality function C"*[4].

Furthermore step 6 as proposed by [3] seems to be unfeasible. To show this, one can plug in the equations for d_{k0} and d_{k1} into the condition. Rearranging the terms yields the following equation:

$$D(I_{j-1} \cup \{i_k\}) - D(I_j \cup \{i_k\}) > d_j - (D(I_j) - D(I_{j-1}))$$

Taking a closer look at the term in brackets on the right hand side reveals that it is equal to d_j[5]. Therefore the right hand side of above equation drops out and the condition can be restated as:

$$D(I_{j-1} \cup \{i_k\}) > D(I_j \cup \{i_k\})$$

One has to keep in mind that [3] introduced this condition in order to cover the degenerate case where i_j and i_{j+1} respectively i_k are exceptions and both have the same value. Therefore it is true to state the condition as $D(I_j) > D(I_j \cup \{i_k\})$. This condition requires that the inclusion of the potential exception i_k lead to a reduction in dissimilarity, which seems rather doubtful.

Applying the algorithm to the exemplary dataset

The application of the algorithm to the exemplary data can be seen in Figure 21: Applying the algorithm to the exemplary data. The columns I1 to I8 represent the subsets of the list of prices. The row "Dissimilarity values" shows the variance for each subset (the data so far corresponds to the steps 1 and 2 of the algorithm). Beneath the dissimilarity values there are the differences in dissimilarity values (d_j). The differences in dissimilarity values are calculated as described in step 3. Repeating steps 2 and 3 for each subset, i.e. for I1, I2,…, I8 yields the upper part of the table shown in Figure 21: Applying the algorithm to the exemplary data.

[4] A simple cardinality function would be to count the number of items in the set.

[5] Cf. $d_j = D(I_j) - D(I_{j-1})$ from step 3 of the algorithm.

Comparing the differences in dissimilarity values for all subsets shows that its maximum of 2345951,28 corresponds to an item value of 3269,11, which definitely is an exception to the series of prices. The affected subset I3 is marked grey.

In order to proceed with step 6, one has to calculate d_{k0} and d_{k1}. This is done in the bottom half of Figure 21: Applying the algorithm to the exemplary data. The calculations are done according to the formulas given by [3]. As a final step, the authors of this paper suggest, to add each element to the list of sequence exceptions, for which the condition $\max(d_{k0},d_{k1}) > d_j$ holds. This condition is satisfied if Item 7 is added to either I_j or I_{j-1}. Therefore the final list of sequence exceptions is {3269.11; 3761.33}.

Based on exemplary data on DVD prices, a simple method for deviation detection has been used to filter out currency irregularities. The algorithm does not only work for single items like a price. For example it can also be applied to complete records of a database. The only thing which has to be adopted is the calculation of the dissimilarity values[6].

[6] A discussion on dissimilartiy functions would be beyond the scope of this text and is therefore omitted. For further details see [3].

Set of Items	*I1*	*I2*	*I3*	*I4*	*I5*	*I6*	*I7*	*I8*
Item 1	19,99	19,99	19,99	19,99	19,99	19,99	19,99	19,99
Item 2		19,99	19,99	19,99	19,99	19,99	19,99	19,99
Item 3			**3269,11**	3269,11	3269,11	3269,11	3269,11	3269,11
Item 4				24,99	24,99	24,99	24,99	24,99
Item 5					24,99	24,99	24,99	24,99
Item 6						22,99	22,99	22,99
Item 7							3761,33	3761,33
Item 8								23,99
Dissimilarity Values	0	0	2345951,28	1977370,38	1686491,63	1463878,1	2506791,87	2302052,35
dj	0	0	**2345951,28**	-368580,901	-290878,755	-222613,524	-1042913,76	-204739,514

Items of Ij-1								
Item 1				19,99	19,99	19,99	19,99	19,99
Item 2				19,99	19,99	19,99	19,99	19,99
ik (k>j)				24,99	24,99	22,99	3761,33	23,99
D(Ij-1 + ik)				5,55555556	5,55555556	2	3110583,33	3,55555556
D(Ij-1)				0	0	0	0	0
dk0				5,55555556	5,55555556	2	**3110583,33**	3,55555556
Items of Ij								
Item 1				19,99	19,99	19,99	19,99	19,99
Item 2				19,99	19,99	19,99	19,99	19,99
Item 3				3269,11	3269,11	3269,11	3269,11	3269,11
ik (k>j)				24,99	24,99	22,99	3761,33	23,99
D(Ij + ik)				1977370,38	1977370,38	1978179,66	3084443,25	1977774,84
D(Ij)				2345951,28	2345951,28	2345951,28	2345951,28	2345951,28
dk1				-368580,901	-368580,901	-367771,621	**738491,971**	-368176,448

Figure 21: Applying the algorithm to the exemplary data

4.4.3 Conclusion

As shown in the previous section, there is no thing such as “the irregularity anomaly”. It is rather a wide-ranged area containing several conflicts which can either stem from the same source or result in the same problems. Therefore it is not possible to find a generic method to prevent, detect or repair inconsistencies and problems caused by irregularities.

Both the regression analysis and the outlier analysis, as they are presented in this section, are only applicable to a specific problem. Therefore it is hard, if not impossible to put those methods on equal footing in order to compare advantages and drawbacks.

4.5 Integrity Constraint Violation

4.5.1 Introduction

[15] define integrity constraints as *“a predicate or query such that if the predicate holds on a state of the data, or equivalently if the query produces an empty answer, then the database is considered valid”*. Based on this definition it is obvious that the anomaly described in this section, deals with implications if such constraints are violated.

[38] classify three types of integrity constraints:

- Integrity constraints as domains of attributes (e.g., weight of a car must be between 1000kg and 2000kg).
- Constraints on different attributes (e.g., the data received of an order must be before the date shipped)
- Inter relation integrity constraints (e.g., foreign key relations)

Papers like [31], [38], [29] agree on the fact that integrity constraint violations are often a result of a lack of appropriate model specification and poor schema design. [31] also mention that integrity constraints cause an overhead for integrity control. Therefore such constraints are often defined in a rather loose way.

Figure 22: Table Part – Integrity Constraint Violation shows an example of a violated integrity constraint as a domain of an attribute. In this example there exists a product with a negative price. A reason for this violation may be that the designers of the database or data warehouse did not implement a constraint considering the range of valid prices.

