Knowledge-Guided Data Cleaning: From Knowledge Acquisition to Reliable Decisions

Yixi Zhou
Hong Kong Baptist University
yxzhou@comp.hkbu.edu.hk

***Abstract*—Data cleaning requires identifying errors and determining which changes the available evidence supports. Integrity constraints reveal inconsistencies, while trusted reference data, statistical regularities, and human feedback help locate errors and select corrections. This survey examines how knowledge supports these decisions, connecting its representation, acquisition, and reuse with candidate generation, evidence combination, and repair selection. We compare methods according to how they obtain knowledge, establish its applicability, and respond when evidence conflicts or confidence is insufficient. The comparison distinguishes factual correctness, constraint consistency, and downstream task utility, which lead systems to select different operations and require different evaluation criteria. It also explains how experimental assumptions about known error locations, candidate coverage, and accepted predictions affect comparisons across studies. As machine learning and large language models extend cleaning into table transformation and preparation planning, knowledge must support both local repairs and coordination across processing steps. Building on these connections, we identify research directions in checking knowledge applicability, selecting responses to conflicts, and acquiring information according to unresolved decisions.**

***Index Terms*—Data cleaning, knowledge acquisition, data repair, data preparation**

I. Introduction

Data cleaning identifies and handles quality problems in raw data so that the data meet the needs of subsequent analysis and applications. Missing values, inconsistent formats, and incorrect values are familiar problems, but errors can also involve relationships among attributes and records. A cleaning system therefore needs information about the conditions that valid data should satisfy, together with evidence for deciding which values require changes. Conditional functional dependencies provide one such basis by describing consistency relationships under specified attribute-value conditions [1]. Detecting a violation establishes a conflict, while locating the error and choosing its replacement require further judgments. Several changes may remove the same conflict, and an inaccurate or inapplicable constraint may itself cause an incorrect diagnosis. The central difficulty is consequently to connect evidence of a quality problem with a supported repair, taking account of both the data and the knowledge used to judge them.

Knowledge-guided data cleaning uses business rules, trusted reference facts, learned statistical regularities, and human feedback to support these judgments. Rules describe requirements, reference data supply facts, and learned patterns and feedback help assess possible changes. Existing approaches differ in how they obtain this information and introduce it into cleaning decisions. Editing rules can derive repairs from trusted master data, probabilistic models can combine several sources of evidence, and learning methods can acquire repair rules or use annotations to guide detection and correction [2]–[6]. These approaches connect knowledge acquisition with practical decisions about data quality. The usefulness of knowledge depends on its relationship with the current records, attributes, and task. As systems gain access to more information, they must determine which evidence applies, how to reconcile disagreement, and when additional information would support a better decision.

This survey follows the connection between knowledge acquisition and reliable cleaning decisions across thirty studies. We examine how systems represent and reuse knowledge, generate and select repairs, and control the effects of inaccurate or insufficient evidence. The analysis compares factual correctness, constraint consistency, and downstream task utility, and considers how evaluation protocols and acquisition costs affect reported results. We also include data preparation research to explain how knowledge supports table restructuring and coordination across multiple operations. Section II introduces the knowledge foundations, Sections III and IV compare acquisition and decision mechanisms, and Section V analyzes reliability and evaluation. Section VI extends the discussion to preparation and develops research directions, before Section VII concludes. Throughout these topics, the focus remains on what information supports a decision and how the system establishes that support.

II. The Knowledge Foundations of Data Cleaning

A. Why Detection and Repair Need Different Evidence

As illustrated in Fig. 1, a value may satisfy its domain constraints but violate relationships within a record, while internally consistent records may still contradict external facts. These cases distinguish three cleaning tasks: error detection identifies suspicious values, data repair selects appropriate corrections, and missing value imputation predicts unknown values. Each requires different evidence, ranging from constraints and contextual regularities to trusted reference facts.

