Ethical Boundaries in Data Cleaning: Issues, Methods, and the Threat of Data Manipulation

Firman Hadi Sukma Pratama[1], Hakkun Elmunsyah[2*] ,Siti Sendari[3*]

Department of Electrical Engineering and Informatics, State University of Malang, Indonesia

***Corresponding Author**

DOI: https://dx.doi.org/10.51244/IJRSI.2025.12110185

Received: 04 December 2025; Accepted: 10 December 2025; Published: 24 December 2025

ABSTRACT

Data integrity is essential to credible scientific research, yet practices such as data cleaning can lead to ethical concerns when conducted without clear methodological justification. This paper examines the ethical limits of data cleaning by reviewing common data issues, valid cleaning techniques, and the risks of manipulation that may compromise research validity. Using a qualitative literature review, the study finds that data cleaning is ethically acceptable only when supported by statistical reasoning, transparent documentation, and reproducible procedures. In contrast, removing or altering data to reinforce hypotheses or improve results constitutes manipulation and violates research ethics. The failure to distinguish these practices risks misinformation, weakened knowledge development, and diminished academic credibility. The findings underscore the need for strict data integrity practices across all research environments.

Keywords— data cleaning, research ethics, data manipulation, data integrity, outliers

INTRODUCTION

Data cleaning also referred to in the literature as *data cleansing* or *data scrubbing*is a fundamental stage in ensuring high-quality data. It encompasses the detection, correction, and removal of erroneous or inconsistent entries. Single-source data such as text files, spreadsheets, or relational databases are highly susceptible to quality degradation caused by typographical errors, incomplete information, inconsistent formats, or inaccurately recorded values [15]. These issues become far more complex when data originate from multiple heterogeneous systems and must be consolidated into a unified representation. In multi-source integration scenarios, differences in representation, duplicated records, and inconsistent storage standards emerge as major challenges requiring robust cleaning strategies [2][24]

The need for systematic data cleaning becomes even more critical in data warehouse environments, which routinely ingest large volumes of data from diverse operational systems. Each loading process introduces the risk of importing data that are invalid, incomplete, or inconsistent. Because data warehouses support strategic decision-making, even minor inaccuracies can lead to misleading analytical outcomes a problem often summarized by the phrase “garbage in, garbage out” [6]. The scale and heterogeneity of incoming data make cleaning one of the most demanding components of modern data warehouse operations [5][16]

Within the ETL (Extraction, Transformation, and Loading) pipeline, data from multiple sources are extracted, translated into a common format, integrated, filtered, and prepared for storage in the central repository like in figure 2. Data cleaning is typically performed in a staging area to ensure that only validated and standardized data are passed to the warehouse. Although ETL tools have become increasingly advanced, a substantial portion of cleaning operations still require manual intervention or adaptive scripts that are difficult to maintain [3][8]

In contrast to data warehouses, federated database systems and web-based information environments rely on a wrapper–mediator architecture to dynamically access and combine data from multiple distributed sources [31][32]. In such environments, data are not pre-cleaned or standardized prior to integration; instead, extraction, minimal cleaning, and transformation occur at query time. This runtime approach increases

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

processing overhead because the system must simultaneously manage format inconsistencies, semantic differences, and varying data quality [7][13]. Despite the computational cost, some degree of cleaning remains necessary to ensure reliable query results.

An ideal data cleaning strategy must be capable of identifying a wide spectrum of errors—ranging from structural inconsistencies to instance-level anomalies—and addressing them effectively in both single-source and multi-source contexts. In practice, this process requires tool support that minimizes manual inspection and provides extensibility for incorporating new data sources without redesigning the entire workflow. Integrating data cleaning with metadata-driven transformation enables reusable mapping functions across various processing tasks, including integration and query answering [3][4][20]

While topics such as schema integration and structural transformation have been extensively studied in the data management community [21][8] comprehensive research on end-to-end data cleaning remains relatively limited. Earlier work primarily focused on isolated issues such as duplicate detection or entity matching [11][15][23], whereas more recent research efforts are moving toward unified frameworks that incorporate multiple cleaning phases, specialized operators, and more efficient execution strategies [12][19][25]. This trend indicates that data cleaning is evolving from a secondary, supportive activity into a central component of the modern data management lifecycle [27][28].

