Vol. 3, No. 1, pp.: 96-107, 2025

DataQualityOps: Continuous Data Quality Observability for AI-Critical Databases

Elena Petrova[1], Marcus Adebayo[2, *], Yuki Tanaka[1]

[1] School of Infocomm, Republic Polytechnic, Woodlands Avenue 9, Singapore 738964, Singapore

[2] School of Computing, National University of Singapore, Singapore 119077

** marcus.adebayo@nus.edu.sg*

Article Information	
Received	14 October 2024
Accepted	22 February 2025
DOI	https://doi.org/10.63646/datamind.2025.030107

Abstract

Machine learning systems depend on data whose quality can degrade silently between training and inference. Missing values, distributional shifts, constraint violations, and stale records can erode model accuracy without triggering conventional database alarms. This paper introduces DataQualityOps, a framework for window-based continuous data quality observability tailored to databases that feed AI and machine learning pipelines. The system profiles incoming data against learned statistical baselines, validates records against declarative quality constraints, and aggregates dimension-level scores into a composite data quality index that serves as a CI/CD gate for model retraining and deployment decisions. We define five quality dimensions relevant to AI-critical databases—completeness, consistency, timeliness, uniqueness, and distributional stability—and implement anomaly detectors for each. Evaluation on a twelve-week simulated production workload with injected quality degradation events shows that DataQualityOps detects quality anomalies with a macro-averaged F1-score of 0.92 across five degradation categories, identifies degradation onset within a median of 1.3 hours, and reduces undetected degradation events under the simulated monitoring protocol by 78 percent compared to a manual inspection baseline. The framework integrates with standard MLOps toolchains and provides actionable quality diagnostics that enable data engineers to intervene before degraded data contaminates model predictions.

Keywords: ***Data quality; data observability; MLOps; data-centric AI; statistical drift detection; CI/CD pipelines***

1. Introduction

Model performance in production depends as much on data quality as on algorithmic sophistication. The data-centric AI movement argues that systematic improvements to training and serving data often yield larger accuracy gains than architectural innovations applied to noisy inputs (Ng, 2021). Yet infrastructure for monitoring data quality remains underdeveloped compared to model performance and system observability stacks (Polyzotis et al., 2019; Sambasivan et al., 2021).

The problem is especially acute for AI-critical databases: production data stores whose contents directly feed feature engineering, model training, or real-time inference pipelines. A hospital records database feeding a diagnostic classifier, a transaction log feeding a fraud detection model, or a sensor archive feeding a predictive maintenance system all qualify as AI-critical because quality degradation in these stores can propagate silently into incorrect predictions with real-world consequences (Sculley et al., 2015). Unlike application bugs that produce visible errors, data quality issues—missing values, distributional shifts, stale records, duplicated entries—often manifest as gradual model accuracy decay rather than immediate system failure, making them difficult to detect through conventional monitoring (Breck et al., 2019).

Existing approaches to data quality assurance fall into three broad categories. First, schema-level validation tools such as Flyway and Liquibase enforce structural correctness but do not monitor statistical properties of data values. Second, expectation-based testing frameworks such as Great Expectations and Deequ allow users to declare quality assertions that are evaluated at batch boundaries, but they require manual rule authoring and lack continuous monitoring capability (Schelter et al., 2018). Third, emerging data observability platforms such as Monte Carlo, Anomalo, and Bigeye provide automated anomaly detection but are typically closed-source, cloud-native, and difficult to integrate with on-premise or hybrid AI infrastructure (Barrak et al., 2021).

This paper presents DataQualityOps, an open-architecture framework designed to bridge these gaps. The system operates as a continuous profiling and validation layer between the AI-critical database and the downstream ML pipeline, providing dimension-level quality scores that function as both observability signals and CI/CD deployment gates. The framework is designed for operational integration, low configuration overhead, and compatibility with standard MLOps toolchains including MLflow, Kubeflow, and Airflow.

DataQualityOps is implemented in Python 3.11, with profiling queries generated via SQLAlchemy supporting PostgreSQL, MySQL, and Snowflake. Quality constraints are declared in a YAML configuration file specifying column-level rules, table-level cross-checks, and distribution baseline parameters. The DQI and dimension-level scores are emitted as Prometheus metrics via a built-in exporter, enabling Grafana dashboard visualisation and PagerDuty alert routing. CI/CD integration is implemented through an Airflow operator and a Kubeflow pipeline step that evaluates the DQI against the deployment threshold before triggering model retraining. The prototype is deployed as a Kubernetes CronJob with configurable scheduling; the research artefact is available upon request.

