DATA CLEANING
Author: BALWANT RAI
Organization: Evaltech, Inc.
Evaltech Research Group,
Data Warehousing Practice.
Date: 07/12/2004
Email: erg@evaltech.com

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.

Intellectual Property / Copyright Material

All text and graphics found in this article are the property of the Evaltech, Inc. and cannot be used or duplicated without the express written permission of the corporation through the Office of Evaltech, Inc.
ETL Approach to Clean Data

Introduction

Companies live and die by the intelligence they can draw out of their data. Intelligence is derived using a combination of data warehousing, advanced analytics and business intelligence. But that intelligence is only as good the quality of the data itself. Drawn from a variety of platforms, formats and even physical locations, today companies need to merge their data warehousing and data quality activities to achieve a ‘rapid return on intelligence’. Many believe that making the most of an organization’s data assets is easier said than done. This is why data quality is often ignored – everyone knows it’s a problem, but perhaps having an idea of the extent of the problem they go into denial about the impact that it has on the overall organization. So, what does data quality mean to business? It can be as simple as making sure that you get the retail catalog that you want, and also ensuring that you don't want an extra copy for every other way that your name or address can be misspelled. The lack of data quality can mean the nightmare of the disappearing customer.

Poor quality information is detrimental to any organization. One of the critical foundations for the effective warehousing and mining of volumes of data is data quality. If data is of insufficient quality, then the knowledge workers who query the data warehouse and the decision-makers that receive the information cannot trust the results. Your data is raw and sometimes unusable. Maybe duplicate names with slight differences cut too deep into your marketing budget and should be consolidated. Maybe you spend hours trying to organize and correct address data. But, when you obtain new data every month, you must correct it, again.

Functionally, most organizations have separate sales, support and marketing groups. It is extremely difficult for businesses to get their hands around all of the customer processes and data infrastructure issues simultaneously.

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 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. 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), further data transformations deal with schema/data translation and integration, and with filtering and aggregating data to be stored in the warehouse. 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.
ETL Approach to Clean Data

ETL provides the keys to keeping data relevant

Managing Big Data
Can you, at this moment in time, imagine managing 500 terabytes of data? Or integrating billions of clicks from your web site with data from multiple channels and business units every day, may be more than once a day? It seems like an extreme scenario, but it’s one that industry analysts uniformly predict organizations will be confronting within the next three to four years. The exact figures may vary slightly, but the consensus is solid: enterprises are going to be swamped with data, and they’re going to have to figure out how to manage it or risk being left behind.

A rapid increase in data volume is not the only challenge enterprises will face. End users want more information at more granular levels of detail, and they want flexible, integrated, timely access. The number of users and the number of queries are also growing dramatically. Additionally, organizations are placing more emphasis on high-value, data-intensive applications such as CRM. All of these developments pose problems for enterprise data management.

Fortunately, there is an effective answer to these problems—scalable data solutions, and more specifically, scalable ETL environments.

Scalability is defined as the retention or improvement of an application’s performance, availability, and maintainability with increasing data volumes. This paper will explore the three dimensions of scalability as they relate to ETL environments, and will suggest some techniques that IT organizations can use to ensure scalability in their own systems.

In the case of performance, scalability implies the ability to increase performance practically without limit. Performance scalability as a concept is not new, but actually achieving it is becoming much more challenging because of the dramatic increases in data volume and complexity that enterprises are experiencing on every front. How long will users tolerate growing latency in the reporting provided to them? How long can enterprises keep adding hardware, installing new software, tweaking or building from scratch new applications? The fact is, yesterday’s scalable solutions aren’t working in the new environment.

The extract, transform and load (ETL) environment poses some especially difficult scalability challenges because it is constantly changing to meet new requirements. Enterprises must tackle the scalability problem in their ETL environments in order to successfully confront increasing refresh frequencies, shrinking batch windows, increasing processing complexity, and growing data volumes. Without scalability in the ETL environment, scalability in the hardware or database layer becomes less effective. In terms of implementing scalable data solutions, enterprises should adopt a "build it once and build it right" attitude. If an ETL environment is designed to be scalable from the start, the organization can avoid headaches later. Let’s consider a situation in which this is not the case, and the ETL environment is architected without consideration for scalability. The first generation of this solution will be fine until data volumes exceed capacity. At that point, the organization will be able to make the fairly easy move to a second-generation environment by upgrading hardware and purchasing additional software licenses. Once this solution is no longer sufficient, however, the enterprise will find it more difficult and costly to evolve to a third-generation solution, which usually involves custom programming and buying point solutions.

Finally, once the third-generation solution has reached its limits, the enterprise will need to rebuild its ETL environment entirely, this time using scalable and parallel technologies. Clearly, enterprises can save time and money by implementing a scalable ETL environment from the very beginning.