A.Data Cleaning Problems

Data quality issues arise in many forms and often interact with one another, making data cleaning a process that cannot be addressed in isolation. Instead, it must be understood as a sequence of interrelated activities involving both cleaning and transformation steps. Data transformation plays a key role in aligning the structure and representation of data, particularly when system requirements change or when heterogeneous sources must be merged into a unified dataset [9]

In the data management literature, problems associated with data cleaning are generally grouped into two major categories: single-source problems and multi-source problems. Additionally, these issues may appear at two different layers: the schema level and the instance level [15] Schema-level issues pertain to the design and representation of data structures, such as inappropriate data types, inconsistent formatting rules, or schema changes resulting from system evolution. Such inconsistencies often lead to structural mismatches across sources. Addressing these issues typically involves techniques such as schema evolution, schema translation, or schema integration [2]

Conversely, instance-level problems originate from the actual data values and are not visible at the schema layer. Common examples include typographical errors, missing entries, inconsistent formats, duplicate records, and invalid values. These issues constitute the primary focus of data cleaning because they directly affect the accuracy and reliability of analytical outcomes. As Christen [19] emphasizes, instance-level inconsistencies are a major source of statistical bias and misinterpretation. In figure 2 although instance-level issues may occur within a single dataset, their complexity increases significantly in multi-source environments. Integrating heterogeneous sources introduces additional challenges such as differences in semantic interpretation, varying formatting standards, cross-source duplication, and attribute-level mismatches. These inconsistencies make the integration process more error-prone and considerably more difficult to manage [9][21]

Given the diversity and interconnected nature of these problems, data cleaning must be approached comprehensively rather than addressing individual issues in isolation. Both structural inconsistencies at the schema level and anomalies at the instance level must be considered simultaneously. A holistic approach ensures that data quality is preserved before the data are used for analysis, modeling, integration, or decision-making processes.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

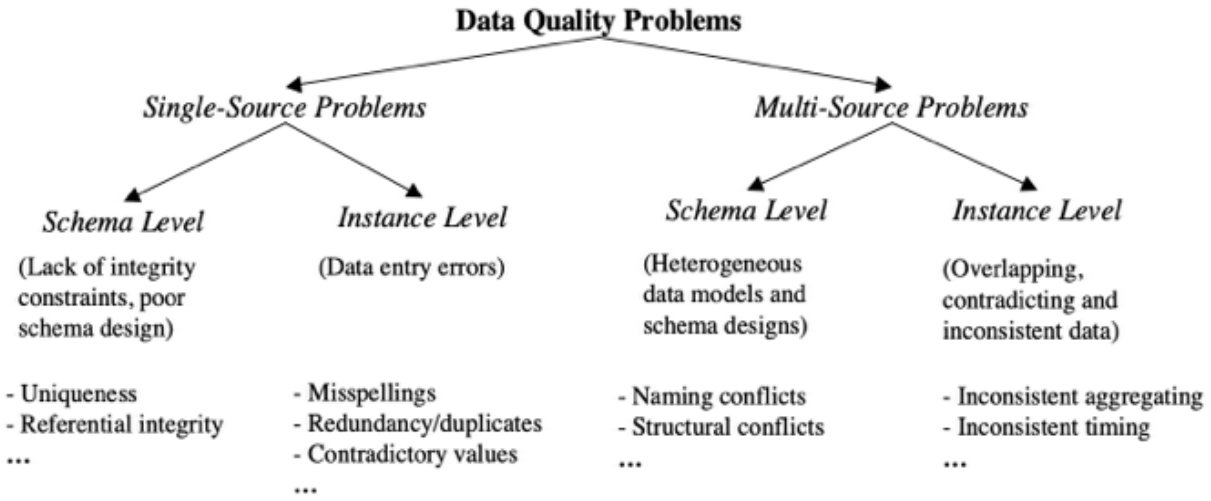

Figure 1. Data Quality Problem

B.Single-Source Data Quality Problems

Data quality issues that arise within a single data source are largely determined by how that source is designed, structured, and governed. The more stringent the rules and constraints that regulate the format and permissible values of the data, the lower the likelihood of errors. In contrast, data sources that lack a formal schema—such as plain text files, CSV datasets, or simple spreadsheets—often store information in a "free-form" manner without enforcing data types, formatting standards, or value constraints. As a result, these sources are highly prone to incorrect, incomplete, or inconsistent entries because no mechanisms exist to prevent erroneous input.

Modern database systems, on the other hand, typically employ a set of structural safeguards, including strict attribute typing, referential integrity rules, and application-level constraints. These mechanisms act as the first line of defense to prevent invalid or improperly structured data from entering the system. However, quality problems may still arise when integrity constraints are poorly defined, when the schema design is inadequate, or when systems deliberately minimize the use of constraints to reduce processing overhead. Under such conditions, structural errors remain possible despite the presence of a formal schema framework [15]

