Data Cleaning: Problems and Current Approaches

Erhard Rahm* Hong Hai Do
University of Leipzig, Germany
http://dbs.uni-leipzig.de

Abstract

We classify data quality problems that are addressed by data cleaning and provide an overview of the main solution approaches. Data cleaning is especially required when integrating heterogeneous data sources and should be addressed together with schema-related data transformations. In data warehouses, data cleaning is a major part of the so-called ETL process. We also discuss current tool support for data cleaning.

1 Introduction

Data cleaning, also called *data cleansing* or *scrubbing*, deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data. Data quality problems are present in single data collections, such as files and databases, e.g., due to misspellings during data entry, missing information or other invalid data. When multiple data sources need to be integrated, e.g., in data warehouses, federated database systems or global web-based information systems, the need for data cleaning increases significantly. This is because the sources often contain redundant data in different representations. In order to provide access to accurate and consistent data, consolidation of different data representations and elimination of duplicate information become necessary.

Data warehouses [6, 16] require and provide extensive support for data cleaning. They load and continuously refresh huge amounts of data from a variety of sources so the probability that some of the sources contain "dirty data" is high. Furthermore, data warehouses are used for decision making, so that the correctness of their data is vital to avoid wrong conclusions. For instance, duplicated or missing information will produce incorrect or misleading statistics ("garbage in, garbage out"). Due to the wide range of possible data inconsistencies and the sheer data volume, data cleaning is considered to be one of the biggest problems in data warehousing. During the so-called ETL process (extraction, transformation, loading), illustrated in Fig. 1, further data transformations deal with schema/data translation and integration, and with filtering and aggregating data to be stored in the warehouse. As indicated in Fig. 1, all data cleaning is typically performed in a separate data staging area before loading the transformed data into the warehouse. A large number of tools of varying functionality is available to support these tasks, but often a significant portion of the cleaning and transformation work has to be done manually or by low-level programs that are difficult to write and maintain.

Federated database systems and web-based information systems face data transformation steps similar to those of data warehouses. In particular, there is typically a *wrapper* per data source for extraction and a *mediator* for integration [32, 31]. So far, these systems provide only limited support for data cleaning, focusing

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

*This work was performed while on leave at Microsoft Research, Redmond, WA.

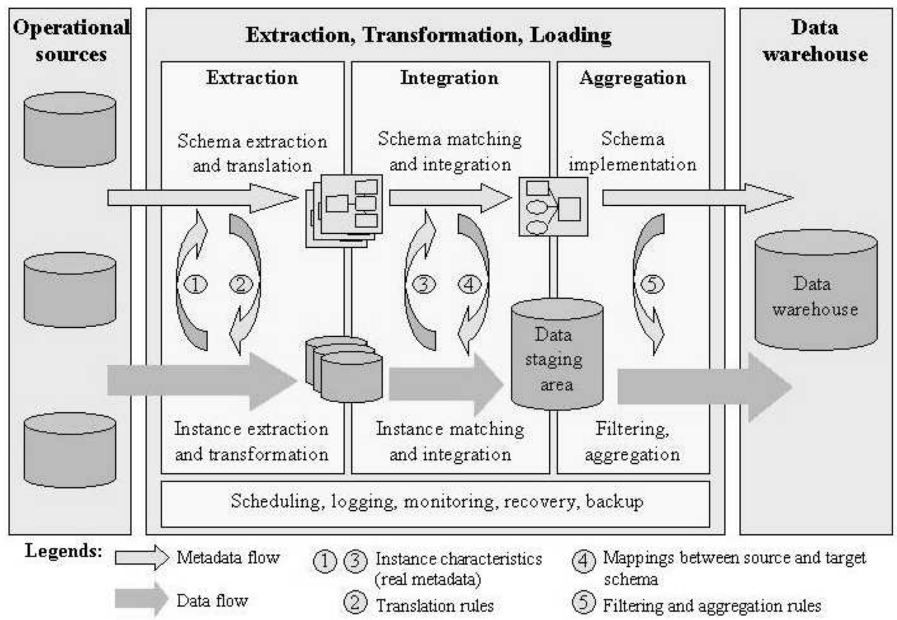

Figure 1: Steps of building a data warehouse: the ETL process

instead on data transformations for schema translation and schema integration. Data is not preintegrated as for data warehouses but needs to be extracted from multiple sources, transformed and combined during query runtime. The corresponding communication and processing delays can be significant, making it difficult to achieve acceptable response times. The effort needed for data cleaning during extraction and integration will further increase response times but is mandatory to achieve useful query results.

A data cleaning approach should satisfy several requirements. First of all, it should detect and remove all major errors and inconsistencies both in individual data sources and when integrating multiple sources. The approach should be supported by tools to limit manual inspection and programming effort and be extensible to easily cover additional sources. Furthermore, data cleaning should not be performed in isolation but together with schema-related data transformations based on comprehensive metadata. Mapping functions for data cleaning and other data transformations should be specified in a declarative way and be reusable for other data sources as well as for query processing. Especially for data warehouses, a workflow infrastructure should be supported to execute all data transformation steps for multiple sources and large data sets in a reliable and efficient way.