Effective data extract, transform and load (ETL) processes represent the number one success factor for your data warehouse project and can absorb up to 70 percent of the time spent on a typical warehousing project. ETL tools promise quick results, improved manageability and meta data integration with other common design and implementation tools. However, due to the potentially huge amounts of money involved in a tool decision, choosing the correct ETL tool for your project can present a daunting challenge. With a bit of internal questioning in advance followed by a careful review of your key needs against the choices available on the market, you should be able to choose the most effective ETL tool for your project.
ETL tools perform, as you may guess, at least three specific functions all of which focus around the movement of data from one place (file type, server, location, etc.) or system to another. More encompassing than a simple file copy process, this class of software generally reads data from an input source (flat file, relational table, message queue, etc.); passes the stream of information through either an engine- or code-based process to modify, enhance, or eliminate data elements based on the instructions of the job; and then writes the resultant data set back out to a flat file, relational table, etc. As you may have guessed, these three steps are known as extraction, transformation and loading, respectively.

**ETL Tools** – These "complete" ETL tools provide a rich mix of functionality and connectivity, but may be significantly more expensive than tools found in the other categories. For extremely complex projects or those attempting to process massive amounts of data, these tools may present the only true option for ensuring success in the ETL phase of the project. In other cases, this class of tool may offer features that simply are not required in the existing environment.

Three key processes commonly used to move data from one application or system to another are known as ETL, for extraction, transformation, and loading.

The goal of virtually every ETL application is to make data available to its audience in a timely fashion. Traditionally, businesses have relied on ETL routines to transfer data from old applications to new ones, or to move operational data into business intelligence systems, such as data warehouses and data marts.

Due to the explosive growth of the Internet, however, ETL processes are now commonly used to support Web applications as well. For example, a manufacturer might use ETL routines to load a Web-based order-status system with production data from an internal legacy application. Similarly, a retailer might use ETL routines to relay order data from its online storefronts to its suppliers. ETL programs have become essential components of many e-commerce initiatives, including business-to-business and business-to-consumer applications.

In broad terms, ETL applications extract data from a source database, transform the data into a format suitable for a target database, and then load the data into the target database. In this analysis, the InfoWorld Test Center provides an overview of ETL processing.

**Data extraction**

To initiate an ETL process, programmers use extraction routines to read records in a source database and make the data in those records available for transformation processing. To extract data from a source database, programmers have three choices: They can write customized programs, rely on specialized ETL tools, or use a combination of both.

In practice, most programmers bolster third-party tools with customized programs. Referred to as user exits, these programs perform specialized functions that are unique to each environment.

Third-party ETL products are typically more effective and less costly than customized programs. Many ETL tools provide programmers with a single, intuitive interface that can be used to extract data from multiple database products. Businesses that rely on a hodgepodge of databases will benefit from unified access to those products.

ETL tools are also preferable to custom programs because they can be used "out of the box"; there is no need to write code to open files, read records, and join tables -- these products perform those functions for you. Furthermore, leading ETL products include prebuilt extraction
ETL Approach to Clean Data

routines designed for popular ERP (enterprise resource planning) applications. Finally, as their acronym implies, ETL tools support not only extraction but also transformation and loading. In many cases, a single ETL tool can meet most of your data-movement requirements.

Data transformation

Once extraction routines have collected the data, transformation routines can prepare that data for its new home. There are several major transformation techniques, including aggregation, value translation, field derivation, and cleansing. Let's use a hypothetical data mart to examine these processes. Note, however, that the use of these routines is not limited to data marts.

Data marts are specialized applications that allow end-users to analyze broad trends in a highly intuitive manner. For instance, a marketing analyst might use a sales data mart to examine revenue per product for each of the past five years. Unlike order processing, manufacturing, and other operational applications, data marts do not require detailed information. In fact, summarized data is preferable because it reduces response times and enhances ease of use.

Before loading a data mart, programmers typically aggregate data. Aggregation routines replace numerous detail records with relatively few summary records. For example, suppose that a year's worth of sales data is stored in several thousand records in a normalized database. Through aggregation, this data is transformed into fewer summary records that will be written to the sales data mart. Although programmers could write code to manually aggregate data, ETL tools are more efficient because they allow programmers to summarize data in one step, with no coding.

Value translation is another common data-transformation technique. Operational databases store encoded information to limit data redundancy and storage requirements. For example, SKUs (stock-keeping units) may be stored in invoice files because they are shorter than their associated product descriptions, and so on.

Because data marts contain summarized information and are designed for ease of use, programmers typically replace encoded data with clearer descriptions. Although programmers could accomplish this task with custom code, ETL tools are more efficient because they allow programmers to use value-translation lists to decode data.