Part

PNO	PNAME	QOH	PRICE	LEVEL
10506	Land Before Time I	200	-19.99	20

Figure 22: Table Part – Integrity Constraint Violation

A negative value stored in PRICE violates integrity constraints. Furthermore it could be the case that the "Part" table shown in Figure 22: Table Part – Integrity Constraint Violation is from a data warehouse which has been filled with data from multiple sources. If such a source is not a database system, e.g. if the data warehouse also gathers information represented in text files it is highly likely that such sources do not provide mechanisms for integrity checking.

4.5.2 Data Cleaning Methods

Similar to the case of irregularities it is hard, if not impossible to define a single phenomenon called "the integrity constraint violation anomaly". Rather there different approaches in the different steps of the process to maintain a valid and consistent database.

[38] and [31] claim that an important source of integrity constraint violations in a data warehouse is the fact that the data for the warehouse often is extracted from a larger number of multiple sources which probably different schemas and/or structural conflicts. Therefore both papers suggest incorporating data mining techniques in the design process of a data warehouse.

Furthermore, authors like [11] argue that integrity checking and simply refusing an update, insert or delete transaction if the check fails, is "*hopelessly inappropriate for dealing with newer, less traditional database applications, particularly in which transactions are of long duration, and high complexity*". Therefore data cleansing must not only provide methods for integrity checking, but also for integrity maintenance.

The previous paragraphs show that the following problems and task arise in the context of integrity constraint violations in data warehouses:

- Gathering and generating constraints, given the data in the data warehouse
- Integrity Checking
- Integrity Maintenance

4.5.2.1 Generating Integrity Constraints

On the basis of the classification of integrity constraints by [38] there are several methods how integrity constraints can be gathered and generated.

In order to generate integrity constraints using a given stock of information [38] suggest to employ methods like cluster analysis or distribution analysis. A simple scatter plot can visualize potential domains of attributes. Furthermore visualization methods can help to achieve the same goal.

[38] also argue that data mining techniques *"can find intervals of attribute values which are rather compact and cover a high percentage of the existing values"*, e.g., it may be that in the DVD database in 99% of the cases the price of a DVD is between €10 and €30. [38] claim that such analysis does not only identify potential integrity constraints, it also gives hints about noise in the data.

On the other hand, constraints on different attributes respectively inter relation integrity constraints can be obtained by discovering functional and approximate dependencies in the database. This is done by algorithms like TANE ([19]), which finds functional dependencies from large databases. TANE is an efficient algorithm which *"is based on partitioning the set of rows with respect to their attribute values, which makes testing the validity of functional dependencies fast even for a large number of tuples*"[7].

Another algorithm to discover functional or inclusion dependencies is explained by [21]. Their algorithm is built to discover *"inclusion dependencies across high-dimensional relations in the order of 100 attributes"*.

4.5.2.2 Integrity Checking

Although many database systems enable the administrator to formulate integrity constraints in a SQL type of language, [15] mention, that the problem arises from the fact that integrity checking is costly, especially in the case of a distributed database.

In their paper [15] want to insert a tuple t into a relation L under the assumption that a set of constraints C hold before the insertion without using information from the inaccessible relation R. For example it could be that the employees of a department must not earn more than the manager of that certain department. Suppose there are the two relations emp and dept. Where emp contains data on the employees (name, social security number, salary, etc.) and dept contains the department number and the salary for the manager of that department.

[7] A detailed discussion on TANE is beyond the scope of the paper and therefore left out.

Furthermore assume that the emp relation is accessible, whereas the dept relation is stored on a database in an other country and is therefore inaccessible.

[15] try to transform the constraint into a new constraint which can be checked against the accessible data. The result of that new constraint should be either a clear “yes, the constraint will hold” or a “the constraint may not hold for the complete data”. The constraint solving this problem would be to check whether the salary of the new employee which has to be inserted into the database is below any other employee’s salary of that department.

As mentioned before, it is assumed, that the database is in a consistent state before the insert is carried out. For this reason one can conclude if the constraint was satisfied before the new employee has been inserted and if the salary of the new employee is below the salary of any other employee of that department, the constraint will also be satisfied after the insert.

4.5.3 Conclusion

Similar to the irregularities anomaly the previous paragraphs have shown that the area of integrity constraint violation is diversified. Therefore one cannot numerate a list of methods which can be compared on the grounds of how well they deal with the problems caused by integrity constraint violations.

4.6 Duplicates

4.6.1 Introduction

Many authors like [31], [10] or [23] mention that duplicate entities in data warehouses are a serious problem. But what are duplicates and what is the goal of duplicate detection?

Firstly, it has to be said that the problem itself is described due different terms and is known for more than five decades in the statistics field. Some terms for duplicate anomalies and also for the process of duplicate detection are

- record linkage (this term is especially use in the statistic research field)
- record matching
- merge-purge problem
- instance identification
- data deduplication or

- name matching

There are even more terms which describe this problem or aspects of it. In this chapter the term duplicate anomaly is used to describe the problem itself and the term duplicate detection and elimination to describe the process of finding and correcting duplicate anomalies [10].

Secondly as it is defined now how we call the problem we try to give a brief overview about what a duplicate anomaly means.

The database represents a mini-world. It consists of tuples which represent an entity in the real world e.g. a DVD-film is stored in the database as a tuple with five attributes product number, product name, quantity, price and level as shown in Figure 23: Table Part – Duplicates. If there are two or more tuples in the database which represent the same real world entity, this is called a duplicate anomaly. The values of the tuples i.e. the value of each attribute of the tuples must not be the same. Some or even all values can be different and it could still be a duplicate anomaly [29]. Duplicate anomalies are often caused during the process of integrating different data sources into a single data warehouse.

Figure 23: Table Part – Duplicates shows a table with two products. Different textual representations of PNAME (that identifies the same real-world entity) cause duplicates stored in this table which results in a duplicate anomaly.

Part

PNO	PNAME	QOH	PRICE	LEVEL
10506	Land Before Time I	200	19.99	20
10508	Land Before Time 1	156	19.99	20

Figure 23: Table Part – Duplicates

Thirdly there is the process of duplicate detection and elimination. The goal of the overall data cleaning process is to identify, find and eliminate anomalies as mentioned in chapter 3. The goal of the duplicate detection process is more special. It has to search and find tuples in the database, which are duplicates and correct those.