While a huge body of research deals with schema translation and schema integration, data cleaning has received only little attention in the research community. A number of authors focussed on the problem of duplicate identification and elimination, e.g., [11, 12, 15, 19, 22, 23]. Some research groups concentrate on general problems not limited but relevant to data cleaning, such as special data mining approaches [29, 30], and data transformations based on schema matching [1, 21]. More recently, several research efforts propose and investigate a more comprehensive and uniform treatment of data cleaning covering several transformation phases, specific operators and their implementation [11, 19, 25].

In this paper we provide an overview of the problems to be addressed by data cleaning and their solution. In the next section we present a classification of the problems. Section 3 discusses the main cleaning approaches used in available tools and the research literature. Section 4 gives an overview of commercial tools for data cleaning, including ETL tools. Section 5 is the conclusion.

2 Data cleaning problems

This section classifies the major data quality problems to be solved by data cleaning and data transformation. As we will see, these problems are closely related and should thus be treated in a uniform way. Data transformations [26] are needed to support any changes in the structure, representation or content of data. These transformations become necessary in many situations, e.g., to deal with schema evolution, migrating a legacy system to a new

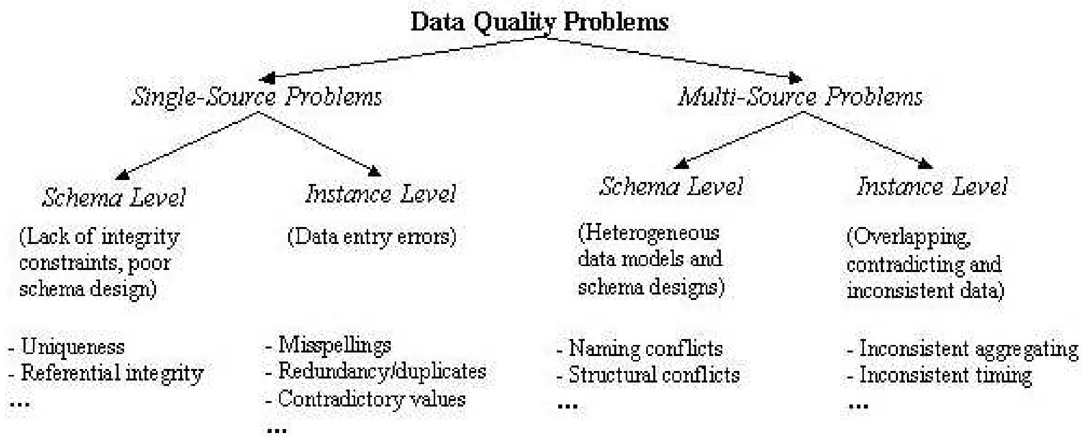

Figure 2: Classification of data quality problems in data sources

information system, or when multiple data sources are to be integrated.

As shown in Fig. 2 we roughly distinguish between single-source and multi-source problems and between schema- and instance-related problems. Schema-level problems of course are also reflected in the instances; they can be addressed at the schema level by an improved schema design (schema evolution), schema translation and schema integration. Instance-level problems, on the other hand, refer to errors and inconsistencies in the actual data contents which are not visible at the schema level. They are the primary focus of data cleaning. Fig. 2 also indicates some typical problems for the various cases. While not shown in Fig. 2, the single-source problems occur (with increased likelihood) in the multi-source case, too, besides specific multi-source problems.

2.1 Single-source problems

The data quality of a source largely depends on the degree to which it is governed by schema and integrity constraints controlling permissible data values. For sources without schema, such as files, there are few restrictions on what data can be entered and stored, giving rise to a high probability of errors and inconsistencies. Database systems, on the other hand, enforce restrictions of a specific data model (e.g., the relational approach requires simple attribute values, referential integrity, etc.) as well as application-specific integrity constraints. Schema-related data quality problems thus occur because of the lack of appropriate model-specific or application-specific integrity constraints, e.g., due to data model limitations or poor schema design, or because only a few integrity constraints were defined to limit the overhead for integrity control. Instance-specific problems relate to errors and inconsistencies that cannot be prevented at the schema level (e.g., misspellings).

Scope/Problem		**Dirty Data**	**Reasons/Remarks**
Attribute	Illegal values	bdate=30.13.70	values outside of domain range
Record	Violated attribute dependencies	age=22, bdate=12.02.70	age = current year - birth year should hold
Record type	Uniqueness violation	emp_1=(name="John Smith", SSN="123456"); emp_2=(name="Peter Miller", SSN="123456")	uniqueness for SSN (social security number) violated
Source	Referential integrity violation	emp=(name="John Smith", deptno=127)	referenced department (127) not defined

Table 1: Examples for single-source problems at schema level (violated integrity constraints)

For both schema- and instance-level problems we can differentiate different problem scopes: attribute (field), record, record type and source; examples for the various cases are shown in Tables 1 and 2. Note that uniqueness constraints specified at the schema level do not prevent duplicated instances, e.g., if information on the same real world entity is entered twice with different attribute values (see example in Table 2).