Problem Category	**Example of Dirty Data**	**Explanation / Likely Cause**
Attribute Value Error	birth_date = 31/14/2010	The attribute contains a value outside the valid domain (there is no 14th month). Errors may arise from typographical mistakes or non-standard date formats.
Attribute Dependency Violation	age = 22, birth_date = 10/03/1965	The reported age is inconsistent with the birth date. A functional dependency should exist between age and birth_date based on the current date.
Entity Duplication / Uniqueness Violation	Record 1: *(Name: "Ahmad Rizki", ID: 987654321)*Record 2: *(Name: "A. Rizki", ID: 987654321)*	Two different rows share the same unique identifier, indicating duplicated entities or incorrect identity recording.
Referential Integrity Violation	employee = (name="Siti Rahma", dept_id=45) but department 45 does not exist in the departments table	A child record references a parent entity that is missing. This indicates relational integrity issues or incomplete data in reference tables.
Inconsistent Data Representation	phone = "0812-3456-7890" vs phone = "+62 812 3456 7890"	Data originating from different sources use inconsistent formatting. Standardization is needed to unify telephone number formats.
Missing Values	address = NULL,	Required fields are empty or not recorded. Missing

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

	postal_code = ""	values hinder matching, analysis, and data integration processes.
Format Inconsistency	salary = "3.500.000" vs salary = "3500000"	The numeric representation differs (with or without grouping separators), complicating parsing and aggregation operations.

Tabel 1. Example of Dirty Data

In addition to structural issues, data quality problems also arise at the instance level, where the errors originate directly from the recorded values themselves. These issues occur when incorrect, incomplete, or inconsistent data are entered into the system and cannot be prevented solely through a well-designed schema. Common examples include typographical errors, incorrect numerical values, inconsistent date formats, and repeated entries that introduce redundancy into the dataset [15][19] emphasize that instance-level anomalies are often the primary contributors to overall data degradation and can only be corrected through direct data cleaning procedures rather than structural adjustments.

Thus, even though a single-source dataset may appear simpler than a multi-source integration scenario, it can still contain numerous anomalies that complicate downstream processing and analysis. Errors originating from a single source act as the “seed” of larger problems; if left unaddressed, they may propagate into significant inconsistencies once the data are integrated, aggregated, or subjected to analytical operations.

Problem Category	**Example of Problematic Data**	**Explanation & Common Causes**
Missing Values	phone = NULL or postal_code = "00000"	The required value was not provided during data entry. Missing values are often replaced with dummy values or left blank due to user uncertainty.
Misspellings	city = "Bandunng"	Typographical errors, phonetic mistakes, or rushed input result in invalid place names.
Cryptic or Abbreviated Values	experience = "A" or job = "Sys Eng."	Use of internal codes, abbreviations, or non-standard shorthand creates ambiguity during data integration.
Embedded Values in a Single Attribute	address = "Jl. Melati 12, Bandung, 40222" stored in one column	Multiple logically distinct elements (street, city, postal code) are combined in a single field, often due to free-text input.
Misfielded Values	city = "Indonesia"	Information is entered into the wrong attribute (e.g., country placed in the city field). Common in multi-line or poorly structured forms.
Attribute Dependency Violation	city = "Denpasar", zip = "40111"	Attributes that should be logically consistent do not match. The postal code does not correspond to the city.
Word Transposition	name = "Sari Putri" vs. name = "Putri Sari"	The order of words is reversed, especially in free-text fields. This often leads to undetected duplicates.
Duplicate Records	Record 1: name = "A. Rahman"Record 2: name = "Abdul Rahman"	The same individual appears twice due to differences in spelling, abbreviation, or formatting.
Conflicting Records	Data A: name = "Dewi Lestari", bdate = "12-01-1990"Data B: name = "Dewi Lestari", bdate = "21-10-1990"	Two records refer to the same entity but contain contradictory values, indicating entry errors or discrepancies between sources.
Wrong References	employee(name="Budi", dept_id=10) but dept_id 10 refers to a different department	The reference points to an existing entity, but the relational link is incorrect (e.g., wrong unit or wrong reference code).

Tabel 2. Problematic Data

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

For both schema-level and instance-level issues, data quality problems can be examined across several scopes, including the attribute (field), record, record type, and source levels; illustrative examples for each category are provided in Tables 1 and 2. It is important to note that even when uniqueness constraints are defined at the schema level, they do not fully prevent duplicate instances particularly when information about the same real-world entity is entered multiple times using different attribute values, as demonstrated in Table 2 [15]