The distinction between tasks explains why the same knowledge can play several roles in a cleaning system. Integrity constraints expose conflicts, reference data suggest replacements, and statistical models or user feedback help judge candidates [1]–[3], [5], [6]. The evidence needed to

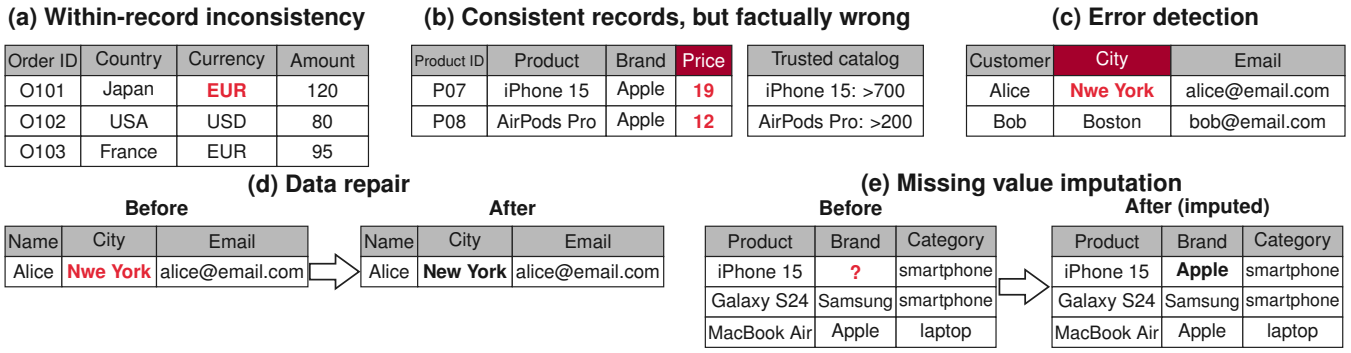


Fig. 1. Different error types and cleaning tasks require different forms of evidence. (a) EUR violates the local-currency rule for an order from Japan. (b) Prices disagree with illustrative catalog values for the same market and period. (c) A dictionary flags "Nwe York." (d) Address context or reference data support choosing "New York" among candidate repairs. (e) Product knowledge supplies the missing brand "Apple."

flag a value may leave several plausible corrections, so repair also requires a basis for selecting among them. Knowledge acquisition and use consequently interact throughout cleaning: an unresolved repair may reveal a missing fact or prompt a request for confirmation. Understanding a method requires identifying which of these decisions its knowledge supports.

B. What Knowledge Can Support Cleaning

We use cleaning knowledge to refer to information that supports data quality judgments and repair operations. Its sources include domain experts, trusted reference datasets, user feedback, and relationships learned from observations. Its representation determines how it enters a decision: explicit rules express business requirements, reference records supply candidate facts, and statistical models assess relationships among values. Source and representation describe different aspects of a method, since a system can learn rules from data or combine model predictions with logical reasoning. Rule-based and learning-based approaches can therefore work within the same system. Comparing knowledge requires considering where it comes from, how the system expresses it, and which cleaning operations use it.

Knowledge also has conditions of applicability that affect its reuse. Business rules concern particular settings, reference facts must match the target entities and attributes, and statistical regularities depend on the data from which a system learns them. The same data type can describe fields with different business meanings, while changing markets or time periods can alter which reference facts remain relevant. A system therefore needs to retain the sources and conditions of its knowledge so that later decisions can check those connections. These requirements link acquisition with semantic interpretation and provide a basis for reassessing knowledge when the data or task changes.

III. Acquiring and Reusing Cleaning Knowledge

A. From Business Rules to Trusted References

Explicit rules translate domain requirements into conditions that a cleaning system can check. Conditional functional dependencies extend functional dependencies with attribute-value conditions, describing which consistency relationships should hold for records matching a particular pattern [1]. NADEEF addresses execution across different rule types by separating the interface for user-defined rules from a shared cleaning core [7]. The two contributions concern different parts of the process: one expresses conditional requirements, while the other provides a common framework for detecting and repairing violations. Both depend on rules that correspond to the application, including the relationships among attributes that those rules constrain. Their use also requires distinguishing a violated requirement from the factual evidence needed to locate an error and select a replacement. This distinction motivates approaches that connect constraints with trusted reference records.

Certain Fixes combines editing rules with trusted master data, using attributes confirmed to be correct to match target records and derive repair operations [2]. KATARA instead identifies the semantic types of columns and the relationships among them, aligns table contents with knowledge-base facts, and uses crowdsourcing to resolve uncertain judgments [8]. Both methods obtain repair evidence from external knowledge, but they establish the connection to that knowledge differently. Certain Fixes relies on trusted matching attributes, whereas KATARA first interprets the table's semantics to find relevant facts. Their comparison separates the availability of a reference from the correspondence that makes it useful: finding additional records supports repair when the system can relate them to the correct entities and attributes. Rules and references thus complement each other by specifying requirements and supplying factual directions for changes. Constructing and maintaining these resources requires human effort, which motivates automatic discovery and reuse when data volumes and business requirements change.