4.6.2 Data Cleaning Methods

This chapter describes which kinds of techniques can be used to find and eliminate duplicates. A large number of duplicate anomalies are caused by typographical variations of strings e. g. different spelling of names in different databases. There are many methods e.g. Q-grams [44], SOUNDEX [29] or WHIRL [7], which can solve this problems and each of them works well on different kind of errors. Those kinds of methods for duplicate detection are called **field comparison methods**. The above mentioned methods compare individual attributes of tuples

but in most situations tuples consists of more than one attribute and this makes the task of duplicate detection much more complicated. Methods which can solve this problem where duplicate tuples have to be detected by comparison of more than one field are called **duplicate detection methods** in the following text.

Field comparison methods

- **Character based similarity**
  Several metrics like the Levenshtein distance, the affine gap metric, the Smith-Waterman distance or the Jaro distance metric can be used to measure the similarity of two strings based on the characters of the string. This metrics are very useful for typographical errors which cause a duplicate anomaly.

  For example the edit-distance counts the minimum number of operations which are necessary to transform one string into another when every operation costs one point. This distance is also called the Levenshtein distance [8]. The main idea behind this character based similarity metrics is that an operation on a string which transforms that string into another costs something. If we use the edit-distance an operation costs one point like mentioned above. But this is not always the best solution. If there are two string lets say two products names for DVD-films like “land before time” and “land b. time” the edit-distance will not work very well, because the original name has been truncated [10]. Therefore, the edit-distance is not the best metric for this problem and using another metric like the affine gap metric would be wiser. The affine gap metric has especially been developed for typographical errors where a string have been truncated or shortened. As to say it strongly depends on the king of typographical error when you have to decide which character based similarity metric is the best.

  Another method to measure string similarity is called q-grams. Q-grams are small substring of strings and if two strings have a lot of identical q-grams it could be said that they are similar [44].

- **Token based similarity**
  Often words in names are rearranged because of different conventions in databases. Character based similarity cannot measure the similarity of such strings. Therefore, token based similarity methods have been developed to solve that problem. One token based similarity method has been described by Cohen and is called WHIRL. This method assumes that every strings consists of words ω an each of the words in a string is weighted.

  $$v(\omega) = \log(tf\omega + 1) * \log(idf\omega)$$

The overall weight of a word is high if it appears very often in the string but on the other hand has not been counted in the database very often. For example the term “Siemens” will have a higher weight than the term “GmbH” because the last one is a quite common word for a company name. To measure the similarity of strings a cosin similarity metric is used which calculates the similarity based on the above mention weights for words.

The WHIRL methods works fine if words in texts or names have been swapped but it cannot measure the similarity of words if there are spelling errors in the text [7].

- **Phonetic similarity**
  Many words or strings are spelled differently but sound very similar. This kind of similarity is measured by phonetic similarity methods. The following figure shows customer names that have a high phonetic similarity.

Customer

CNO	CNAME	STREET	ZIP	PHONE
1111	Meier	123 Main St.	67226	316-636-5555
3333	Mayer	111 Inwood Street.	60606	316-111-1234
4444	Maya	10 Carpent Str.	67226	456-333-2567
5555	Maier	111 Inwood Street.	60606	316-765-4915

Figure 24: Customers with a phonetic similar name

The two kinds of field matching methods which have been explained above will not work very well if someone wants to find duplicate entries for such names. The most common method to measure phonetic similarity was developed by Russel and is called the SOUNDEX [29]. SOUNDEX assumes that you identify phonetic similar groups of consonants by the same code digits and with that you can measure the similarity especially of surnames.

- **Numeric similarity**
  In comparison to the amount of methods which have been developed to compare the similarity of strings you can say that there are no special methods to measure the similarity of numeric attributes. Except for primitive methods which compare the range of to numeric attributes in most cases they are treated like strings and compared with the methods mentioned above [10].

Duplicate detection methods

- **Probabilistic methods**
  The detection of duplicate anomalies within a data warehouse can be seen as a

Bayesian inference problem. As this problem is commonly known various techniques have been developed to solve such a problem [10]:

- The Bayes Decision Rule for Minimum Error
  These Rules try to minimize the probability of an error of a misclassification.
- The Bayes Decision Rule for Minimum Cost
  The Rules try to minimize the costs of a misclassification.
- Decision with a Reject Region
  A third classification category "reject region" is used to mark those tuples which cannot be classified.

- **Learning methods**
  [10] distinguish between:
  - Supervised and Semi-Supervised Learning
  - Active-Learning
  - Unsupervised Learning

  First, supervised and semi-supervised learning methods train how to classify tuples and so they can detect duplicate anomalies. It is necessary that there exists training data where duplicate tuples are marked.

  Secondly, active learning systems try to compensate the problem that supervised learning techniques often need a huge amount of training tuples. Each of the training tuples has to be marked either as a duplicate or as an individual value. In addition, to create a really good duplicate detection system tuples are needed which cannot be classified easily. Such tuples contain a lot of information and this information is needed to train the system especially. Active learning systems do not work with a static training set of data.

  ALIAS developed by Sarwagi and Bhamidipaty is such an active learning system.

  "*The goal is to seek out from the unlabeled data pool those instances which, when labeled, will improve the accuracy of the classifier at the fastest possible rate.*" says Elmagarmid et al. about ALIAS.

  Thirdly, unsupervised learning systems do not need a training set of data. Every learning method tries to cluster the data and mark duplicate and non-duplicate tuples. To cluster a set of data you need a comparison vector which indicates the differences

of each attribute of two tuples. Many unsupervised learning methods assume that similar comparison vectors point to the same class [10].

- **Distance based methods**
  Often there are no training data to learn the systems which tuples are similar and which are not. Therefore, distance based methods have been developed because they do not need training data. Distance based methods define a distance metric for a whole tuple like the field comparison methods did for single attributes.

  One way to implement such distance based methods is to treat the whole tuple as one string attribute and use the before introduced field matching methods. But as you can imagine this is not always the best way because in many situations important information is lost if you merge all attributes of a tuple into a single one.

  Another way to measure the similarity of tuples is introduced by [2] and [23]. Both methods consider that the tuples in a database are linked together for example through well defined foreign keys. If those foreign keys are analysed it is possible to find duplicate anomalies even if the attributes of the duplicate tuples are not similar.

  As mentioned above, the main advantage of distances based methods is the fact that these methods do not need training data. But they need a threshold to measure the similarity of tuples and finding such a threshold is not an easy task [10].

- **Rule based methods**
  Rule based methods are quite similar to distance based methods where the distance is always 0 or 1. If there is no global key a rule can be used to create a unique attribute assumes [45]. Furthermore, this new unique attribute can help clustering the data and so finding duplicate tuples.