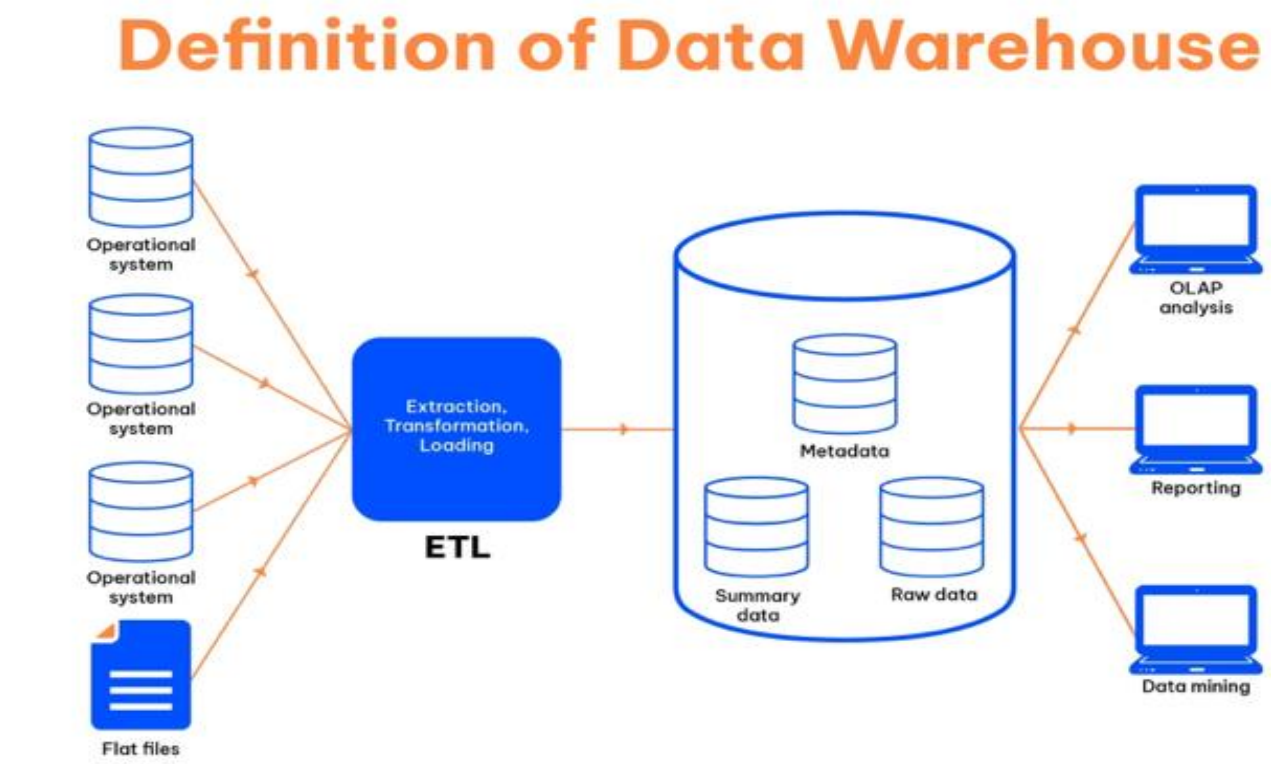

Figure 2 . ETL Process

Cleaning data after errors have already occurred is a resource-intensive process that demands substantial time, effort, and computational cost. Consequently, the most effective strategy in data quality management is to prevent dirty data from entering the system in the first place. Preventive measures include designing stricter database schemas, enforcing appropriate integrity constraints, and developing input interfaces that minimize user mistakes. Moreover, the process of designing and maintaining a data warehouse often reveals recurring patterns of data errors, enabling organizations to refine validation rules and strengthen integrity constraints within existing systems. These improvements contribute to more robust data quality practices aligned with current operational demands [5][6][16]

D. Approaches to Data Cleaning (Overview)

Data cleaning typically involves a sequence of interrelated phases designed to detect, transform, validate, and propagate corrected data. Because data inconsistencies manifest in many forms, the process must be executed holistically rather than through isolated operations (Savasere et al., 1995 29; Quass, 1999 27). Below is a detailed outline of the major components of a data cleaning workflow.

1. Initial Data Analysis

The first stage involves performing an in-depth analysis of the available data sources to determine which types of errors exist. This analysis requires both manual inspection and automated tools that generate metadata and statistical indicators revealing quality issues. Such analysis facilitates the detection of value-range violations, abnormal distributions, duplicate frequencies, and inconsistent formats problems often invisible from schema information alone [10] [20].

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

2. Designing the Transformation Workflow and Mapping Rules

Once data problems have been identified, the next step is designing a transformation workflow that specifies the sequence of cleaning operations. The complexity of this workflow depends on the number of sources, the degree of heterogeneity, and the severity of data anomalies.

Typical activities at this stage include:

1. **Schema translation**, required when sources follow different data models and must be aligned to a target model such as a relational schema used in data warehouses [1].

2. **Single-source cleaning**, addressing localized issues such as inconsistent formats, missing values, or malformed entries.

3. **Multi-source cleaning**, such as schema harmonization, entity matching, and duplicate elimination [15].

In data warehouse environments, these operations are typically orchestrated through a well-defined ETL workflow [6]. Declarative transformation languages are preferred to facilitate automatic code generation and maintainability. Nevertheless, the system must also interface with user-defined scripts or external modules for complex cases not covered by built-in mechanisms [13][14].

User interaction may be required for ambiguous cases where automated decision-making is insufficient.