B. Learning Regularities from Existing Data

Automatic discovery reduces the work of writing rules by extracting relationships from existing data, with different methods emphasizing discovery efficiency or the connection to repair. Parallel Rule Discovery uses sampling and parallel

computation to search large datasets for candidate rules [9]. Garf uses sequence generative adversarial networks to learn attribute relationships and convert them into interpretable repair rules, then progressively improves rules and data through their mutual checks [4]. Faster discovery and effective repair address different needs: the former expands the relationships a system can examine, while the latter connects learned relationships with particular corrections. The structure of the data also affects the knowledge to discover. TSDDiscover studies dependencies in time-series data, where changes over time provide evidence alongside relationships among attributes [10].

Reusing knowledge across datasets adds a further requirement because the system must establish that a learned relationship concerns the new data. Auto-Test learns semantic-domain constraints from a corpus of tables, determines whether they apply to a target column, and identifies values that violate the selected constraints [11]. Whereas Garf primarily learns repair rules from the data under cleaning, Auto-Test uses accumulated knowledge from other tables [4], [11]. This difference changes the basis for validation: a relationship supported in the source data also needs a semantic connection to the target column. Errors in the learning data can affect the relationship itself, while differences in business meaning can affect its applicability. Human feedback can supply evidence about either issue when the available data leave the judgment unresolved.

C. Using Feedback to Supply Information Missing from Data

Human feedback supplies different kinds of supervision according to the decision a method needs to make. Raha combines detection-strategy outputs with user annotations to improve error detection, whereas Baran learns corrections from repair examples and cell context [5], [6]. Certain Fixes requests confirmation of trusted attributes to support repairs through editing rules [2]. These approaches differ in the information requested and how that feedback supports subsequent decisions. A judgment about whether a cell is erroneous helps locate problems, a correct replacement helps learn corrections, and confirmation of matching attributes supports the use of reference facts. Comparing their feedback therefore requires specifying what users know and what the system asks them to provide.

The cost of interaction depends on the content and reuse of each answer as well as the number of questions. Confirming one value can support a single detection decision, while confirming trusted attributes may help derive repairs for related records. Feedback costs should therefore be evaluated in terms of the information requested and the subsequent decisions each response supports. This connects interactive cleaning with knowledge reuse: useful feedback resolves uncertainty in the current case and can also guide later judgments. Evaluating that benefit together with human effort explains more about practical value than reporting the number of interactions alone.

IV. From Knowledge to Cleaning Decisions

A. Selecting Repairs from Candidate Solutions

Candidate generation determines which corrections a repair system can consider before it selects a final value. Baran builds context from the cell itself and information in its row and column, uses several models to propose replacements, and learns to rank the candidates [6]. DataVinci combines syntactic and semantic information with edit programs that describe executable string transformations [12]. Both use contextual information, while DataVinci also represents changes as programs that can apply to other strings with similar errors. In either case, the set of available candidates constrains selection: a ranking mechanism can recover the correct value only if generation makes it available. Evaluating generation and selection separately therefore helps explain how knowledge improves a repair.

Missing value imputation adds a decision about whether to accept an answer when the available evidence is uncertain. Auto-Fill trains specialist language models for world knowledge, text reasoning, and code reasoning, calibrates their confidence scores, and selects predictions using those scores [13]. Calibration relates confidence to observed prediction correctness, giving the system a basis for deciding which outputs to accept. When every model falls below the specified threshold, Auto-Fill leaves the value missing. Candidate ranking and acceptance control therefore address different sources of error: the system may omit the correct candidate, select the wrong candidate, or accept a prediction whose confidence provides insufficient support. These decisions also affect evaluation, since increasing the acceptance threshold changes both prediction quality and the proportion of cases with an answer. A complete assessment should connect candidate availability, selection accuracy, and the scope of accepted output.