  Example Rule: IF age < 22 THEN status = undergraduate

  ELSE status = graduate

  Another possible way to use rules is shown by [17]. Rules can be used to specify when tuples are similar. For example the similarity of two employees is given when the name and the address of the two tuples are similar. If that is the case it can be assumed that there is a duplicate anomaly within the data set.

  Example Rule: FORALL (r1,r2) in EMPLOYEE

  IF r1.name is similar to r2.name AND r1.address = r2.address

THEN r1 matches r2

To measure the similarity of name field comparison methods like shown in the previous chapter should be used.

4.6.3 Sorted Neighbourhood Approach

The above mentioned methods are all dealing with the question of how the data cleaning process can detect duplicate anomalies as good as possible. But there is a different aspect of the data cleaning process which has not been pointed out before, the speed of the duplicate detection. The authors picked one approach namely the sorted neighbourhood approach to show in an example how the efficiency of the data cleaning process can be improved.

The sorted neighbourhood approach introduced by [17] creates a single list of N tuples from a given collection of databases. Then it performs three steps:

1. The method creates a key for each tuple. Therefore, it picks attributes or parts of different attributes and joins them to a new key. For example, the method can pick the first three characters of the attribute product name and the numbers of the attribute price and create a key as shown in the figure below.

Part

PNO	PNAME	QOH	PRICE	LEVEL	KEY
10506	Land Before Time I	200	19.99	20	LAN1999
10508	Land Before Time 1	156	19.99	20	LAN1999

Figure 25: Extended table "Part"

2. After that the list is sorted e.g. with a quick-sort or a merge-sort algorithm. Attributes that appear earlier in the new key have a higher priority.

3. In the third step the sorted list is merged. Therefore, a window which contains w tuples is moved sequentially through the list (see Figure 26). Every new tuple which enters the window is only compared to w-1 other tuples and though the number of comparisons during the duplicate detection process can be drastically decreased.

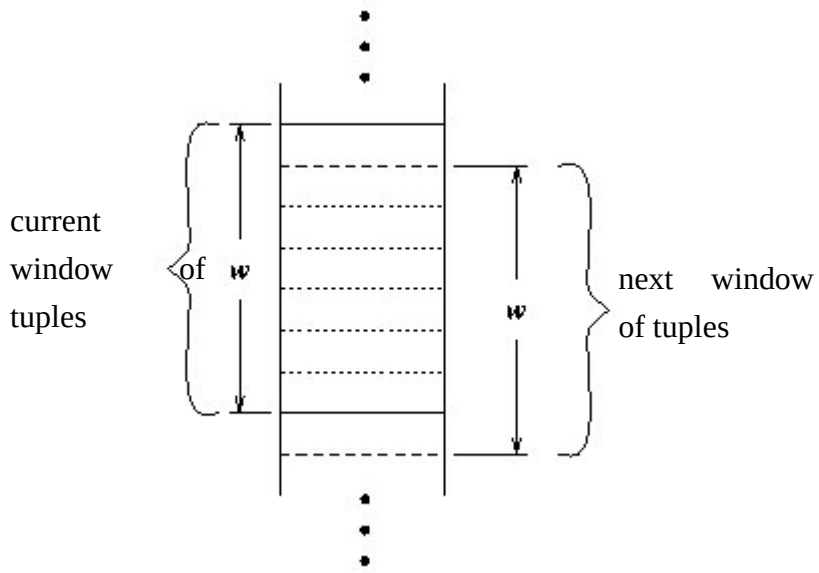

Figure 26: Merging of tuples using sliding window [34]

4.6.4 Conclusion

Though there are several methods which can be used to detect duplicate anomalies. But which of them is the best method? Which method should be used? None of the question can be answered easily. It all depends on the kind of data when you perform a duplicate detection and the authors do not believe that one single method will be found in the future which is the best for every kind of data

"The problem of choosing the best method for duplicate data detection is very similar to the problem of model selection and performance prediction for data mining: we expect that progress in that front will also benefit the task of selecting the best method for duplicate detection." [10].

4.7 Invalid Tuples

4.7.1 Introduction

Although it can not be identified as an anomaly in the first place, the "invalid tuple" error is one of the most complicated anomalies found in data warehouses. At the first look the tuple seems to be valid, because no value is missing and the values seem to be correct. However, they may be invalid because they have entities that do not appear in the real world or in the mini-world of the data warehouse. The identification of invalid tuples is very complicated and sometimes also impossible. Beside of that, the elimination of this anomaly is even harder. As they can not be really identified, the most common way is the deletion of these tuples. A

major problem of this approach is that at least one attribute of the tuple can be valid, so an information loss can occur because of the deletion of the whole tuple [29].

Odetails

ONO	PNO	QTY
1020	10506	1000

Figure 27: Table Odetails – Invalid Tuples

Again, this tuple seems to be valid. However, there is no link between this tuple and the real-world where no such entity ever existed.

4.7.2 Data Cleaning Methods

No adequate cleaning methods for this topic could be found in the literature research.

4.7.3 Conclusion

Given the fact that no cleaning methods were found, it is not really to say how appropriate invalid tuples can be solved.

4.8 Missing Value

4.8.1 Introduction

The incompleteness of data or to say it more in detail, the specific form of data incompleteness, the missing attribute values issue is the most common failure in data warehouses. Incomplete data has further consequences, despite the fact that the database is inconsistent. The data from the databases is normally used by analysts too, who need the data for statistical charting or simple data visualization. Missing data or values are a common origin for the difficulties in the process of gathering data.

Many approaches, which are explained below, exist for this kind of database errors. The approaches and methods can be split up in two groups, which are defined through the way how they are going to solve the errors. The first group of methods uses statistical information as base for the value completion. The second group is based on association rules on which the missing data will be restored.

The authors of [14] have made a comparison of different approaches in their paper. To give a short overview of the different methods, these approaches are listed below:

Most Common Attribute Value.

This is one of the simplest methods. The attribute value that occurs most often in the data is chosen for all the unknown values of the attribute [14].

Concept Most Common Attribute Value.

This method is a restriction of the first method to the concept, which means that the selection of the new attribute values depend on the most common values within the concept. This method is also called maximum relative frequency method or maximum conditional probability method [14].

"C4.5."

This is an algorithm that generates a decision tree. It is based on entropy and splitting the example with missing values to all concepts [14].

Method of Assigning All Possible Values of the Attribute.

In this method, an example with missing a missing value is replaced by a set of new examples. There, the missing attribute value is replaced by all possible values of the attribute. If there is more than one missing value, the substitution is made for one attribute first, then for the second and so on [14].