ISSN: 3071-5601 | © 2025 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

2. Related Work

Wang and Strong (1996) proposed the foundational data quality taxonomy distinguishing intrinsic, contextual, representational, and accessibility dimensions. Batini et al. (2009) surveyed assessment methodologies, while Gudivada et al. (2017) focused on how data defects affect ML model behaviour.

Sculley et al. (2015) identified data dependencies as a major source of technical debt in ML systems. Polyzotis et al. (2019) formalised data validation for ML pipelines. Breck et al. (2019) introduced the ML Test Score including data quality prerequisites. Schelter et al. (2018) developed Deequ for declarative data quality verification on Spark.

Table 1. *Comparison of data quality assurance approaches for ML-serving databases.*

Approach	Monitoring Mode	Statistical Drift	Quality Scoring	ML Pipeline Integration
Great Expectations	Batch assertions	No (rule-based only)	No	Partial (Airflow)
Deequ (Schelter et al.)	Batch profiling	Basic distribution checks	Partial	Spark pipelines
Monte Carlo	Continuous SaaS	Yes (automated)	Yes	Cloud-native MLOps
Evidently AI	Batch + dashboard	Yes (report-based)	Partial	MLflow, Grafana
DataQualityOps (this work)	Continuous on-premise	Yes (learned baselines)	Yes (composite index)	MLflow, Kubeflow, Airflow

Table 1 positions DataQualityOps relative to existing tools. The key differentiators are hourly-window on-premise monitoring with learned statistical baselines, a composite quality index suitable for automated CI/CD gating, and integration with multiple MLOps orchestrators. Unlike SaaS-only platforms, DataQualityOps can be deployed within air-gapped or regulated environments where data cannot leave the organisation (Sidi et al., 2012).

3. System Architecture

3.1 Overview

DataQualityOps operates as a sidecar service that connects to the target database via read-only credentials and executes a four-stage pipeline: profiling, drift detection, constraint validation, and quality score aggregation. The profiling engine computes statistical summaries for each monitored column, including null rates, cardinality, value distributions, and freshness timestamps. These profiles are compared against learned baselines established during an initial two-week calibration period. Monitoring operates on hourly batch windows: each cycle scans only records ingested since the previous window using incremental timestamp-based queries, avoiding full-table scans. For tables exceeding one million rows per window, stratified sampling at ten percent is applied. Baselines are updated via a 30-day rolling window to accommodate gradual legitimate change while preserving sensitivity to abrupt degradation. Deviations exceeding configurable thresholds, controlled by Benjamini-Hochberg false discovery rate correction across all monitored columns, are flagged as quality anomalies (Ehrlinger and Woess, 2022).

ISSN: 3071-5601 | © 2025 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Figure 1. *DataQualityOps pipeline architecture: profiling, drift detection, constraint validation, and quality score aggregation feeding CI/CD gates.*

Figure 1 illustrates the pipeline. The statistical drift detector identifies distributional changes using the Kolmogorov-Smirnov test for continuous variables and the chi-squared test for categorical variables, with Benjamini-Hochberg false discovery rate control applied across all monitored columns within each window to limit cumulative false positives. For high-cardinality categorical variables (more than 500 distinct values), categories are binned into frequency-ranked groups before applying the chi-squared test. Significance thresholds adapt to sample size variation across monitoring windows (Rabanser et al., 2019). The constraint validator evaluates declarative quality rules specified in a YAML configuration file, covering nullability, uniqueness, referential integrity, format patterns, and range boundaries. Both detection paths feed into the quality score aggregator, which computes dimension-level scores and a composite data quality index.

3.2 Quality dimensions

We operationalise five quality dimensions for AI-critical databases. Completeness measures the proportion of non-null values across required columns. Consistency captures cross-column and cross-table referential and logical coherence. Timeliness tracks the freshness of records relative to expected ingestion schedules. Uniqueness measures the absence of unwanted duplicates based on declared key columns. Distributional stability quantifies the degree to which value distributions remain within the baseline envelope established during calibration (Heinrich et al., 2018).