Field derivation is a third technique used to transform data. Through field derivation, new information is created for end-users. For example, suppose our operational database contains one field for sales quantity and one for unit price. Rather than have end-users calculate revenue, programmers could create a revenue field during transformation. Leading ETL products enable programmers to use mathematical operations, statistical functions, string manipulation, date arithmetic, and conditional logic to derive new fields.

A fourth transformation routine, cleansing, has many uses. Programmers rely on cleansing algorithms to keep inaccurate data out of other systems. For example, cleansing routines typically verify that numeric fields contain numeric data, that dates and numbers are valid and reasonable, and so on. Cleansing routines can also be used in cases when one unique value is represented by a database in several ways. For example, IBM might be depicted as IBM Co., International Business Machines, IBM, etc. During cleansing, multiple versions of the same data element are replaced with a single value. Cleansing can also be used to adjust field attributes so they match those of the target database.

Data loading

after data has been transformed for the target database, programmers use load procedures to write that information to the new database. During this phase, you must determine whether to
ETL Approach to Clean Data

propagate data periodically or continuously. Periodic replenishment occurs regularly, such as daily, weekly, or monthly. Sometimes known as snapshot propagation, this approach captures the state of operational systems at a specific moment. If users require current information, continuous propagation can load data into the target database on a real-time basis. Continuous replication requires a dedicated, high-speed communications channel. Your affinity for periodic vs. continuous replenishment will depend largely on your users' needs for up-to-date information, as well as your system infrastructure and budget.

Most ETL tools support both periodic and continuous data loads. Advanced products also let you perform propagation on a net-change basis, which allows you to transmit only modified data, and consequently requires minimal communications overhead.

You can further categorize data loading by the method used to replicate data. In push replication, the source application "pushes" transformed data to the target application. In pull replication, the target application "pulls" data as it's needed, such as when an end-user runs a query. You can also design load procedures that use both push and pull processes. In this approach, the source application typically pushes the data to a staging database, where it is transformed and then pulled into the target application as needed. This "mixed-mode" approach requires more disk space, but can enhance performance.

Whether you use custom coding, third-party ETL tools, or a combination of the two to move data from one system to another, a thorough understanding of ETL practices, as well as of your source and target applications, is essential to success. Given the importance and price of many ETL solutions, you should also insist on a trial period so that you may test each solution in your environment.

Steps of building a data warehouse:

ETL explained above should be used for process instead on data transformations for schema translation and schema integration. Data is not pre integrated 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 focused on the problem of duplicate identification and elimination. Some research groups concentrate on general problems not limited but relevant to data cleaning, such as special data mining approaches ,and data transformations based on schema matching .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 .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.
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 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 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. 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).

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.

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 designs are naming and structural conflicts. 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, duplicate elimination or the merge/purge problem. Frequently, the information is only partially redundant and the sources may
ETL Approach to Clean Data

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.

Data cleaning approaches

In general, data cleaning involves several phases

1) 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 "dirty ness" 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. 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 re doing the cleaning work for future data extractions. For data warehousing, the cleaned data is available from the data staging area.

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. 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. 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. Name conflicts are typically resolved by renaming; structural conflicts require a partial restructuring and merging of the input schemas.
ETL Approach to Clean Data

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 analyze 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. There are two related approaches for data analysis, data profiling and data mining. Data profiling focuses 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.

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 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 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 RDBMS.

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 Schema SQL proposal are required. Data cleaning at the instance level can also benefit from special language extensions such as a Match operator supporting "approximate joins". 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.

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. 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 (birth date - age, total price - unit price / quantity, city - phone area code, can be utilized to detect problems and substitute missing values or correct wrong values.
ETL Approach to Clean Data

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 upper or lower case, etc. Text data maybe condensed and unified by performing stemming, removing prefixes, suffixes, and stop words. Further 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 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 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 else wherein this issue. 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. 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 wild cards, character frequency, edit distance; keyboard distance and phonetic similarity are useful. More complex string matching approaches also considering abbreviations are presented in. 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.

Determining matching instances with such an approach is typically a very expensive operation for large datasets. 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 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.
ETL Approach to Clean Data

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.

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. We first discuss tools for data analysis and data reengineering 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.

**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 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 (Wiz Soft) and DATAMININGSUITE (Information Discovery), 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.

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, and 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.

**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 .

Special domain cleaning: Names and addresses are recorded in many sources and typically have high cardinality.
4.3 ETL tools-

A large number of commercial tools support the ETL process for data warehouses in a comprehensive way, e.g., COPYMANAGER (Information Builders), DATASTAGE (Informix/Ardent), EXTRACT (ETI), power mart (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 these 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.