Method of Assigning All Possible Values of the Attribute Restricted to the Given Concept.

This method is a restriction of the method of assigning all possible values of the attribute to the concept, indicated by an example with a missing attribute value [14].

Method of Ignoring Examples with Unknown Attribute Values.

This is the simplest method so far. It ignores the examples that have at least one unknown attribute value and then uses the rest of the table as input to the successive learning process [14].

Event-Covering Method.

This method is a probabilistic approach to fill the missing values. The term "event-covering" means "selecting a subset of statistically interdependent events in the outcome space of variable-pairs". The static independency of variables is irrelevant to the appliance of this method [14].

Special LEM2 Algorithm.

It omits the examples with unknown values "when building the block for that attribute". A rule set is then induced by using the original LEM2 method [14].

4.8.2 Data Cleaning Methods

A current approach of solving the problem with the missing values consists of trying to determine these values. As it is said in [30] “techniques to guess the missing values must be efficient”. If the techniques are not efficient, it can happen that “the completion introduces noise”. In the following, a method is presented that satisfies these criteria and is also very good in performance. The method is called “Missing Values Completion” (MVC) [30] and works on the base of the RAR algorithm. RAR stands for “Robust Association Rules” and is an algorithm that finds quickly all the association rules in a database. Normal association rules algorithms have the problem that a lot of relevant rules are lost, the RAR algorithm has fixed this issue. As it is on the basis of the fast association rules algorithms, and that is why it became so efficient in performance, which is why it can perfectly be used in practice.

To give an overview how the MVC method works, the figure of [30] is used:

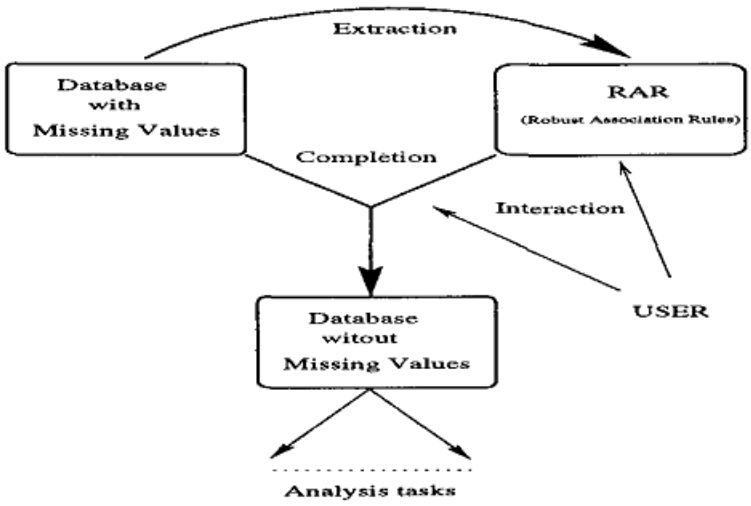

Figure 28: The MVC Process [30]

Firstly, the MVC Process uses the original database with the missing values, and searches it for association rules. These rules will then be extracted by the RAR algorithm and a set of rules is constructed as a result of this step. Secondly, this rule set will be used for the completion of the missing values.

> “*For completion, the rules matching a data and concluding on an attribute which is missing are used to decide its value. Two situations are possible:*
>
> 1. *All the matching rules indicate the same conclusion, then it is used.*
> 2. *Several matching rules conclude on different values.*”

In the second case, the user can intervene in the completion, as it is shown in Figure 29. Normally, the conflict is automatically solved with the use of "the confidence measure and the number of rules concluding on a same value".

To give an impression on how the RAR algorithm is used in practice, the example that was introduced in section 2.5 is used for illustration.

Zip code

ZIP	CITY
67226	null

Figure 29: Table Zip code – Missing Value

Each tuple stored in this table is supposed to consist of two values (because every zip code belongs to a city), but null was stored in CITY, which represents a missing value.

As first step we have to extract rules, which are used in the second step to complete the record. In this case, the example is very trivial, so the extracted rule is very simple. From the base table, Mail-Order Database – Table Zip code, shown in Figure 6, we come to: *ZIP* → *CITY*. There exists only one tuple in the Table Zip code with the ZIP value "67226". The appliance of the extracted rule leads to *67226* → *Wichita*. Accordingly to the result, the missing attribute value will be filled in as the city "Wichita".

However, this example just shows in an easy way how the algorithm can be deployed. In real life, the algorithm has to deal with much more complex table structures and many additional rules that are extracted from the greater databases.

4.8.3 Conclusion

As already mentioned in section 4.8.1 there exist a wide range of methods that deal with missing values. The RAR algorithm, and in particular, the MVC process is a straight way to handle missing values. One of the main advantages of MVC and why it was examined in this paper "is to be a pre-processing method" [30] and the possibility of an interaction of the user during the runtime of the process.

4.9 Missing Tuple

4.9.1 Introduction

Missing tuples can occur in a data warehouse if some entities that exist in the mini-world are not represented by tuples in the database. This can happen if a wrong Join-Operator is used to join two or more tables. To illustrate this circumstance in a short example is given:

T1

A	B
A1	B1
A2	B2

T2

A	C
A2	C1
A2	C2
A3	C3

Figure 30: Example Tables

T1 and T2 are our base tables. Now a join of the two tables is going to be made. Firstly, a Left-Outer-Join to combine the two simple relations is used. The result of this Join will look like this:

T1 ▷ T2

A	B	C
A1	B1	Null
A2	B2	C1
A2	B2	C2

Figure 31: Left-Outer-Join Example Table

As seen in Figure 32, the new table T1 ▷ T2 has not only a missing value, but the whole tuple [A3] [C3] from T2 is missing. Here the same procedure with a Right-Outer-Join:

T1 ◁ T2

A	B	C
A2	B2	C1
A2	B2	C2
A3	Null	C3

Figure 32: Right-Outer-Join Example Table

Again, one value is missing and also the tuple [A1] [B1] from T1. But the missing value is not the anomaly that should be cared about. In order to show how the joined Table changes a Full Natural Outer-Join is used:

T1 ◁▷ T2

A	B	C
A1	B1	Null
A2	B2	C1
A2	B2	C2
A3	Null	C3

Figure 33: Full Natural Outer-Join Example Table

With a Full Natural Outer-Join the new table has no missing tuples. That is because both base tables, T1 and T2, are dominant for this type of Join. No tuple and aside of that no information get lost within or after the Join.