B. Combining Different Evidence in a Judgment

Evidence combination determines how several sources of knowledge influence a shared repair decision. HoloClean integrates integrity constraints, external data, and statistical regularities in a probabilistic model, using relationships among attributes and records for joint inference over candidate values [3]. Rock embeds machine learning models in logic rules so that their predictions participate in rule-based deductions [14]. HoloClean thus combines evidence through probabilistic judgments about possible values, while Rock specifies how model outputs enter logical reasoning. In both approaches, the contribution of a signal depends on how the system combines it with other information. A repair can therefore require examining which evidence favored the selected value and how competing evidence affected that choice, connecting the combination mechanism with later reliability checks.

Systems that generate cleaning knowledge during processing also need to assess the intermediate rules or functions they create. GIDCL uses graph relationships among attributes to select representative records for annotation, generates cleaning

functions with large language models, and refines those functions through classifier feedback [15]. Graph information also supports correction, while generated functions supply training examples for the classifier. Compared with Garf's learned repair rules, GIDCL connects function generation, training examples, and model feedback in an iterative process [4], [15]. These mechanisms create different objects for validation: the rules that guide repairs in one case and the generated functions and examples in the other. A function can execute successfully while applying an unsuitable change, so checking program execution and checking its cleaning effect answer separate questions. The comparison links the quality of generated knowledge with the correctness of the decisions that later use it.

C. Repairing Values, Records, and Distributions

Some quality problems require changing the structure of a record because its attributes belong to different entities. Splitting Tuples uses entity information in knowledge graphs and correlations among attributes to separate mistakenly merged records and reassign their attributes [16]. Individual values may be correct even when their combination describes the wrong entity, so this task concerns which attributes belong together. The repair object consequently differs from the fixed cell positions assumed in ordinary value replacement. Reference knowledge supports both tasks, but entity splitting uses it to recover record structure, which requires evaluating entity correspondence alongside the correctness of attribute values.

Cleaning can also target statistical relationships across a dataset. Conditional independence means that, after fixing specified conditioning attributes, one attribute group provides no additional information about another group. OTClean uses optimal transport and probabilistic mappings to adjust distributions that violate a required conditional independence relationship while controlling the cost of changes [17]. The target is a property of the distribution, so correct individual values and well-formed records do not alone establish success. Comparing cell repair, entity splitting, and distribution repair therefore requires specifying what each operation changes and which knowledge defines its objective. Cell repair selects values, splitting restores associations between attributes and entities, and distribution repair adjusts statistical relationships. Their different objects explain why a common label such as cleaning quality requires different measurements in each setting, even when the methods draw on related forms of knowledge.

V. Reliability and Evaluation of Cleaning Decisions

A. Whether Conflicts Come from Data or Knowledge

A conflict between data and knowledge can originate in either source, so reliability control must consider which one requires correction. Garf jointly improves learned rules and data, while bNDCRepair handles numerical sequential data by adjusting inaccurate constraint ranges alongside data repairs [4], [18]. By treating the knowledge behind a diagnosis as an object of correction, these mechanisms let a system respond when an apparent data error originates in a rule. The form of that reconsideration matters: revising numerical ranges changes a particular part of a constraint, while rule learning concerns the relationships represented in repair rules. Evaluation should therefore check whether the revised knowledge becomes more accurate and whether its use preserves correct data, connecting knowledge adjustment with the quality of subsequent repairs.

Accurate constraints still leave a separate question about which values caused a violation. Changing either of two conflicting records may restore consistency, even when only one contains the actual error. Chen et al. [19] use Bayesian statistics to identify erroneous attribute values among cells that violate integrity constraints. Their error detection approach highlights the difference between eliminating a conflict with few changes and locating the values that need correction. Together, these studies distinguish checking the knowledge from assigning errors within the data; a reliable repair depends on both judgments because either can direct the system toward an inappropriate change.

B. Controlling Repairs under Uncertainty

The strength of a constraint determines how it should influence repair. Hard constraints specify requirements that an accepted result must satisfy, while soft constraints express preferences among otherwise possible results. SHoTClean combines hard bounds on rates of change with soft statistical regularities for multivariate time-series cleaning [20]. The hard bounds restrict permissible changes, and statistical evidence helps select values within that space. This division gives mandatory requirements and empirical patterns distinct roles, making the conditions behind each constraint relevant to the resulting decision.