Table 2. *Quality dimension definitions and detection mechanisms.*

Dimension	Operational Definition	Detection Method	Typical AI Impact

ISSN: 3071-5601

Completeness	Fraction of non-null required values	Null rate monitoring per column	Missing features cause imputation errors
Consistency	Logical coherence across columns/tables	Cross-field rule validation	Contradictory inputs confuse classifiers
Timeliness	Record freshness vs. expected schedule	Ingestion timestamp tracking	Stale features yield outdated predictions
Uniqueness	Absence of unwanted duplicate records	Key-based deduplication check	Duplicates bias training set distributions
Distributional stability	Value distribution within baseline envelope	KS-test (continuous), chi-sq (categorical)	Feature drift degrades model calibration

Table 2 links each dimension to its operational definition, detection mechanism, and typical downstream impact on AI model behaviour. The distributional stability dimension is particularly important for AI-critical databases because even when individual records pass all constraint checks, a gradual shift in the joint feature distribution can cause model performance degradation that is invisible to rule-based validators (Rabanser et al., 2019; Lu et al., 2019).

Table 3. *Quality dimension selection rationale: included, excluded, and planned dimensions.*

Dimension	Status	Rationale
Completeness	Included	Directly observable via null rates; critical for ML feature availability
Consistency	Included	Cross-field logic detectable from metadata and rules
Timeliness	Included	Observable from ingestion timestamps; freshness drives prediction relevance
Uniqueness	Included	Key-based deduplication computable without external truth
Distributional stability	Included	Critical for ML feature drift; detectable via statistical tests
Accuracy	Excluded	Requires external ground truth unavailable at runtime
Fairness / bias	Future work	Requires protected-attribute context and task specification
Label quality	Future work	Applicable only to supervised training sets, not general databases
Schema drift	Separate tool	Structural quality addressed by dedicated schema monitoring

Table 3 explains why these five dimensions were selected and which commonly discussed quality dimensions were excluded. The guiding criterion is runtime observability without external ground truth: all five included dimensions can be assessed from the database content and metadata alone, without requiring labelled validation sets or domain-specific accuracy benchmarks (Wang and Strong, 1996; Heinrich et al., 2018).

3.3 Composite quality index

The composite data quality index (DQI) is computed as a weighted average of dimension-level scores: DQI = w_c * S_completeness + w_s * S_consistency + w_t * S_timeliness + w_u * S_uniqueness + w_d * S_distribution, where default weights are w_c = 0.25, w_s = 0.20, w_t = 0.20, w_u = 0.15, and w_d = 0.20. Weights were calibrated through pairwise comparison (AHP method) with five ML engineering practitioners (mean experience 6.2 years; consistency ratio below 0.10). Ablation analysis shows that removing distributional stability reduces macro-F1 from 0.92 to 0.85, removing timeliness reduces it to 0.89, and using equal weights yields a macro-F1 of 0.90, confirming that the elicited weights outperform naive uniform allocation. Sensitivity analysis shows that

perturbations of plus or minus fifteen percent on any weight change the DQI by less than 0.04. The DQI maps to four operational states: healthy (DQI above 0.90), degraded (0.75 to 0.90), critical (0.60 to 0.75), and failed (below 0.60). When integrated as a CI/CD gate, a DQI below the degraded threshold blocks automated model retraining until the quality issue is resolved (Polyzotis et al., 2019).

4. Experimental Evaluation

4.1 Setup

We evaluate DataQualityOps on a simulated twelve-week production workload modelled after a healthcare claims database containing 42 tables, 380 columns, and approximately 2.4 million records per week. Five categories of quality degradation are injected at controlled intervals: null spikes (sudden increase in missing values), distribution shifts (gradual change in feature means and variances), duplicate surges (batch insertion of near-duplicate records), format violations (malformed date or code fields), and freshness delays (ingestion lag exceeding the expected schedule). Each category is injected three to five times across the twelve-week period, yielding 22 degradation events in total (Barrak et al., 2021).

Detection performance is measured at the hourly monitoring-window level: each window is classified as containing or not containing a degradation signal, and precision, recall, and macro-averaged F1-score are computed across all windows for each of the five degradation categories. The 22 injected events produce a total of 347 degradation-positive windows out of 2,016 total windows (12 weeks times 24 hours times 7 days). Detection latency is the time from degradation onset to the first alert emission. Incident reduction rate compares undetected degradation events under DataQualityOps versus a manual weekly inspection baseline.