Given that cleaning data sources is an expensive process, preventing dirty data to be entered is obviously an important step to reduce the cleaning problem. This requires an appropriate design of the database schema and integrity constraints as well as of data entry applications. Also, the discovery of data cleaning rules during warehouse design can suggest improvements to the constraints enforced by existing schemas.

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Missing values	phone=9999-999999	unavailable values during data entry (dummy values or null)
	Misspellings	city="Liipzig"	usually typos, phonetic errors
	Cryptic values, Abbreviations	experience="B"; occupation="DB Prog."	
	Embedded values	name="J. Smith 12.02.70 New York"	multiple values entered in one attribute (e.g. in a free-form field)
	Misfielded values	city="Germany"	
Record	Violated attribute dependencies	city="Redmond", zip=77777	city and zip code should correspond
Record type	Word transpositions	$name_1$= "J. Smith", $name_2$="Miller P."	usually in a free-form field
	Duplicated records	emp_1=(name="John Smith",. . .); emp_2=(name="J. Smith",. . .)	same employee represented twice due to some data entry errors
	Contradicting records	emp_1=(name="John Smith", bdate=12.02.70); emp_2=(name="John Smith", bdate=12.12.70)	the same real world entity is described by different values
Source	Wrong references	emp=(name="John Smith", deptno=17)	referenced department (17) is defined but wrong

Table 2: Examples for single-source problems at instance level

2.2 Multi-source problems

The problems present in single sources are aggravated when multiple sources need to be integrated. Each source may contain dirty data and the data in the sources may be represented differently, overlap or contradict. This is because the sources are typically developed, deployed and maintained independently to serve specific needs. This results in a large degree of heterogeneity w.r.t. data management systems, data models, schema designs and the actual data.

At the schema level, data model and schema design differences are to be addressed by the steps of schema translation and schema integration, respectively. The main problems w.r.t. schema design are naming and structural conflicts [2, 24, 17]. Naming conflicts arise when the same name is used for different objects (homonyms) or different names are used for the same object (synonyms). Structural conflicts occur in many variations and refer to different representations of the same object in different sources, e.g., attribute vs. table representation, different component structure, different data types, different integrity constraints, etc.

In addition to schema-level conflicts, many conflicts appear only at the instance level (data conflicts). All problems from the single-source case can occur with different representations in different sources (e.g., duplicated records, contradicting records,. . .). Furthermore, even when there are the same attribute names and data types, there may be different value representations (e.g., for marital status) or different interpretation of the values (e.g., measurement units Dollar vs. Euro) across sources. Moreover, information in the sources may be provided at different aggregation levels (e.g., sales per product vs. sales per product group) or refer to different points in time (e.g. current sales as of yesterday for source 1 vs. as of last week for source 2).

A main problem for cleaning data from multiple sources is to identify overlapping data, in particular matching records referring to the same real-world entity (e.g., customer). This problem is also referred to as the object identity problem [11], duplicate elimination or the merge/purge problem [15]. Frequently, the information is only partially redundant and the sources may complement each other by providing additional information about an entity. Thus duplicate information should be purged out and complementing information should be consolidated and merged in order to achieve a consistent view of real world entities.

The two sources in the example of Fig. 3 are both in relational format but exhibit schema and data conflicts. At the schema level, there are name conflicts (synonyms *Customer/Client, Cid/Cno, Sex/Gender*) and structural conflicts (different representations for names and addresses). At the instance level, we note that there are different gender representations ("0"/"1" vs. "F"/"M") and presumably a duplicate record (Kristen Smith). The latter observation also reveals that while *Cid/Cno* are both source-specific identifiers, their contents are not comparable between the sources; different numbers (11/493) may refer to the same person while different persons can have the same number (24). Solving these problems requires both schema integration and data cleaning; the third table shows a possible solution. Note that the schema conflicts should be resolved first to allow data

Customer (source 1)

CID	*Name*	*Street*	*City*	*Sex*
11	Kristen Smith	2 Hurley Pl	South Fork, MN 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

Client (source 2)

Cno	*LastName*	*FirstName*	*Gender*	*Address*	*Phone/Fax*
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666

Customers (integrated target with cleaned data)

No	*LName*	*FName*	*Gender*	*Street*	*City*	*State*	*ZIP*	*Phone*	*Fax*	*CID*	*Cno*
1	Smith	Kristen L.	F	2 Hurley Place	South Fork	MN	48503-5998	444-555-6666		11	493
2	Smith	Christian	M	2 Hurley Place	South Fork	MN	48503-5998			24	
3	Smith	Christoph	M	23 Harley Street	Chicago	IL	60633-2394	333-222-6542	333-222-6599		24

Figure 3: Examples of multi-source problems at schema and instance level

cleaning, in particular detection of duplicates based on a uniform representation of names and addresses, and matching of the *Gender/Sex* values.