4.9.2 Data Cleaning Methods

As shown in the example in the introduction of missing tuples, the selection of a correct Join-Operator is important to avoid incompleteness in the data. However, adequate methods and approaches could not be found which are dealing with solving of missing tuples in databases.

In the example, which is described in section 3, the table “Employee” consists of three tuples. The Employee „Smith“ is working for the company, but the appropriate tuple is missing in the table below.

Employee

ENO	ENAME	ZIP	HDATE
1000	Jones	67226	12-DEC-95
1002	Brown	50302	01-SEP-94

Figure 34: Table Employee – Missing Tuple

It is suggested that this is a result of the use of a wrong Join-Operator, and can be avoided by using an appropriate one.

4.9.3 Conclusion

Missing Tuples are a very common problem in data warehouses. The example in section 4.9.1 sounds trivial, which indeed, is the case. However, the discovery of missing tuples is not easy as that, the recovery of a missing tuple is even harder if the joined relations are not known, maybe a result of missing or weak documentation.

5 Data Cleaning Frameworks

As one can observe from the sections on the different anomalies it is not possible to put all the detection and elimination methods on equal footing in order to carry out a decent comparison. Therefore this section will give an overview about data cleansing frameworks which incorporate different data cleansing methods. It concludes with a comparison of the following five approaches:

- AJAX[8]
- FraQL
- Potter's Wheel
- ARKTOS
- IntelliClean

5.1 AJAX

[12] and [13] claim that existing ETL (Extraction Transformation Loading) and data cleaning tools are insufficient for writing data cleaning programs. Therefore they develop a framework, called AJAX, which allows declarative expression of data cleaning specifications. One of the essential characteristics of the presented framework is the clear separation between the logical specification of data transformations and their physical implementation.

In the specification of AJAX [12] and [13] describe the following three data problems: (1) the Object Identity problem, (2) Keyboard errors in the data and (3) inconsistencies in data coming from multiple-sources.

In order to deal with these problems the described framework provides the services data transformation, duplicate elimination and multi-table matching. For this purpose the authors introduce SQL-like macro-operators: Mapping, Matching, Decision and Clustering. The authors provide a detailed description of all macro-operators including examples.

In addition the framework incorporates a metadata repository to allow the explanation of the data transformation process. Furthermore, the authors list a number of optimization

[8] Not to be confused with „Asynchronous JavaScript and XML"

techniques to reduce the performance penalty imposed by the fact that the matching operator in the framework is based on the Cartesian product.

5.2 FraQL

According to [39] and [40] the main problem of data integration is different modelling of real-world entities. As part of data integration different representations of one and the same object have to be resolved by a mapping procedure.

Therefore, [39] and [40] have developed FraQL, a SQL based declarative language. The main purpose of FraQL is the specification of the data cleansing process. FraQL enables the user to identify and eliminate duplicates by extending the standard join and union operators. Furthermore it provides functionality to fill in missing values or deleting invalid tuples by detection and removal of outliers.

5.3 Potter's Wheel

[32] describe Potter's Wheel as *"an interactive framework for data cleaning that tightly integrates transformation and discrepancy detection"*. Potter's Wheel enables the user to carry out data transformations and error detection in a spreadsheet-like manner. Therefore the user can immediately see the results on the screen.

The approach of Potter's Wheel is an adequate method to detect syntax errors and irregularities. This is achieved by solving schematic heterogeneities. Furthermore Potter's Wheel can be used to specify the data cleaning process in an interactive way, because the user always receives immediate feedback. For this reason Potter's Wheel enables the user to gradually refine and develop the data cleaning process.

5.4 ARKTOS

ARKTOS is a metamodel for the ETL (extraction, transformation and loading) process which should reduce the problems of complexity, usability and price. ARKTOS can be used to describe ETL scenarios. Therefore it provides three ways: a graphical point-and-click front end, XADL (which is an XML variant) and SADL (which is a SQL-like language with a compact syntax).

[29] point out that with ARKTOS data cleansing is "an integral part of this ETL process. The framework itself consists of activities, where each activity represents a single cleansing operation. Furthermore the activities are linked to input and output relations. ARKTOS provides an SQL-like language to define the logic of the different activities.

ARKTOS is able to detect the following six types of errors:

- Primary key violation
- Uniqueness violation
- Reference violation
- Null existence
- Domain mismatch
- Format mismatch

Therefore ARKTOS can be used to detect irregularities, domain format errors, integrity constraint violations, contradictions and duplicates. The error correction possibilities in ARKTOS are rather limited. If an error is found, the relevant records can be ignored, deleted, written to a file or inserted in a specific table.

5.5 IntelliClean

The main aim of IntelliClean is the detection and elimination of duplicate records. This framework has been developed and implemented by [24] and [25]. A specific feature of this framework is that the process of data cleansing is approached from a rule based point of view.

IntelliClean basically consists of the following three steps or stages:

- The pre-processing stage where syntactical errors are eliminated. Furthermore this stage is used to put the values from the different data sources in a standardized format.
- The processing stage which, according to [29] *"represents the evaluation of cleansing rules on the conditioned data items that specify actions taken under certain circumstances"*.
- The human verification and validation stage, where the results produced by the previous to stages are verified.

The IntelliClean framework is an adequate solution to the anomaly of duplicate records.

5.6 Comparison

To conclude the section about current approaches to data cleaning, a comparison carried out by [29] will be presented.

	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			

Figure 35: Comparison of Data Cleaning Frameworks[9]

Figure 35 shows an overview about the current approaches to data cleaning and which methods they implement in order to tackle the different types of anomalies. Cases where the detection and elimination of an anomaly is possible but not specified are indicated by the term "*User Defined*".

Many problems in the field of data cleaning arise from the process of merging data from different sources with different schemes. All of the cleansing frameworks presented in this section try to enable the user to model the cleansing and merging process. Therefore all

[9] Found in [29]

methods try to cover the anomalies described in this paper. Frameworks like AJAX, IntelliClean and Potter's Wheel originate from scientific work and projects. For this reason it is difficult to obtain these frameworks in an executable form.

Although the authors of this paper cannot provide a ranking of the frameworks, the following section highlights the strengths and weaknesses of the different frameworks concerning the adequacy to handle the different anomalies. The information presented in the table below should enable readers to pick out a cleansing framework adequate for their needs.