4.2 Quality score trends

Over the twelve-week evaluation window, the composite DQI tracked quality degradation events with clear signal separation from baseline noise. During weeks five through eight, a simulated database migration introduced transient null spikes and consistency violations that reduced the DQI from 0.96 to 0.84, correctly triggering degraded-state alerts. Post-remediation in week nine, scores recovered to healthy levels within two monitoring cycles.

ISSN: 3071-5601 | © 2025 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Figure 2. *Weekly quality score trends across three dimensions during a twelve-week evaluation (scores normalised to [0, 1]). The shaded region marks weeks 5–8, during which null spikes and consistency violations were injected to simulate a database migration event.*

Figure 2 shows the temporal evolution of completeness, consistency, and timeliness scores. The migration event between weeks five and eight produces visible degradation in completeness and consistency scores, while timeliness remains relatively stable because the migration did not affect ingestion scheduling. This dimensional disaggregation enables targeted diagnostics: an ML engineer observing the dashboard can immediately identify which quality dimension is degrading and prioritise remediation accordingly (Heinrich et al., 2018).

4.3 Detection performance

DataQualityOps detects quality degradation events with a macro-averaged F1-score of 0.92 across the five categories. Format violations achieve the highest F1-score (0.96) because they produce unambiguous pattern-matching failures. Distribution shifts show the lowest F1-score (0.86) because gradual distributional changes are inherently harder to distinguish from natural data variability, particularly during early onset when the shift magnitude is small (Lu et al., 2019).

Table 3. *Detection performance by quality degradation category.*

Degradation Category	Injected Events	Precision	Recall	F1-Score
Null spike	5	0.96	0.93	0.945
Distribution shift	4	0.88	0.84	0.860
Duplicate surge	5	0.93	0.90	0.915
Format violation	4	0.97	0.95	0.960
Freshness delay	4	0.91	0.88	0.895

ISSN: 3071-5601 | © 2025 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Overall (macro-avg.)	22	0.93	0.90	0.920

Table 4 presents per-category detection results. The performance gradient from format violations (highest) to distribution shifts (lowest) reflects the inherent difficulty spectrum: discrete violations produce sharp signals, while statistical drifts require accumulated evidence across multiple monitoring windows before confidence thresholds are crossed.

Figure 3. *Precision, recall, and F1-score by quality degradation category (computed at hourly monitoring-window level across a single simulation run; error bars omitted as results represent one realisation).*

Figure 3 visualises the performance breakdown. The median detection latency across all events is 1.3 hours from degradation onset to alert emission, with a range from 12 minutes for abrupt null spikes to 4.7 hours for slow-onset distribution shifts. Compared to the manual weekly inspection baseline, DataQualityOps reduces undetected degradation events by 78 percent, from 18 undetected events under manual inspection to 4 under continuous monitoring (Schelter et al., 2018).

Table 4. *Incident reduction: DataQualityOps versus multiple baselines.*

Metric	Manual Baseline	DataQualityOps	Improvement
Detected incidents	4 of 22	18 of 22	+14 events
Undetected incidents reaching ML	18 of 22	4 of 22	–78%
Median detection latency	7 days (batch)	1.3 hours	–99.2%
False positive rate	N/A	0.08 per dimension per week	Acceptable

ISSN: 3071-5601 | © 2025 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Table 5 summarises the comparison across three baselines. The manual baseline detects only events visible during weekly reviews. The Great Expectations rule suite detects constraint violations effectively but misses distributional shifts that do not violate declared rules. DataQualityOps achieves a false positive rate of 0.08 per dimension per monitoring week (approximately two false alerts per week across all dimensions), while detecting both rule-violating and statistical anomalies that rule-only frameworks miss (Schelter et al., 2018).

For comparison, a Great Expectations rule suite configured with 45 hand-authored expectations detects 11 of 22 events (50 percent), primarily constraint violations and null spikes, but misses all four distribution shift events and two of four freshness delays. A simple statistical threshold baseline (fixed z-score alerts per column) detects 13 of 22 events but generates 0.31 false positives per dimension per week, nearly four times the DataQualityOps rate. These comparisons confirm that learned-baseline monitoring with statistical drift detection provides meaningful incremental value over rule-only and threshold-only approaches.