3 Data cleaning approaches

In general, data cleaning involves several phases

- *Data analysis*: In order to detect which kinds of errors and inconsistencies are to be removed, a detailed data analysis is required. In addition to a manual inspection of the data or data samples, analysis programs should be used to gain metadata about the data properties and detect data quality problems.
- *Definition of transformation workflow and mapping rules*: Depending on the number of data sources, their degree of heterogeneity and the "dirtyness" of the data, a large number of data transformation and cleaning steps may have to be executed. Sometime, a schema translation is used to map sources to a common data model; for data warehouses, typically a relational representation is used. Early data cleaning steps can correct single-source instance problems and prepare the data for integration. Later steps deal with schema/data integration and cleaning multi-source instance problems, e.g., duplicates. For data warehousing, the control and data flow for these transformation and cleaning steps should be specified within a workflow that defines the ETL process (Fig. 1).
  The schema-related data transformations as well as the cleaning steps should be specified by a declarative query and mapping language as far as possible, to enable automatic generation of the transformation code. In addition, it should be possible to invoke user-written cleaning code and special-purpose tools during a data transformation workflow. The transformation steps may request user feedback on data instances for which they have no built-in cleaning logic.
- *Verification*: The correctness and effectiveness of a transformation workflow and the transformation definitions should be tested and evaluated, e.g., on a sample or copy of the source data, to improve the definitions if necessary. Multiple iterations of the analysis, design and verification steps may be needed, e.g., since some errors only become apparent after applying some transformations.
- *Transformation*: Execution of the transformation steps either by running the ETL workflow for loading and refreshing a data warehouse or during answering queries on multiple sources.
- *Backflow of cleaned data*: After (single-source) errors are removed, the cleaned data should also replace the dirty data in the original sources in order to give legacy applications the improved data too and to avoid redoing the cleaning work for future data extractions. For data warehousing, the cleaned data is available from the data staging area (Fig. 1).

The transformation process obviously requires a large amount of metadata, such as schemas, instance-level data characteristics, transformation mappings, workflow definitions, etc. For consistency, flexibility and ease of reuse, this metadata should be maintained in a DBMS-based repository [4]. To support data quality, detailed information about the transformation process is to be recorded, both in the repository and in the transformed instances, in particular information about the completeness and freshness of source data and lineage information about the origin of transformed objects and the changes applied to them. For instance, in Fig. 3, the derived table *Customers* contains the attributes *CID* and *Cno*, allowing one to trace back the source records.

In the following we describe in more detail possible approaches for data analysis (conflict detection), transformation definition and conflict resolution. For approaches to schema translation and schema integration, we refer to the literature as these problems have extensively been studied and described [2, 24, 26]. Name conflicts are typically resolved by renaming; structural conflicts require a partial restructuring and merging of the input schemas.

3.1 Data analysis

Metadata reflected in schemas is typically insufficient to assess the data quality of a source, especially if only a few integrity constraints are enforced. It is thus important to analyse the actual instances to obtain real (reengineered) metadata on data characteristics or unusual value patterns. This metadata helps finding data quality problems. Moreover, it can effectively contribute to identify attribute correspondences between source schemas (schema matching), based on which automatic data transformations can be derived [20, 9].

There are two related approaches for data analysis, data profiling and data mining. *Data profiling* focusses on the instance analysis of individual attributes. It derives information such as the data type, length, value range, discrete values and their frequency, variance, uniqueness, occurrence of null values, typical string pattern (e.g., for phone numbers), etc., providing an exact view of various quality aspects of the attribute. Table 3 shows examples of how this metadata can help detecting data quality problems.

Problems	**Metadata**	**Examples/Heuristics**
Illegal values	cardinality	e.g., cardinality (gender) $>$ 2 indicates problem
	max, min	max, min should not be outside of permissible range
	variance, deviation	variance, deviation of statistical values should not be higher than threshold
Misspellings	attribute values	sorting on values often brings misspelled values next to correct values
Missing values	null values	percentage/number of null values
	attribute values + default values	presence of default value may indicate real value is missing
Varying value representations	attribute values	comparing attribute value set of a column of one table against that of a column of another table
Duplicates	cardinality + uniqueness	attribute cardinality = # rows should hold
	attribute values	sorting values by number of occurrences; more than 1 occurrence indicates duplicates

Table 3: Examples for the use of reengineered metadata to address data quality problems

Data mining helps discover specific data patterns in large data sets, e.g., relationships holding between several attributes. This is the focus of so-called descriptive data mining models including clustering, summarization, association discovery and sequence discovery [10]. As shown in [28], integrity constraints among attributes such as functional dependencies or application-specific "business rules" can be derived, which can be used to complete missing values, correct illegal values and identify duplicate records across data sources. For example, an association rule with high confidence can hint to data quality problems in instances violating this rule. So a confidence of 99% for rule "$total = quantity * unit\ price$" indicates that 1% of the records do not comply and may require closer examination.