Framework	Strengths (+)	Weaknesses (-)
AJAX	+ Data-integration from multiple sources + Elimination of duplicates + Constraint checking with exception handling	- Handling of lexical errors - Handling of missing values and/or tuples - Correction of integrity violating tuples
FraQL	+ Identification and elimination of duplicates through extended join and union operators + Elimination of invalid tuples through detection and exclusion of outliers + Handling of missing values	- Handling of lexical errors - Handling of constraint violations - Handling of missing tuples
Potter's Wheel	+ Detection of syntax errors + Detection of irregularities + Interactive process	- Matching and merging are not implemented in the system - Detection and elimination of duplicates - Handling of missing or invalid values and/or tuples
ARKTOS	+ Constraint checking + Detection of lexical errors + Detection of domain format errors	- Handling of found violations - Detection and elimination of duplicates - Handling of missing or invalid values and/or tuples

IntelliClean	+ Detection and elimination of duplicates + Elimination of syntactical errors + Handling of constraint violations	- Handling of missing values and/or tuples

Figure 36: Strengths and weaknesses of the cleansing frameworks

6 Conclusion and Open Questions

One of the major problems of anomalies that were identified are invalid tuples. As there were no adequate cleaning methods or approaches found, the only way to resolve this anomaly is the deletion of the relevant tuples. Valid data in the tuples can be deleted too, which results in the loss of relevant data. A recovery method would be very important in this case, as the anomaly can only be prevented by correct joining.

Many different approaches exist for missing values. It can not be said which one is the best, because every method has its pros and cons in repairing the anomaly. The use of statistical information for retrieving missing tuples is practiced by a lot of approaches like Most Common Attribute Value or Concept Most Common Attribute Value. However, this statistical information can also inhabit wrong interpretation or simply wrong data, so the retrieved value will not correspond to the lost value. The main problem here is that the statistical information itself can be erroneous and/or vague.

Furthermore, it has been discovered that there are plenty of methods to detect anomalies especially for duplicate anomalies. [10] classifies those methods into field comparison methods and duplicate detection methods. It strongly depends on the kind of anomaly which occurs in the data provide if experts have to decide which they want to use. For example, the Levenshtein distance is a relatively simple metric to measure the similarity of two words and therefore the costs when you use it are low but if you want to measure the similarity of a name field in a database other methods will be more efficient.

The Sorted Neighbourhood Approach is also described in this paper. This method does not care about how good the detection of an anomaly is it is all about the detection speed. A Data Warehouse contains a huge amount of data and therefore, methods have been developed to speed up the detection of anomalies for example by reducing the amount tuples which have to be compared.

The main focus of the methods and algorithms presented in this paper is on the detection of the different types of anomalies. Furthermore, the authors of this paper also pointed out methods, which prevent circumstances under which the anomalies arise. Nevertheless, only little has been said about how to repair “invalid” records or tables. There is only little literature about how to handle inconsistent data in data warehouses. The first-best choice for many methods is the deletion of dirty data.

Authors like [11] argue that this kind of anomaly handling is not adequate for different cases. Therefore there are still open questions like how to present violated integrity constraints to the user in a meaningful way to enable the user to modify the transaction in order to be valid.

7 Glossary

Attribute
Column of a table in a relational database.

Cartesian Product
Each tuple of one table is combined with each and every tuple of another table.

Data Joining Process
Process, integrating data from different, probably spatially distributed databases.

Duplicate
If two or more entries in a database represent exactly the same entity.

Entity
An object which can be identified uniquely.

Foreign Key
"A referential constraint between two tables[1]. The foreign key identifies a column or a set of columns in one (referencing) table that refers to a column or set of columns in another (referenced) table."[10]

Functional Dependency
"A constraint between two sets of attributes in a relation from a database"[11].

Integrity Constraint
Rules to keep the database in a consistent state.

Metric
An abstraction of the notion of distance in a metric space.

OLS
Ordinary Least Squares – "*a method for linear regression that determines the values of unknown quantities in a statistical model by minimizing the sum of the residuals (difference between the predicted and observed values) squared"*[12].

[10] Source: http://en.wikipedia.org/wiki/Foreign_key, accessed in June 2007

[11] Source: http://en.wikipedia.org/wiki/Functional_dependency, accessed in June 2007

Outlier Analysis
Methods to detect observations which are distant from the rest of the data.

Q-Gram
Small substring of strings. If two strings have a lot of identical q-grams it could be said that they are similar.

Regression Analysis
Statistical method to determine the influence of one or many independent variables on a specific dependent variable.

Relation
Table in a database, consisting of rows (=tuples) and columns (=attributes).

SOUNDEX
Common method to measure phonetic similarity, developed by Russel.

Transaction
Set of queries or commands sent to a database which has to be either fully completed or aborted.

Tuple
Row in a table of a relational database, representing an entity.

WHIRL
Token based similarity method, described by Cohen.

[12] Source: http://en.wikipedia.org/wiki/Ordinary_Least_Squares, accessed in June 2007

8 Annotation

8.1 List of Figures

8.2 References

[1] S. Abiteboul, S. Cluet, T. Milo, P. Mogilevsky, J. Simeon, S. Zohar. Tools for Data Translation and Integration. In IEEE Data Engineering Bulletin, 22(1), pages 3–8, 1999.

[2] R. Ananthakrishna, S. Chaudhuri, V. Ganti. Eliminating fuzzy duplicates in data warehouses. In Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002), 2002.

[3] Arning, A.; Agrawal, R.; Raghavan, P.: A Linear Method for Deviation Detection in Large Databases, in Proc. of the 2nd International Conference on Knowledge Discovery & Data Mining, AAAI Press, 1996.

[4] D.P. Ballou, H.L. Pazer. Modeling data and process quality in multi-input, multi-output information systems. In Management Science 31 (2), pages 150–162, 1985.

[5] D.P. Ballou, G.K. Tayi. Enhancing Data Quality in Data Warehouse Environments. In Communications of the ACM 42 (1), pages 73–78, 1999.

[6] P.A. Bernstein, T. Bergstraesser. Meta-Data Support for Data Transformations Using Microsoft Repository. In Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, pages 9–14, 1999.

[7] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In Proceedings of ACM SIGMOD-98, Seattle, WA, 1998.

[8] W. W. Cohen, P. Ravikumar, S. E. Fienberg. A comparison of string metrics for matching names and records. In Proceedings of the KDD2003, 2003.

[9] R. Davidson, J.G. MacKinnon. Econometric Theory and Methods, 2004.

[10] A. Elmagarmid, P. Ipeirotis, V. Verykios. Duplicate Record Detection: A Survey. In Transactions on Knowledge and Data Engineering, 2006.

[11] S. M. Embury: Coping with Constraint Violation: the Practical Face of Database Integrity. In FMLDO, pages 141-148, 1996.