3. Verification and Validation of Cleaning Results

Each designed transformation must be tested to ensure that the cleaning process is effective. Testing often uncovers hidden anomalies that only become apparent after initial corrections. Consequently, data analysis, workflow design, and verification typically occur iteratively until the resulting dataset achieves the required level of quality [16].

4. Execution of Transformations

This stage corresponds to the actual implementation of the transformation workflow. In data warehouses, execution occurs during ETL-based loading or refresh cycles. In multi-source or federated environments, transformations may be executed during query processing, since these systems often lack a staging area [32].

5. Backflow of Cleaned Data

When cleaning identifies errors originating from a single source, the corrected values should ideally be propagated back to the original system. This prevents reoccurrence of the same issues in subsequent extraction processes and ensures that legacy applications also benefit from improved data quality[5].

6. Importance of Metadata and Documentation

All cleaning stages depend heavily on comprehensive metadata, including schema definitions, instance characteristics, mapping rules, and workflow descriptions. Maintaining this metadata in a repository-based system enhances consistency and facilitates reusability [3]. Detailed documentation especially concerning data lineage, completeness, update timestamps, and transformation history is essential. Lineage tracking enables users to trace cleaned records back to their origins and understand what transformations were applied [11].

Further details regarding conflict analysis, transformation design, and conflict resolution are discussed in the sections that follow. For comprehensive discussion of schema translation and schema integration, foundational works by [2][9] and [15] remain key references.

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

E. Data Analysis

Schema-level metadata alone is often insufficient for assessing data quality. Even when integrity constraints are enforced, many quality issues become visible only after examining the actual data instances. Thus, instance-level exploration is essential to uncover unusual patterns, detect anomalies, and produce reengineered metadata that reflects the true state of the dataset [30].

Two analytical approaches are widely used:

1. Data Profiling

Data profiling examines attribute-level statistics to extract factual information such as:

- dominant data types,
- common string lengths,
- value ranges,
- discrete value sets and frequencies,
- uniqueness ratios,
- proportions of missing values,
- recurring patterns (e.g., phone-number formats).

These metadata elements provide actionable insights into attribute quality. For example, excessively wide value ranges may signal input errors, while high uniqueness may indicate inconsistent formatting. Profiling therefore forms a crucial foundation for designing appropriate cleaning strategies

Type of Problem	Metadata Examined	Detection Indicators / Techniques
Unusual Number of Categories	Attribute cardinality	If the number of gender categories exceeds two or contains unexpected values beyond “M/F,” the system flags a potential anomaly.
Out-of-Range Values	Minimum & maximum values	Examples include age < 0 or salary exceeding reasonable upper bounds, indicating invalid entries.
Domain Violations	Domain rules & valid ranges	Values such as month = 15 or grade = 12 (on a 0–10 scale) clearly violate predefined domain constraints.
Statistical Deviations	Variance & standard deviation	Extremely high variance for certain attributes may suggest outliers or erroneous recordings.
Misspellings	Value distribution & alphabetical sorting	Sorting city names may place misspelled entries (e.g., “Bandunng”) next to their correct forms (“Bandung”), enabling easier detection.
Missing Values	Null counts & null patterns	A high percentage of nulls may indicate systematic data loss or incomplete data entry processes.
Suspicious Default Values	Default-value detection	The appearance of values such as “00000” for postal codes or “999999” for phone numbers suggests missing true information.
Inconsistent Value Representations	Cross-column or cross-table value set comparison	Example: marital_status = "M" in one table but "Married" in another → signals a need for standardization.
Duplicate Records	Count of unique values & frequency distribution	If the number of unique entries is smaller than the total number of rows, or if certain values appear multiple times, duplication is likely.
Cross-Source Inconsistencies	Metadata comparison across sources	When value sets across two related sources differ in format or representation, it indicates potential inter-

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

		source conflicts.

Tabel 3. Metadata Examined

F.Multi-Source Integration Problems

Data quality challenges become significantly more complex once information originates from multiple heterogeneous systems. While single-source datasets typically exhibit issues such as typos, inconsistent formats, or missing values, multi-source integration amplifies these problems because each source is developed independently, follows its own operational goals, and adopts distinct recording standards and storage structures. This independence leads to high degrees of heterogeneity in schema design, data models, semantic conventions, and data quality levels [2]

At the schema level, multi-source environments often face fundamental structural inconsistencies. These include naming conflicts where two different concepts share the same name (homonyms) or where a single concept is represented using multiple names (synonyms) as well as structural conflicts involving divergent representations of the same real-world entity. For instance, attributes may be represented as separate tables in one system but embedded as composite fields in another; or sources may employ different data types, integrity constraints, and decomposition granularity. Such conflicts necessitate schema translation and schema integration before any meaningful data cleaning can take place [15]

Beyond schema discrepancies, even more challenging inconsistencies arise at the instance level. Typical single-source problems such as misspellings, duplicate entries, variant date formats, or missing values reappear in multi-source settings but often with additional complexity due to differences in semantic conventions across systems. For example, marital status may be encoded as {M, S} in one source and {Married, Single} in another; income may be recorded in different currencies; and sales data may reflect different levels of aggregation (e.g., per product versus per product group). Temporal misalignment is also common—one system may store current-month data, while another retains last-quarter snapshots [9]

One of the most critical challenges is determining whether two or more records describe the same real-world entity. This issue—commonly referred to as entity resolution, record linkage, or the merge/purge problem—has been widely studied in the context of data quality [15].