3.2 Defining data transformations

The data transformation process typically consists of multiple steps where each step may perform schema- and instance-related transformations (mappings). To allow a data transformation and cleaning system to generate transformation code and thus to reduce the amount of self-programming it is necessary to specify the required transformations in an appropriate language, e.g., supported by a graphical user interface. Various ETL tools (see Section 4) offer this functionality by supporting proprietary rule languages. A more general and flexible approach is the use of the standard query language SQL to perform the data transformations and utilize the possibility of application-specific language extensions, in particular user-defined functions (UDFs) supported in SQL:99 [13, 14]. UDFs can be implemented in SQL or a general-purpose programming language with embedded SQL statements. They allow implementing a wide range of data transformations and support easy reuse for different transformation and query processing tasks. Furthermore, their execution by the DBMS can reduce data access cost and thus improve performance. Finally, UDFs are part of the SQL:99 standard and should (eventually) be portable across many platforms and DBMSs.

CREATE VIEW	Customer2 (LName, FName, Gender, Street, City, State, ZIP, CID)
AS SELECT	**LastNameExtract** (Name), **FirstNameExtract** (Name), Sex, Street, **CityExtract** (City), **StateExtract** (City), **ZIPExtract** (City), CID
FROM	Customer

Figure 4: Example of data transformation mapping

Fig. 4 shows a transformation step specified in SQL:99. The example refers to Fig. 3 and covers part of the necessary data transformations to be applied to the first source. The transformation defines a view on which further mappings can be performed. The transformation performs a schema restructuring with additional attributes in the view obtained by splitting the name and address attributes of the source. The required data extractions are achieved by UDFs (shown in boldface). The UDF implementations can contain cleaning logic, e.g., to remove misspellings in city names or provide missing zip codes.

UDFs may still imply a substantial implementation effort and do not support all necessary schema transformations. In particular, simple and frequently needed functions such as attribute splitting or merging are not generically supported but need often to be re-implemented in application-specific variations (see specific extract functions in Fig. 4). More complex schema restructurings (e.g., folding and unfolding of attributes) are not supported at all. To generically support schema-related transformations, language extensions such as the SchemaSQL proposal are required [18]. Data cleaning at the instance level can also benefit from special language extensions such as a Match operator supporting "approximate joins" (see below). System support for such powerful operators can greatly simplify the programming effort for data transformations and improve performance. Some current research efforts on data cleaning are investigating the usefulness and implementation of such query language extensions [11, 25].

3.3 Conflict resolution

A set of transformation steps has to be specified and executed to resolve the various schema- and instance-level data quality problems that are reflected in the data sources at hand. Several types of transformations are to be performed on the individual data sources in order to deal with single-source problems and to prepare for integration with other sources. In addition to a possible schema translation, these preparatory steps typically include:

- *Extracting values from free-form attributes (attribute split)*: Free-form attributes often capture multiple individual values that should be extracted to achieve a more precise representation and support further cleaning steps such as instance matching and duplicate elimination. Typical examples are name and address fields (Table 2, Fig. 3, Fig. 4). Required transformations in this step are reordering of values within a field to deal with word transpositions, and value extraction for attribute splitting.
- *Validation and correction*: This step examines each source instance for data entry errors and tries to correct them automatically as far as possible. Spell checking based on dictionary lookup is useful for identifying and correcting misspellings. Furthermore, dictionaries on geographic names and zip codes help to correct address data. Attribute dependencies (birthdate - age, total price - unit price / quantity, city - phone area code,...) can be utilized to detect problems and substitute missing values or correct wrong values.
- *Standardization*: To facilitate instance matching and integration, attribute values should be converted to a consistent and uniform format. For example, date and time entries should be brought into a specific format; names and other string data should be converted to either upper or lower case, etc. Text data may be condensed and unified by performing stemming, removing prefixes, suffixes, and stop words. Further-

more, abbreviations and encoding schemes should consistently be resolved by consulting special synonym dictionaries or applying predefined conversion rules.

Dealing with multi-source problems requires restructuring of schemas to achieve a schema integration, including steps such as splitting, merging, folding and unfolding of attributes and tables. At the instance level, conflicting representations need to be resolved and overlapping data must to be dealt with. The *duplicate elimination* task is typically performed after most other transformation and cleaning steps, especially after having cleaned single-source errors and conflicting representations. It is performed either on two cleaned sources at a time or on a single already integrated data set. Duplicate elimination requires to first identify (i.e. match) similar records concerning the same real world entity. In a second step, similar records are merged into one record containing all relevant attributes without redundancy. Furthermore, redundant records are purged. In the following we discuss the key problem of instance matching. More details on the subject are provided elsewhere in this issue [22].

In the simplest case, there is an identifying attribute or attribute combination per record that can be used for matching records, e.g., if different sources share the same primary key or if there are other common unique attributes. Instance matching between different sources is then achieved by a standard equi-join on the identifying attribute(s). In the case of a single data set, matches can be determined by sorting on the identifying attribute and checking if neighboring records match. In both cases, efficient implementations can be achieved even for large data sets. Unfortunately, without common key attributes or in the presence of dirty data such straightforward approaches are often too restrictive. To determine most or all matches a ”fuzzy matching” (approximate join) becomes necessary that finds similar records based on a matching rule, e.g., specified declaratively or implemented by a user-defined function [14, 11]. For example, such a rule could state that person records are likely to correspond if name and portions of the address match. The degree of similarity between two records, often measured by a numerical value between 0 and 1, usually depends on application characteristics. For instance, different attributes in a matching rule may contribute different weight to the overall degree of similarity. For string components (e.g., customer name, company name,) exact matching and fuzzy approaches based on wildcards, character frequency, edit distance, keyboard distance and phonetic similarity (soundex) are useful [11, 15, 19]. More complex string matching approaches also considering abbreviations are presented in [23]. A general approach for matching both string and text data is the use of common information retrieval metrics. WHIRL represents a promising representative of this category using the cosine distance in the vector-space model for determining the degree of similarity between text elements [7].