Other mechanisms address different sources of uncertainty: HoloClean reconciles competing evidence, bNDCRepair revises inaccurate constraint ranges, and Auto-Fill abstains from predictions with insufficient confidence [3], [13], [18]. Evidence combination requires checking whether inference improves selection, knowledge revision requires checking the revised constraints, and abstention requires examining prediction quality together with the amount of accepted output. These tests explain what each reliability mechanism contributes without treating the mechanisms as interchangeable.

C. Cleaning to Restore Facts or Improve Tasks

Downstream applications change which cleaning actions deserve priority because different errors can have different effects on a model. ActiveClean targets statistical models with convex loss functions, selects records for priority cleaning, and progressively updates the model using cleaned data [21]. SAGA searches for cleaning pipelines and parameters according to downstream machine learning performance [22]. The two approaches allocate effort at different levels: ActiveClean chooses records, while SAGA chooses operations and their organization. Both connect cleaning with the application's objective, making the value of a change depend on how it affects the final task. This also makes resource allocation part

of the cleaning problem, since the system must decide where limited effort can produce the most useful improvement.

Factual correctness, constraint consistency, and downstream task utility consequently describe different aspects of a cleaned dataset. Factual correctness concerns whether values reflect reality, consistency concerns compliance with specified requirements, and utility concerns performance on a particular application. These objectives can lead to different choices because a change may remove a violation without restoring the true value, while correcting an error may have little effect on a model. Comparing methods requires identifying the objective and the operations through which the system pursues it. When an approach optimizes downstream utility, evaluation should also report what happens to data-level correctness so that readers can interpret the relationship between improved task performance and the remaining factual errors.

D. Evaluation Should Explain What a System Improves

Evaluation should reflect the decisions a method makes and clearly specify the metrics and experimental assumptions used to assess them. Detection metrics assess which errors a system locates and which it misses, while repair metrics additionally require correct replacements. Candidate generation concerns whether the true value appears among possible answers; entity splitting and distribution repair instead require measurements of record structure and statistical relationships. Methods that withhold answers also need measurements of accepted output. Auto-Fill reports recall at a specified target precision as the proportion of all test cases with a correct prediction that the system accepts [13]. This differs from the proportion receiving any answer, since accepted wrong answers contribute to the latter. Changing the confidence threshold affects both correctness and output scope, so assessing reliable imputation requires reporting these quantities together. The same distinction helps explain whether an apparent improvement reflects better decisions on comparable cases or a change in which cases the system handles.

Experimental assumptions also determine which capabilities a result can establish. Baran's main experiments assume accurate error locations and assess correction at those locations, whereas GIDCL includes end-to-end experiments that combine detection and repair [6], [15]. The latter measure the effect of detection on final repairs, while the former isolate correction given the required input. A comparison should therefore identify whether a reported gain comes from error localization, candidate generation, or repair selection, and should preserve the information available to each method. For interactive systems, this includes the content and quantity of feedback; for methods using external knowledge, it includes the reference information and its correspondence with the target data. Cleaning costs then connect these experimental conditions to practical use. Writing and maintaining rules, obtaining annotations, training and running models, and retrieving and matching reference facts all consume resources. Reporting these costs alongside repair quality, output scope, and downstream performance helps explain which evidence makes a method effective and what effort its use requires.

Repair evaluation should also track changes to originally correct cells, since accuracy on known erroneous cells leaves this source of damage unmeasured. Running the same repair component with known and detector-predicted error locations, under fixed candidate sources and feedback budgets, would separate replacement quality from the consequences of missed errors and false alarms. Both corrected and newly introduced errors should contribute to the assessment of cleaning quality.

VI. Knowledge Needs in Data Preparation and Research Directions

A. From Understanding Data to Understanding Tasks

Data preparation extends the knowledge requirements of cleaning from valid values to the meaning and intended use of data. Enterprise data engineering research identifies proprietary fields, internal knowledge, and complex business requirements as difficulties for general-purpose language models [23]. Such information helps distinguish fields with similar formats but different meanings and clarifies which structures and transformations a task requires. Preparation systems therefore need access to organizational context as well as the relationships visible in the tables. This connects understanding the input with understanding the result the user expects, since an operation that preserves valid values can still produce an unsuitable structure.