Frequently, duplicate-looking records contain complementary information rather than redundant content; for instance, one source may store the most recent address, while another may provide transaction history that does not exist elsewhere. Thus, multi-source integration requires not only the elimination of redundant duplicates but also the consolidation of complementary values to produce a complete, accurate representation of each entity.

G.Schema and Instance Conflicts Illustrated

The example shown in *Figure 3* demonstrates how two relational sources may exhibit discrepancies at both the schema and instance levels. Schema-level conflicts include naming inconsistencies such as *Customer* versus *Client, Cid* versus *Cno,* and *Sex* versus *Gender* as well as structural differences, especially in the representation of composite fields like names and addresses. Such misalignments reflect typical schema-level heterogeneity discussed in the literature [24][2]

Instance-level inconsistencies are also evident. For example, gender values are encoded numerically (*0/1*) in one source and alphabetically (*F/M*) in the other. Differences in coding schemes require standardization before matching or integration can occur. The repeated appearance of "Kristen Smith" with variations across sources also signals potential duplication. Importantly, identifiers such as *Cid* and *Cno* serve only as source-specific keys and cannot be directly compared; different numbers may refer to the same person, and identical numbers may refer to unrelated individuals [8]. This reinforces the necessity of schema integration before applying duplicate detection or entity matching.

To resolve these issues, integration and cleaning must be performed in two phases:

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

(1) **schema integration**, reconciling structure and naming; and

(2)**instance-level cleaning**, including standardization, duplicate detection, and attribute normalization.

As illustrated in the final integrated table, schema conflicts are harmonized first so that downstream cleaning operations can be performed consistently.

H. Data Cleaning Approaches

Data Analysis

Schema metadata alone is insufficient for understanding data quality, especially when only minimal integrity constraints are enforced. Therefore, instance-level analysis is essential for uncovering unexpected patterns, anomalies, and implicit rules within the data. Such reengineered metadata facilitates not only the identification of data quality issues but also schema matching across heterogeneous sources [20][9]

Two main techniques are used in practice: data profiling and data mining**.** Data profiling provides a detailed examination of attribute-level characteristics, such as:

- dominant data types,
- typical character lengths,
- value distributions and ranges,
- frequency of discrete values,
- percentage of null or missing values,
- string patterns (e.g., telephone formats or ZIP codes),
- uniqueness constraints, and
- variance of numeric attributes.

Such statistics help uncover anomalies. e.g., extreme variance may signal erroneous values, unusual cardinality may indicate inconsistent coding, and alphabetical sorting may reveal misspellings. Profiling is therefore a foundational step in designing effective cleaning strategies.

Use of Metadata in Schema Matching

Metadata derived from profiling not only identifies data defects but also plays a crucial role in schema matching. When two sources contain attributes that appear different on the surface, metadata patterns such as similar value distributions, shared string formats, or correlated numeric ranges can suggest that the attributes serve equivalent semantic functions. This allows automated or semi-automated mapping across schemas, an essential capability for large-scale data integration [8][9]

Role of Data Mining in Identifying Data Quality Issues

Beyond profiling, data mining techniques support the discovery of hidden patterns in large datasets. Descriptive models including clustering, association rule mining, summarization, and sequence analysis help identify functional dependencies and implicit business rules that were never explicitly encoded [10][29][30]

Patterns discovered by data mining can be used to:

- infer missing values,
- correct implausible entries,

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

- detect duplicates across sources, and
- validate records that violate inferred rules.

For example, if association mining detects that *total* ≈ *quantity* × *unit_price* with 99% confidence, then the remaining 1% of records likely contain data-entry errors and should be inspected [28]

I. Defining Data Transformations

Data transformation workflows typically consist of multiple steps addressing both schema-level and instance-level modifications. To support automation and reproducibility, these transformations are best specified using declarative mapping languages.

Modern ETL tools support proprietary rule languages; however, SQL augmented by user-defined functions (UDFs) remains the most flexible and interoperable approach. Under SQL:99, UDFs enable developers to

encode complex cleaning logic, including:

- pattern-based extraction,
- standardization routines,
- domain-specific correction rules,
- approximate matching functions, and
- attribute reconstruction [13][14]

Advantages of UDFs include:

1. **Reusability** across multiple transformations and queries.
2. **Execution inside DBMS engines**, reducing data transfer overhead and improving performance.
3. **Portability**, as UDFs are part of SQL:99 and supported by major database platforms.