[12] H. Galhardas, D. Florescu, D. Shasha, E. Simon, C.-A. Saita. An Extensible Framework for Data Cleaning (Extended Version). In Proceedings of the 16th International Conference on Data Engineering, 2000.

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

[14] J.W. Grzymala-Busse, M. Hu. A Comparison of Several Approaches to Missing Attribute Values in Data Mining. In Rough Sets and Current Trends in Computing: Second International Conference, RSCTC 2000 Banff, October 16-19, 2000, Revised Papers. Also in Lecture Notes in Computer Science, Volume 2005/2001, 2001.

[15] A. Gupta, Y. Sagiv, J. D. Ullman, and J. Widom. Efficient and complete tests for database integrity constraint checking. In Principles and Practice of Constraint Programming, pages 173–180, 1994.

[16] M. Helfert, C. Herrmann. Proactive data quality management for data warehouse systems. In Competence Center 'Data Warehousing 2', pages 97–106, 2002.

[17] M.A. Hern´andez, S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery 2(1), pages 9-37, January 1998.

[18] H. Hinrichs, T. Aden. An ISO 9001:2000 Compliant Quality Management System for Data Integration in Data Warehouse Systems. In Proceedings of the International Workshop on Design and Management of Data Warehouses (1), pages 1–12, 2001.

[19] Huhtala Y., Karkkainen J., Porkka P., and Toivonen H. Tane. An efficient algorithm for discovering functional and approximate dependencies. In The Computer Journal, 42(2), pages 100-111, 1999.

[20] International Organisation for Standardisation: ISO 9001:2000. At http://www.iso.org/, accessed in May 2007.

[21] A. Koeller and E. A. Rundensteiner. Discovery of highdimensional inclusion dependencies. Technical Report WPICS -TR-02-15, Worcester Polytechnic Institute, Dept. of Computer Science, 2002.

[22] R.L. Leitheiser. Data Quality in Health Care Data Warehouse Environments. In Proceedings of the 34th Hawaii International Conference on System Sciences, pages 1–10, 2001.

[23] M.L. Lee, W. Hsu, V. Kothari. Cleaning the Spurious Links in Data. In IEEE Intelligent Systems, 2004.

[24] M.L. Lee, T.W. Ling, W.L. Low. IntelliClean: A Knowledge-Based Intelligent Data Cleaner. In Knowledge Discovery and Data Mining, pages 290–294, 2000.

[25] W.L. Low, M.L Lee, T.W. Ling. A knowledge-based approach for duplicate elimination in data cleaning. In Information Systems 26 (8), pages 585–606, 2001.

[26] G. Mokotoff. SOUNDEX. At http://www.avotaynu.com/soundex.htm, accessed in May 2007.

[27] A. Motro. Integrity = Validity + Completeness. In ACM Transactions on Database Systems (TODS) 14 (4), pages 480–502, 1989.

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

[29] H. Müller, J.C. Freytag. Problems, Methods, and Challenges in Comprehensive Data Cleansing. In Technical Report HUB-IB-164, 2003.

[30] A. Ragel. Preprocessing of Missing Values Using Robust Association Rules. In Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery, Lecture Notes In Computer Science, Vol. 1510, 1998.

[31] E. Rahm, H.H. Do. Data Cleaning: Problems and Current Aproaches. In IEEE Technical Bulletin on Data Engineering, 2000.

[32] V. Raman, J.M. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. In Proc. of VLDB, 2001.

[33] R. Ramanathan. Introductory Econometrics with Applications. Thomson Learning, Auflage 5, 2001.

[34] V.T. Raisinghani. Cleaning methods in data warehousing. In Seminar Report, 1999.

[35] T.C. Redman. The Impact of Poor Data Quality on the Typical Enterprise. In Communications of the ACM 41 (2), pages 79–82, 1998.

[36] M. Rittmann. Data Profiling and Automated Cleansing Using Oracle Warehouse Builder 10g Release 2. In Oracle DBA: Data Warehousing & Integration, 2006.

[37] K. Rother, H. Müller, S. Trissl, I. Koch, T. Steinke, R. Preissner, C. Frömmel, U. Leser. COLUMBA: Multidimensional Data Integration of Protein Annotations. In Data Integration in the Life Sciences (2994), pages 156–171, 2004.

[38] Sapia, C.; Höfling, G.; Müller, M.; Hausdorf, C.; Stoyan, H.; Grimmer, U.: On Supporting the Data Warehouse. Design by Data Mining Techniques. Proc. GI-Workshop Data Mining and Data Warehousing, 1999.

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

[40] K. Sattler, E. Schallehn. A Data Preparation Framework based on a Multidatabase Language. In Proc. of Int. Database Engineering and Applications Symposium, pages 219–228, 2001.

[41] G.K. Tayi, D.P. Ballou. Examining Data Quality. In Communications of the ACM 41 (2), pages 54–57, 1998

[42] D. Tonn. Data Cleansing Verfahren für Data Warehouses. Diploma Thesis, HUB, 2000.

[43] J.L. Tripp, A.A. Hanson, M. Gokhale, H. Mortveit. Improving Data Quality in Practice: A Case Study. In Proceedings of the 2005 ACM/IEEE conference on Supercomputing SC '05, 2005.

[44] E. Ukkonen. Approximate string matching with q-grams and maximal matches. Theoretical Computer Science 92(1), pages 191-211, 1992.

[45] Y.R. Wang, S.E. Madnick. The inter-database instance identifcation problem in integrating autonomous systems. In Proceedings of the Fifth IEEE International Conference on Data Engineering (ICDE 1989), pages 46-55, 1989.

[46] R. Wang, D. Strong. Beyond accuracy: What data quality means to data consumers. In Journal of Management Information Systems 2 (4), pages 5–34, 1996.

[47] J. M. Wooldridge. Econometric Analysis of Cross Section and Panel Data. Southwestern, 2002.

[48] J. M. Wooldridge. Introductory Econometrics: A Modern Approach. Southwestern, 2006.

8.3 Distribution of Work

	Mario Hofer	Tobias Oberascher	Alexander Sgardelli	Markus Teufl
Introduction				
Abstract				
Data Quality Management				
Data Cleaning Process				
Anomalies – Introduction				
Lexical Error & Domain Format Error				
Irregularities				
Integrity Constraint Violation				
Duplicates				
Invalid Tuples				
Missing Value				
Missing Tuple				

Data Cleaning Frameworks				
Conclusion & Open Questions, Glossary				
Coordination				

Figure 37: Distribution of Work