Determining matching instances with such an approach is typically a very expensive operation for large data sets. Calculating the similarity value for any two records implies evaluation of the matching rule on the cartesian product of the inputs. Furthermore sorting on the similarity value is needed to determine matching records covering duplicate information. All records for which the similarity value exceeds a threshold can be considered as matches, or as match candidates to be confirmed or rejected by the user. In [15] a multi-pass approach is proposed for instance matching to reduce the overhead. It is based on matching records independently on different attributes and combining the different match results. Assuming a single input file, each match pass sorts the records on a specific attribute and only tests nearby records within a certain window on whether they satisfy a predetermined matching rule. This reduces significantly the number of match rule evaluations compared to the cartesian product approach. The total set of matches is obtained by the union of the matching pairs of each pass and their transitive closure.

4 Tool support

A large variety of tools is available on the market to support data transformation and data cleaning tasks, in particular for data warehousing.† Some tools concentrate on a specific domain, such as cleaning name and address data, or a specific cleaning phase, such as data analysis or duplicate elimination. Due to their restricted domain, specialized tools typically perform very well but must be complemented by other tools to address the broad spectrum of transformation and cleaning problems. Other tools, e.g., ETL tools, provide comprehensive transformation and workflow capabilities to cover a large part of the data transformation and cleaning process.

†For comprehensive vendor and tool listings, see commercial websites, e.g., Data Warehouse Information Center (www.dwinfocenter.org), Data Management Review (www.dmreview.com), Data Warehousing Institute (www.dw-institute.com)

A general problem of ETL tools is their limited interoperability due to proprietary application programming interfaces (API) and proprietary metadata formats making it difficult to combine the functionality of several tools [8].

We first discuss tools for data analysis and data rengineering which process instance data to identify data errors and inconsistencies, and to derive corresponding cleaning transformations. We then present specialized cleaning tools and ETL tools, respectively.

4.1 Data analysis and reengineering tools

According to our classification in 3.1, *data analysis tools* can be divided into data profiling and data mining tools. MIGRATIONARCHITECT (EvokeSoftware) is one of the few commercial *data profiling tools*. For each attribute, it determines the following real metadata: data type, length, cardinality, discrete values and their percentage, minimum and maximum values, missing values, and uniqueness. MIGRATIONARCHITECT also assists in developing the target schema for data migration. *Data mining tools*, such as WIZRULE (WizSoft) and DATAMININGSUITE (InformationDiscovery), infer relationships among attributes and their values and compute a confidence rate indicating the number of qualifying rows. In particular, WIZRULE can reveal three kinds of rules: mathematical formula, if-then rules, and spelling-based rules indicating misspelled names, e.g., "$value$ Edinburgh $appears$ 52 $times\ in\ field$ Customer; 2 $case(s)\ contain\ similar\ value(s)$". WIZRULE also automatically points to the deviations from the set of the discovered rules as suspected errors.

Data reengineering tools, e.g., INTEGRITY (Vality), utilize discovered patterns and rules to specify and perform cleaning transformations, i.e., they reengineer legacy data. In INTEGRITY, data instances undergo several analysis steps, such as parsing, data typing, pattern and frequency analysis. The result of these steps is a tabular representation of field contents, their patterns and frequencies, based on which the pattern for standardizing data can be selected. For specifying cleaning transformations, INTEGRITY provides a language including a set of operators for column transformations (e.g., move, split, delete) and row transformation (e.g., merge, split). INTEGRITY identifies and consolidates records using a statistical matching technique. Automated weighting factors are used to compute scores for ranking matches based on which the user can select the real duplicates.

4.2 Specialized cleaning tools

Specialized cleaning tools typically deal with a particular domain, mostly name and address data, or concentrate on duplicate elimination. The transformations are to be provided either in advance in the form of a rule library or interactively by the user. Alternatively, data transformations can automatically be derived from schema matching tools such as described in [21].