However, UDFs require manual coding and are not always sufficient for complex schema-level restructuring. Extensions such as **SchemaSQL** have been proposed to support generic schema transformations [18]. Similarly, instance-level cleaning can benefit from advanced query operators such as approximate joins and match operators [12][23]

K. Tool Support

A wide range of software tools is available on the market to support data cleaning and data transformation, particularly in the context of data warehouse environments. Some tools are highly specialized, focusing on narrow tasks such as name-and-address standardization or duplicate detection. Because of their limited scope, these domain-specific tools typically perform very well in their niche but often need to be combined with other components to cover the full spectrum of transformation and cleaning activities [11][12][19].

On the other hand, comprehensive ETL platforms provide broader capabilities for data extraction, transformation, and workflow orchestration. However, these systems frequently rely on proprietary APIs and internal metadata formats, which restrict interoperability and make it difficult to integrate multiple tools or to deploy solutions across heterogeneous platforms [3][8][13][26].

J. Tools for Data Analysis and Data Reengineering

Following the classification in Section 3.1, data analysis tools for assessing data quality can broadly be divided into data profiling tools **and** data mining tools**.**

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

- **Data Profiling Tools**

Commercial profiling tools are designed to compute "real" metadata for each attribute, including data type, maximum length, number of distinct values, frequency distributions, minimum and maximum values, and the proportion of missing entries. These statistics not only provide an empirical description of the data but also help in designing target schemas during data migration or warehouse construction [3][4][8]

- **Data Mining Tools**

Tools for data mining such as those implementing association rule mining, summarization, or clustering are used to discover hidden relationships among attributes and values. Systems of this kind can infer mathematical rules, conditional *if–then* patterns, and spelling-based anomalies, for example pointing out that "Edinbourgh" is highly similar to "Edinburgh" and therefore likely to be a misspelling. Deviations from the discovered rules are then flagged as potential data errors [10][28][29][30]

- **Data Reengineering Tools**

Data reengineering tools exploit the patterns and rules identified during analysis to define and execute cleaning transformations. They typically apply several stages such as parsing, data typing, pattern analysis, and frequency analysis—to produce a tabular summary of value patterns. This summary enables users to select appropriate standardization strategies and to configure the cleaning pipeline in a more informed way [19][27]

To specify cleaning operations, some tools provide their own transformation language with operators for column-level changes (e.g., move, split, delete) and row-level operations (e.g., merge or split records). These operators allow users to implement complex restructuring of legacy data. In addition, many tools support statistically based matching to identify records that represent the same real-world entity. Automated weighting schemes are often used to assign scores to candidate matches, and ranked similarity scores help users decide which record pairs should be treated as true duplicates [15][19][22]

L. Specialized Cleaning Tools

Specialized data cleaning tools generally focus on a specific domain or cleaning phase. Common examples include tools dedicated to name and address data or to duplicate detection and elimination. Their core logic typically relies on large rule libraries that encode known patterns, plus configuration options for user-defined rules. In some cases, rules can also be inferred automatically from schema matching or profiling results [20][21].

1. Domain-Specific Cleaning: Names and Addresses

Name and address data frequently exhibit quality problems due to abbreviations, alternative spellings, and a variety of cultural or regional formats. In CRM systems and other customer-centric applications, maintaining consistent names and addresses is crucial for accurate customer identification and segmentation.

Domain-specific tools in this area commonly:

- parse and decompose names and addresses into structured components,
- validate street names, cities, and postal codes against reference dictionaries,
- normalize formats across heterogeneous sources, and
- provide extensive libraries of pre-defined business rules tailored to address and name data.

Some commercial solutions incorporate hundreds of thousands of rules in their parsing and matching modules, enabling robust detection of irregular patterns while still allowing users to extend or modify the rule base for organization-specific needs [11][19][23].

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

2. Duplicate Detection and Elimination

Another important category consists of tools designed specifically for identifying and removing duplicate records. These tools typically operate on datasets that have already undergone basic format cleaning, so that identifiers and key attributes are in a comparable form.

They usually support:

- multiple similarity measures for attribute comparisons,
- configurable thresholds for match acceptance, and
- user-defined matching rules to capture domain-specific equivalences.

Tools in this category are often used to create "golden records" for customers, products, or other key entities by consolidating overlapping entries and purging redundant records [15][22][23]

M. ETL (Extract–Transform–Load) Tools

A large number of commercial platforms have been developed to support the ETL process in data warehouses. Examples include CopyManager, DataStage, Extract, PowerMart, DecisionBase, Data Transformation Services, MetaSuite, Sagent Solution Platform, and Warehouse Administrator. These systems typically rely on a DBMS-based repository for storing metadata about source schemas, target schemas, transformation mappings, scripts, and workflow definitions [3][6][16][26]

Operational data are extracted from source systems via:

- native DBMS connectors,
- standard interfaces such as ODBC, and
- other data access protocols.