5. Discussion

The results demonstrate that continuous data quality observability is both technically feasible and operationally valuable for AI-critical databases. Several implications merit discussion. First, CI/CD gating creates a feedback mechanism preventing degraded data from contaminating model retraining. When the DQI falls below the degraded threshold (0.90), the Airflow operator blocks the retraining DAG until the quality issue is resolved, producing an auditable gate decision in the pipeline log (Sculley et al., 2015; Paleyes et al., 2022).

Second, the dimensional disaggregation of quality scores enables targeted remediation. Rather than receiving a generic data quality alert, engineers see which specific dimension is degrading, in which tables, and with what severity. This diagnostic specificity reduces mean time to resolution because the remediation strategy differs fundamentally across dimensions: null spikes typically require upstream pipeline fixes, distribution shifts may signal legitimate concept drift requiring model adaptation rather than data repair, and freshness delays point to ingestion infrastructure bottlenecks (Sambasivan et al., 2021; Whang et al., 2023).

Third, the distinction between data quality degradation and legitimate concept drift warrants careful treatment. Not every distributional shift represents a quality defect; some shifts reflect genuine changes in the underlying phenomenon that the model should adapt to rather than reject. DataQualityOps currently classifies distributional anomalies into three provisional categories: data defect (co-occurring with constraint violations), statistical drift (distribution change without constraint violation), and expected variation (within seasonal baseline tolerance). The system leaves final drift-versus-defect adjudication to the engineer but provides diagnostic context to support the decision. Future versions should integrate model performance feedback and seasonal baseline calendars to automate this distinction further (Lu et al., 2019; Gama et al., 2014).

ISSN: 3071-5601 | © 2025 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index

Regarding computational overhead, each hourly monitoring cycle completes in approximately 45 seconds for the 42-table test schema under incremental scanning, consuming less than two percent of database CPU capacity. Preliminary scaling tests with 200 tables show cycle times of approximately three minutes; systematic evaluation at 500-plus table scale remains future work. The study has additional limitations. The evaluation uses a simulated workload with controlled degradation injections rather than uncontrolled production data. The 42-table healthcare schema, while realistic in structure, does not capture the full complexity of multi-database, multi-pipeline environments where quality issues can propagate across service boundaries. The statistical drift detectors assume approximately stationary baselines; environments with strong seasonality or trend require baseline models that account for expected temporal variation. Future work should evaluate DataQualityOps in production deployments across multiple domains and extend the framework to support streaming data sources alongside batch-oriented databases.

6. Conclusion

This paper has presented DataQualityOps, a framework for window-based continuous data quality observability in AI-critical databases. The system monitors five quality dimensions through statistical profiling and constraint validation, aggregates dimension-level scores into a composite quality index, and integrates with CI/CD pipelines to prevent degraded data from reaching downstream ML models. Evaluation on a twelve-week simulated workload demonstrates a macro-averaged F1-score of 0.92 for degradation detection, median detection latency of 1.3 hours, and a 78 percent reduction in undetected degradation events compared to manual inspection. The broader conclusion is that data quality observability should be treated as a first-class component of MLOps infrastructure, on par with model monitoring and system observability, because the reliability of AI systems ultimately depends on the quality of the data they consume.

Declaration of AI-assisted language editing

During the preparation of this manuscript, language-model assistance was used for English polishing and document organisation. The authors take full responsibility for the content, system design, experiments, and interpretations.

References

Barrak, A., Eghan, E. E., & Adams, B. (2021). On the co-evolution of ML pipelines and source code. In IEEE International Conference on Software Analysis, Evolution and Reengineering (pp. 492–502). https://doi.org/10.1109/SANER50967.2021.00060

Batini, C., Cappiello, C., Francalanci, C., & Maurino, A. (2009). Methodologies for data quality assessment and improvement. ACM Computing Surveys, 41(3), 1–52. https://doi.org/10.1145/1541880.1541883

Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2019). The ML test score: A rubric for ML production readiness and technical debt reduction. In IEEE International Conference on Big Data (pp. 1123–1132). https://doi.org/10.1109/BigData47090.2019.9005986

Ehrlinger, L., & Woess, W. (2022). Automated data quality monitoring. Journal of Data and Information Quality, 14(1), 1–29. https://doi.org/10.1145/3488055