- *Special domain cleaning*: Names and addresses are recorded in many sources and typically have high cardinality. For example, finding customer matches is very important for customer relationship management. A number of commercial tools, e.g., IDCENTRIC (FirstLogic), PUREINTEGRATE (Oracle), QUICKADDRESS (QASSystems), REUNION (PitneyBowes), and TRILLIUM (TrilliumSoftware), focus on cleaning this kind of data. They provide techniques such as extracting and transforming name and address information into individual standard elements, validating street names, cities, and zip codes, in combination with a matching facility based on the cleaned data. They incorporate a huge library of pre-specified rules dealing with the problems commonly found in processing this data. For example, TRILLIUM's extraction (parser) and matcher module contains over 200,000 business rules. The tools also provide facilities to customize or extend the rule library with user-defined rules for specific needs.
- *Duplicate elimination*: Sample tools for duplicate identification and elimination include DATACLEANSER (EDD), MERGE/PURGELIBRARY (Sagent/QMSoftware), MATCHIT (HelpITSystems), and MASTERMERGE (PitneyBowes). Usually, they require the data sources already be cleaned for matching. Several approaches for matching attribute values are supported; tools such as DATACLEANSER and MERGE/PURGELIBRARY also allow user-specified matching rules to be integrated.

4.3 ETL tools

A large number of commercial tools support the ETL process for data warehouses in a comprehensive way, e.g., COPYMANAGER (InformationBuilders), DATASTAGE (Informix/Ardent), EXTRACT (ETI), POWERMART (Informatica), DECISIONBASE (CA/Platinum), DATATRANSFORMATIONSERVICE (Microsoft), METASUITE

(Minerva/Carleton), SAGENTSOLUTIONPLATFORM (Sagent) and WAREHOUSEADMINISTRATOR (SAS). They use a repository built on a DBMS to manage all metadata about the data sources, target schemas, mappings, script programs, etc., in a uniform way. Schemas and data are extracted from operational data sources via both native file and DBMS gateways as well as standard interfaces such as ODBC and EDA. Data transformations are defined with an easy-to-use graphical interface. To specify individual mapping steps, a proprietary rule language and a comprehensive library of predefined conversion functions are typically provided. The tools also support reusing existing transformation solutions, such as external C/C++ routines, by providing an interface to integrate them into the internal transformation library. Transformation processing is carried out either by an engine that interprets the specified transformations at runtime, or by compiled code. All engine-based tools (e.g., COPYMANAGER, DECISIONBASE, POWERMART, DATASTAGE, WAREHOUSEADMINISTRATOR), possess a scheduler and support workflows with complex execution dependencies among mapping jobs. A workflow may also invoke external tools, e.g., for specialized cleaning tasks such as name/address cleaning or duplicate elimination.

ETL tools typically have little built-in data cleaning capabilities but allow the user to specify cleaning functionality via a proprietary API. There is usually no data analysis support to automatically detect data errors and inconsistencies. However, users can implement such logic with the metadata maintained and by determining content characteristics with the help of aggregation functions (sum, count, min, max, median, variance, deviation,). The provided transformation library covers many data transformation and cleaning needs, such as data type conversions (e.g., date reformatting), string functions (e.g., split, merge, replace, sub-string search), arithmetic, scientific and statistical functions, etc. Extraction of values from free-form attributes is not completely automatic but the user has to specify the delimiters separating sub-values.

The rule languages typically cover *if-then* and *case* constructs that help handling exceptions in data values, such as misspellings, abbreviations, missing or cryptic values, and values outside of range. These problems can also be addressed by using a table lookup construct and join functionality. Support for instance matching is typically restricted to the use of the join construct and some simple string matching functions, e.g., exact or wildcard matching and soundex. However, user-defined field matching functions as well as functions for correlating field similarities can be programmed and added to the internal transformation library.

5 Conclusions

We provided a classification of data quality problems in data sources differentiating between single- and multi-source and between schema- and instance-level problems. We further outlined the major steps for data transformation and data cleaning and emphasized the need to cover schema- and instance-related data transformations in an integrated way. Furthermore, we provided an overview of commercial data cleaning tools. While the state-of-the-art in these tools is quite advanced, they do typically cover only part of the problem and still require substantial manual effort or self-programming. Furthermore, their interoperability is limited (proprietary APIs and metadata representations).

So far only a little research has appeared on data cleaning, although the large number of tools indicates both the importance and difficulty of the cleaning problem. We see several topics deserving further research. First of all, more work is needed on the design and implementation of the best language approach for supporting both schema and data transformations. For instance, operators such as Match, Merge or Mapping Composition have either been studied at the instance (data) or schema (metadata) level but may be built on similar implementation techniques. Data cleaning is not only needed for data warehousing but also for query processing on heterogeneous data sources, e.g., in web-based information systems. This environment poses much more restrictive performance constraints for data cleaning that need to be considered in the design of suitable approaches. Furthermore, data cleaning for semi-structured data, e.g., based on XML, is likely to be of great importance given the reduced structural constraints and the rapidly increasing amount of XML data.

Acknowledgments
We would like to thank Phil Bernstein, Helena Galhardas and Sunita Sarawagi for helpful comments.

References

[1] Abiteboul, S.; Clue, S.; Milo, T.; Mogilevsky, P.; Simeon, J.: *Tools for Data Translation and Integration*. In [26]:3-8, 1999.

[2] Batini, C.; Lenzerini, M.; Navathe, S.B.: *A Comparative Analysis of Methodologies for Database Schema Integration*. In Computing Surveys 18(4):323-364, 1986.