Graphical interfaces are provided to allow users to design transformation flows. Each step can be defined using a rule language supported by the tool, combined with a comprehensive library of built-in conversion functions. Many ETL platforms also provide interfaces for integrating external logic, such as C/C++ routines, into the transformation process [13][14][26]

Execution Engines and Workflow Support

ETL tools generally adopt one of two execution approaches:

1. **Interpreter-based engines**, which read and execute transformation definitions at runtime.
2. **Compiled code**, which translates mappings into executable programs for higher performance.

Most platforms, such as CopyManager, DecisionBase, PowerMart, DataStage, and Warehouse Administrator, include schedulers and workflow engines that manage dependencies among individual jobs. Workflows can also trigger external tools, for example to perform advanced name/address cleaning or duplication removal [3][26]

Cleaning Capabilities in ETL Tools

While ETL platforms are powerful for orchestrating large-scale transformations, their built-in data cleaning capabilities are typically limited. Advanced cleaning logic often needs to be implemented via:

- proprietary APIs,
- custom rule definitions, or

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

- user-defined functions.

Automatic detection of data anomalies is usually not provided. Nonetheless, users can implement their own checks using:

- available metadata,
- aggregation functions (e.g., COUNT, MIN, MAX, variance), and
- conditional expressions in the mapping language.

Standard transformation libraries typically cover:

- data type conversions (e.g., date and time formats),
- string operations (split, merge, replace, substring),
- arithmetic and statistical functions, and
- basic standardization procedures.

Extraction from free-form attributes still tends to require manual specification of delimiters or pattern rules, as this functionality is rarely fully automated.

Instance Matching Support

Support for instance-level matching in ETL tools is generally modest. In most systems it is limited to:

1. standard join operations,
2. simple string comparison functions, and
3. basic pattern search mechanisms.

For more advanced tasks such as similarity-based matching, approximate joins, or probabilistic record linkage—organizations typically rely on external specialized tools or implement custom matching logic via scripts or UDFs [15][7][23].

Some ETL and cleaning tools offer basic support for wildcard matching and phonetic algorithms like Soundex to handle name variations. However, more complex similarity assessment usually depends on user-defined matching functions and correlation logic that can be plugged into the internal transformation library and reused across multiple processes [7][11][25]

CONCLUSIONS

This study provides a comprehensive classification of data quality problems arising both within single-source environments and in multi-source integration settings. It further distinguishes between schema-level and instance-level issues, as each requires distinct methodological treatments.

The discussion reinforces that data transformation and data cleaning should not be viewed as isolated procedures; instead, they must be approached as an integrated workflow that simultaneously addresses structural inconsistencies in schemas and correctness issues within the data itself [15][2].

The paper also reviews the range of commercial tools available to support cleaning processes. Although many of these systems have reached a high degree of maturity, most of them address only fragments of the overall problem. Users still need to perform considerable manual work, define custom rules, or implement additional functions to complete the cleaning pipeline. Moreover, interoperability remains limited because many tools

ISSN No. 2321-2705 | DOI: 10.51244/IJRSI |Volume XII Issue XI November 2025

rely on proprietary APIs and non-standard metadata formats, which complicates integration across platforms [3][8]

Overall, research on data cleaning remains relatively sparse, despite the large number of tools on the market—an indication that data cleaning is both essential and highly challenging. Several directions offer promising opportunities for future research:

1. **Developing more expressive transformation languages.**

A unified language capable of handling both schema-level and instance-level transformations would greatly simplify integration tasks. Operators such as *Match*, *Merge*, and *Mapping Composition* should ideally function consistently across metadata and data layers, supported by shared implementation techniques [18][11]

2. **Optimizing cleaning techniques for heterogeneous and web-based environments.**

Modern information systems especially those that perform on-demand data integration require extremely fast response times. In such contexts, data cleaning approaches must be designed with strict performance constraints in mind [14][32]

3. **Advancing cleaning methods for semi-structured data.**

As the use of XML, JSON, and other flexible formats continues to increase, handling semi-structured data becomes a critical challenge. These formats provide minimal structural constraints, making traditional relational cleaning techniques insufficient. New models and algorithms are needed to manage the inherent irregularities of semi-structured data [24][30]

With the growing diversity of data formats and the rising demand for large-scale integration across modern systems, continued research in data cleaning is essential to ensure the reliability and integrity of information used for analysis and decision-making.

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, Dept. of Computer Science, Univ. 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 Intl. Conf. Database and Expert Systems Applications (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. Computer 25(3): 38-49, 1992.