Pneuma combines table schemas, sampled contents, and associated context, using keyword and semantic retrieval to find sources relevant to a current need [24]. Retrieval determines which information becomes available, while semantic alignment determines how that information corresponds to the target data, as illustrated by KATARA [8]. These roles differ even when both methods use external knowledge: a relevant source must still concern the appropriate entities, attributes, and task. The comparison links source discovery with later validation and helps explain why a preparation workflow needs both access to background information and a basis for using it in a particular operation.

B. Using Knowledge Throughout Operation Selection and Execution

Preparation systems must coordinate operations because changes to one table can affect later transformations and joins. Auto-Tables synthesizes multiple transformation steps to convert irregular tables into relational form, whereas Auto-Prep uses a global search graph to consider transformations and joins together [25], [26]. Their comparison distinguishes organizing one table from selecting compatible operations across sources, where a locally plausible transformation may obstruct a useful join. CoWrangler introduces user judgment into operation selection by recommending wrangling steps, displaying code and rationales, and allowing users to accept the suggestions [27]. The feedback here concerns which action to execute, extending the role of interaction beyond the correctness labels and replacement examples used in cleaning. These

approaches make task objectives part of operation selection, since the system needs a representation of the desired result to assess how several steps work together.

Natural-language-driven preparation further connects task interpretation with implementation and execution feedback. MegaTran uses a fine-tuned lightweight model to refine coarse requests into specific transformation requirements, then generates code and improves it through execution checks and document retrieval on demand [28]. DeepPrep uses tree-based exploration to retain alternative preparation paths and backtracks in response to execution results, allowing earlier choices to change when they impede later processing [29]. MegaTran thus connects request clarification with code refinement, while DeepPrep extends reconsideration to the preparation workflow itself. PrepBench evaluates instruction clarification, code generation, and translation from code to workflows, making these different capabilities explicit within its evaluation [30]. Together, these studies connect the knowledge needed to understand a request with the evidence needed to implement and revise a plan. Successful execution checks one aspect of the result, while correspondence with the user's requirements concerns whether the selected sequence accomplishes the intended task. Reliable preparation therefore needs judgments at both levels and a way to revise the choices that produced an unsuitable result.

C. Research Directions for Knowledge-Guided Decisions

Knowledge applicability concerns whether available information belongs to the current data and task. The applicability conditions in conditional functional dependencies, semantic alignment in KATARA, and constraint selection in Auto-Test provide starting points for this question [1], [8], [11]. Future systems could retain the sources, meanings, and validity conditions of knowledge together with the decisions that use it, so that the system can check those connections again when the data or workflow changes. A useful evaluation would hold reference facts correct while varying the target field's business meaning, the relevant period, or the data distribution. This would separate retrieving an accurate source from applying it in the correct setting and reveal whether a system recognizes that a previously useful fact no longer supports a repair. The resulting measures should include incorrect changes to valid data and appropriate deferral, alongside successful corrections. The research problem is to make such checks work across rules, references, and learned regularities as their conditions change through a preparation process.

Decisions under conflicting evidence require identifying which source of uncertainty an action would address. Existing mechanisms offer choices including data repair, knowledge revision, and abstention, as discussed in Sections IV and V [3], [4], [13], [18], [20]. Future research should examine how a system chooses among these responses when it initially lacks a reliable account of the conflict. A controlled comparison could introduce errors separately into data, rules, and reference facts, then allow the system to change a value, revise a rule, retain the current state, or request confirmation. Measuring which errors remain and which valid data survive would show when each response protects correctness. The same study should account for uncertainty about the user's task, since changing data cannot resolve a requirement the system has misunderstood. This direction connects conflict diagnosis with response selection, making the justification for an action part of what the evaluation tests.

Decision-driven knowledge acquisition asks which additional information would most improve an unresolved judgment. Requests for confirmation in Certain Fixes, annotation selection in Raha, and document retrieval in MegaTran illustrate acquisition during processing [2], [5], [28]. A broader strategy could compare retrieving a reference fact, asking a user, making another model call, or stopping, according to the expected improvement in the current decision and the resources required. The choice should depend on what remains uncertain: another candidate helps when generation lacks a plausible correction, while checking a rule's scope helps when its applicability is unclear. Acquired evidence should also retain its source and conditions so that later decisions can reuse it appropriately. Acquisition policies should therefore consider how the same evidence can support later decisions across related records and preparation steps when comparing the cost of different information sources. Evaluation could fix computation and feedback budgets, compare acquisition choices, and report factual quality, accepted output, downstream utility, and maintenance effort separately. This would establish whether an acquisition strategy improves supported decisions at a given cost and when further information provides too little benefit to justify continuing.