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), 1–37. https://doi.org/10.1145/2523813

Gudivada, V. N., Apon, A., & Ding, J. (2017). Data quality considerations for big data and machine learning. In IEEE International Conference on Big Data (pp. 2166–2173). https://doi.org/10.1109/BigData.2017.8258168

Heinrich, B., Hristova, D., Klier, M., Oberweis, A., & Schiller, M. (2018). Requirements for data quality metrics. Journal of Data and Information Quality, 9(2), 1–32. https://doi.org/10.1145/3148238

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

Paleyes, A., Urma, R.-G., & Lawrence, N. D. (2022). Challenges in deploying machine learning: A survey of case studies. ACM Computing Surveys, 55(6), 1–29. https://doi.org/10.1145/3533378

Polyzotis, N., Zinkevich, M., Roy, S., Breck, E., & Whang, S. (2019). Data validation for machine learning. In MLSys. https://doi.org/10.48550/arXiv.1906.13088

Rabanser, S., Gunnemann, S., & Lipton, Z. C. (2019). Failing loudly: An empirical study of methods for detecting dataset shift. In Advances in Neural Information Processing Systems 32 (pp. 1396–1408). https://doi.org/10.48550/arXiv.1810.11953

Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., & Aroyo, L. M. (2021). Everyone wants to do the model work, not the data work. In ACM CHI Conference on Human Factors in Computing Systems (pp. 1–15). https://doi.org/10.1145/3411764.3445518

Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. Proceedings of the VLDB Endowment, 11(12), 1781–1794. https://doi.org/10.14778/3229863.3229867

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. In Advances in Neural Information Processing Systems 28 (pp. 2503–2511). https://doi.org/10.5555/2969442.2969519

Sidi, F., Panahy, P. H. S., Affendey, L. S., Jabar, M. A., Ibrahim, H., & Mustapha, A. (2012). Data quality: A survey of data quality dimensions. In IEEE International Conference on Information Retrieval and Knowledge Management (pp. 300–304). https://doi.org/10.1109/InfRKM.2012.6205040

Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. Journal of Management Information Systems, 12(4), 5–33. https://doi.org/10.1080/07421222.1996.11518099

Ng, A. (2021). MLOps: From model-centric to data-centric AI. https://doi.org/10.48550/arXiv.2104.01858

Naumann, F. (2014). Data profiling revisited. ACM SIGMOD Record, 42(4), 40–49. https://doi.org/10.1145/2590989.2590995

Rekatsinas, T., Chu, X., Ilyas, I. F., & Re, C. (2017). HoloClean: Holistic data repairs with probabilistic inference. Proceedings of the VLDB Endowment, 10(11), 1190–1201. https://doi.org/10.14778/3137628.3137631

Abadi, D., Ailamaki, A., Andersen, D. G., Bailis, P., Balazinska, M., Bernstein, P., & Zaharia, M. (2020). The Seattle report on database research. ACM SIGMOD Record, 48(4), 44–53. https://doi.org/10.1145/3385658.3385668

Fernandez, R. C., Subramaniam, P., & Franklin, M. J. (2020). Data market platforms: Trading data assets to solve data problems. Proceedings of the VLDB Endowment, 13(12), 1933–1947. https://doi.org/10.14778/3415478.3415479

Halevy, A., Korn, F., Noy, N. F., Olston, C., Polyzotis, N., Roy, S., & Whang, S. E. (2016). Managing Google's data lake: An overview of the Goods system. IEEE Data Engineering Bulletin, 39(3), 5–14. https://doi.org/10.1109/ICDE.2016.7498275

Whang, S. E., Roh, Y., Song, H., & Lee, J.-G. (2023). Data collection and quality challenges in deep learning: A data-centric AI perspective. The VLDB Journal, 32(4), 791–813. https://doi.org/10.1007/s00778-022-00775-9

Caveness, E., Smith, G., Schelter, S., Ramasamy, H., Baylor, D., & Polyzotis, N. (2020). TensorFlow data validation: Data analysis and validation in continuous ML pipelines. In ACM SIGMOD (pp. 2793–2796). https://doi.org/10.1145/3318464.3384707

ISSN: 3071-5601 | © 2025 The Author(s). Published by DATAMIND under CC BY 4.0
https://inatgi.in/index.php/dm/index