[3] Bernstein, P.A.; Bergstraesser, T.: *Metadata Support for Data Transformation Using Microsoft Repository*. In [26]:9-14, 1999

[4] Bernstein, P.A.; Dayal, U.: *An Overview of Repository Technology*. Proc. 20th VLDB, 1994.

[5] Bouzeghoub, M.; Fabret, F.; Galhardas, H.; Pereira, J; Simon, E.; Matulovic, M.: *Data Warehouse Refreshment*. In [16]:47-67.

[6] Chaudhuri, S., Dayal, U.: *An Overview of Data Warehousing and OLAP Technology*. ACM SIGMOD Record 26(1), 1997.

[7] Cohen, W.: *Integration of Heterogeneous Databases without Common Domains Using Queries Based Textual Similarity*. Proc. ACM SIGMOD Conf. on Data Management, 1998.

[8] Do, H.H.; Rahm, E.: *On Metadata Interoperability in Data Warehouses*. Techn. Report 1-2000, Department of Computer Science, University of Leipzig. http://dol.uni-leipzig.de/pub/2000-13.

[9] Doan, A.H.; Domingos, P.; Levy, A.Y.: *Learning Source Description for Data Integration*. Proc. 3rd Intl. Workshop The Web and Databases (WebDB), 2000.

[10] Fayyad, U.: *Mining Database: Towards Algorithms for Knowledge Discovery*. IEEE Techn. Bulletin Data Engineering 21(1), 1998.

[11] Galhardas, H.; Florescu, D.; Shasha, D.; Simon, E.: *Declaratively cleaning your data using AJAX*. In Journees Bases de Donnees, Oct. 2000. http://caravel.inria.fr/ galharda/BDA.ps.

[12] Galhardas, H.; Florescu, D.; Shasha, D.; Simon, E.: *AJAX: An Extensible Data Cleaning Tool*. Proc. ACM SIGMOD Conf., p. 590, 2000.

[13] Haas, L.M.; Miller, R.J.; Niswonger, B.; Tork Roth, M.; Schwarz, P.M.; Wimmers, E.L.: *Transforming Heterogeneous Data with Database Middleware: Beyond Integration*. In [26]:31-36, 1999.

[14] Hellerstein, J.M.; Stonebraker, M.; Caccia, R.: *Independent, Open Enterprise Data Integration*. In [26]:43-49, 1999.

[15] Hernandez, M.A.; Stolfo, S.J.: *Real-World Data is Dirty: Data Cleansing and the Merge/Purge Problem*. Data Mining and Knowledge Discovery 2(1):9-37, 1998.

[16] Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: *Fundamentals of Data Warehouses*. Springer, 2000.

[17] Kashyap, V.; Sheth, A.P.: *Semantic and Schematic Similarities between Database Objects: A Context-Based Approach*. VLDB Journal 5(4):276-304, 1996.

[18] Lakshmanan, L.; Sadri, F.; Subramanian, I.N.: *SchemaSQL - A Language for Interoperability in Relational Multi-Database Systems*. Proc. 26th VLDB, 1996.

[19] Lee, M.L.; Lu, H.; Ling, T.W.; Ko, Y.T.: *Cleansing Data for Mining and Warehousing*. Proc. 10th DEXA, 1999.

[20] Li, W.S.; Clifton, S.: *SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks*. In Data and Knowledge Engineering 33(1):49-84, 2000.

[21] Milo, T.; Zohar, S.: *Using Schema Matching to Simplify Heterogeneous Data Translation*. Proc. 24th VLDB, 1998.

[22] Monge, A. E.: *Matching Algorithm within a Duplicate Detection System*. IEEE Techn. Bulletin Data Engineering 23(4), 2000 (this issue).

[23] Monge, A. E.; Elkan, P.C.: *The Field Matching Problem: Algorithms and Applications*. Proc. 2nd Intl. Conf. Knowledge Discovery and Data Mining (KDD), 1996.

[24] Parent, C.; Spaccapietra, S.: *Issues and Approaches of Database Integration*. Comm. ACM 41(5):166-178, 1998.

[25] Raman, V.; Hellerstein, J.M.: *Potter's Wheel: An Interactive Framework for Data Cleaning*. Working Paper, 1999. http://www.cs.berkeley.edu/ rshankar/papers/pwheel.pdf.

[26] Rundensteiner, E. (ed.): Special Issue on Data Transformation. IEEE Techn. Bull. Data Engineering 22(1), 1999.

[27] Quass, D.: *A Framework for Research in Data Cleaning*. Unpublished Manuscript. Brigham Young Univ., 1999

[28] 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.

[29] Savasere, A.; Omiecinski, E.; Navathe, S.: *An Efficient Algorithm for Mining Association Rules in Large Databases*. Proc. 21st VLDB, 1995.

[30] Srikant, R.; Agrawal, R.: *Mining Generalized Association Rules*. Proc. 21st VLDB conf., 1995.

[31] Tork Roth, M.; Schwarz, P.M.: *Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources*. Proc. 23rd VLDB, 1997.

[32] Wiederhold, G.: *Mediators in the Architecture of Future Information Systems*. In IEEE Computer 25(3): 38-49, 1992.