VII. Conclusion

Knowledge-guided data cleaning connects the acquisition of information with decisions about what to change and how to assess the result. This survey compares rules, trusted reference facts, learned regularities, and human feedback through the roles they play in detection, candidate generation, inference, and repair selection. The comparison shows why knowledge applicability and reliability matter alongside its availability, and why different repair objects and downstream objectives require different evaluation criteria. Experimental assumptions about error locations, reference information, and accepted predictions further shape what reported results establish. As cleaning extends into data preparation, these questions also concern task understanding and coordination across multiple operations. Future research can develop the connection by checking applicability under changing conditions, choosing responses according to the source of a conflict, and acquiring evidence according to unresolved decisions. Evaluating these choices together with factual correctness, task utility, and total cost would clarify how knowledge supports reliable and efficient preparation.

REFERENCES

[1] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for data cleaning," in *Proceedings of the 23rd IEEE International Conference on Data Engineering*, ser. ICDE '07. Istanbul, Turkey: IEEE, 2007, pp. 746–755.

[2] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, "Towards certain fixes with editing rules and master data," *The VLDB Journal*, vol. 21, no. 2, pp. 213–238, 2012.

[3] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "HoloClean: Holistic data repairs with probabilistic inference," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1190–1201, 2017.

[4] J. Peng *et al.*, "Self-supervised and interpretable data cleaning with sequence generative adversarial networks," *Proceedings of the VLDB Endowment*, vol. 16, no. 3, pp. 433–446, 2022.

[5] M. Mahdavi *et al.*, "Raha: A configuration-free error detection system," in *Proceedings of the 2019 International Conference on Management of Data*, ser. SIGMOD '19. Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 865–882.

[6] M. Mahdavi and Z. Abedjan, "Baran: Effective error correction via a unified context representation and transfer learning," *Proceedings of the VLDB Endowment*, vol. 13, no. 11, pp. 1948–1961, 2020.

[7] M. Dallachiesa *et al.*, "NADEEF: A commodity data cleaning system," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 541–552.

[8] X. Chu *et al.*, "KATARA: A data cleaning system powered by knowledge bases and crowdsourcing," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. Melbourne, VIC, Australia: Association for Computing Machinery, 2015, pp. 1247–1261.

[9] W. Fan, Z. Han, Y. Wang, and M. Xie, "Parallel rule discovery from large datasets by sampling," in *Proceedings of the 2022 International Conference on Management of Data*, ser. SIGMOD '22. Philadelphia, PA, USA: Association for Computing Machinery, 2022, pp. 384–398.

[10] X. Ding, Y. Li, H. Wang, C. Wang, Y. Liu, and J. Wang, "TSDDISCOVER: Discovering data dependency for time series data," in *Proceedings of the 40th IEEE International Conference on Data Engineering*, ser. ICDE '24. Utrecht, Netherlands: IEEE, 2024, pp. 3668–3681.

[11] Q. Chen *et al.*, "Auto-Test: Learning semantic-domain constraints for unsupervised error detection in tables," *Proceedings of the ACM on Management of Data*, vol. 3, no. 3, pp. 1–27, 2025, art. no. 133.

[12] M. Singh *et al.*, "DataVinci: Learning syntactic and semantic string repairs," *Proceedings of the ACM on Management of Data*, vol. 3, no. 1, pp. 1–26, 2025, art. no. 27.

[13] Y. Liu *et al.*, "Auto-Fill: Learning to predict missing values accurately with specialist language models," *Proceedings of the VLDB Endowment*, vol. 19, no. 11, pp. 3160–3173, 2026.

[14] X. Bao *et al.*, "Rock: Cleaning data by embedding ML in logic rules," in *Companion of the 2024 International Conference on Management of Data*, ser. SIGMOD '24. Santiago, Chile: Association for Computing Machinery, 2024, pp. 106–119.

[15] M. Yan, Y. Wang, Y. Wang, X. Miao, and J. Li, "GIDCL: A graph-enhanced interpretable data cleaning framework with large language models," *Proceedings of the ACM on Management of Data*, vol. 2, no. 6, pp. 1–29, 2024, art. no. 236.

[16] W. Fan *et al.*, "Splitting tuples of mismatched entities," *Proceedings of the ACM on Management of Data*, vol. 1, no. 4, pp. 1–29, 2023, art. no. 269.

[17] A. Pirhadi, M. H. Moslemi, A. Cloninger, M. Milani, and B. Salimi, "OTClean: Data cleaning for conditional independence violations using optimal transport," *Proceedings of the ACM on Management of Data*, vol. 2, no. 3, pp. 1–26, 2024, art. no. 160.

[18] X. Ding, M. Zhou, Y. Liu, C. Wang, H. Wang, and J. Wang, "bNDCRepair: Cleaning both data errors and inaccurate constraints on numerical sequential data," *Proceedings of the VLDB Endowment*, vol. 18, no. 13, pp. 5676–5688, 2025.

[19] J. Chen, Y. Sun, S. Song, H. Zhang, and X. Yuan, "Minimum change $\neq$ best cleaning: Parallel and incremental error detection under integrity constraints," *Proceedings of the ACM on Management of Data*, vol. 3, no. 4, pp. 1–26, 2025, art. no. 256.

[20] Z. Fang, W. Shao, Z. Lu, L. Chen, and Y. Gao, "SHoTClean: Bridging soft and hard constraints for multivariate time series cleaning," *Proceedings of the ACM on Management of Data*, vol. 4, no. 1, pp. 1–27, 2026, art. no. 84.

[21] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg, "ActiveClean: Interactive data cleaning for statistical modeling," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 948–959, 2016.

[22] S. Siddiqi, R. Kern, and M. Boehm, "SAGA: A scalable framework for optimizing data cleaning pipelines for machine learning applications," *Proceedings of the ACM on Management of Data*, vol. 1, no. 3, pp. 1–26, 2023, art. no. 218.

[23] J.-M. Bodensohn, U. Brackmann, L. Vogel, A. Sanghi, and C. Binnig, "Unveiling challenges for LLMs in enterprise data engineering," *Proceedings of the VLDB Endowment*, vol. 19, no. 2, pp. 196–209, 2025.

[24] M. I. L. Balaka, D. Alexander, Q. Wang, Y. Gong, A. Krisnadhi, and R. Castro Fernandez, "Pneuma: Leveraging LLMs for tabular data representation and retrieval in an end-to-end system," *Proceedings of the ACM on Management of Data*, vol. 3, no. 3, pp. 1–28, 2025, art. no. 200.

[25] P. Li, Y. He, C. Yan, Y. Wang, and S. Chaudhuri, "Auto-Tables: Synthesizing multi-step transformations to relationalize tables without using examples," *Proceedings of the VLDB Endowment*, vol. 16, no. 11, pp. 3391–3403, 2023.

[26] E. Y. Lai, Y. He, and S. Chaudhuri, "Auto-Prep: Holistic prediction of data preparation steps for self-service business intelligence," *Proceedings of the VLDB Endowment*, vol. 18, no. 7, pp. 2212–2225, 2025.

[27] B. Chopra *et al.*, "CoWrangler: Recommender system for data-wrangling scripts," in *Companion of the 2023 International Conference on Management of Data*, ser. SIGMOD '23. Seattle, WA, USA: Association for Computing Machinery, 2023, pp. 147–150.

[28] C. Li, C. Yang, Y. Luo, J. Fan, and N. Tang, "Weak-to-strong prompts with lightweight-to-powerful LLMs for high-accuracy, low-cost, and explainable data transformation," *Proceedings of the VLDB Endowment*, vol. 18, no. 8, pp. 2371–2384, 2025.

[29] M. Fan *et al.*, "DeepPrep: An LLM-powered agentic system for autonomous data preparation," *Proceedings of the VLDB Endowment*, vol. 19, no. 11, pp. 3371–3384, 2026.

[30] J. Xu, R. Wang, J. Wang, and G. Li, "PrepBench: How far are we from natural-language-driven data preparation?" *Proceedings of the VLDB Endowment*, vol. 19, no. 10, pp. 2866–2879, 2026.

GENERATIVE AI USE DISCLOSURE

OpenAI Codex assisted with polishing and translating the manuscript from original Chinese version of the survey, condensing repetitive content to meet the six-page body limit, creating Fig. 1 in LaTeX, and checking the LaTeX formatting.