QEC lake: Data Lake for AI-enhanced Quantum Error Correction

Aadi Patwardhan
TU Delft
A.Patwardhan@student.tudelft.nl

Max de Groot
TU Delft
M.R.deGroot-1@student.tudelft.nl

Andrei Ilinescu
TU Delft
A.Ilinescu@student.tudelft.nl

Aditya Shankar
TU Delft
A.Shankar@tudelft.nl

Yuandou Wang
TU Delft
Y.Wang-45@tudelft.nl

Wenbo Sun
TU Delft
W.Sun-2@tudelft.nl

Floris Geerts
U of Antwerp
floris.geerts@uantwerp.be

Rihan Hai
TU Delft
R.Hai@tudelft.nl

ABSTRACT

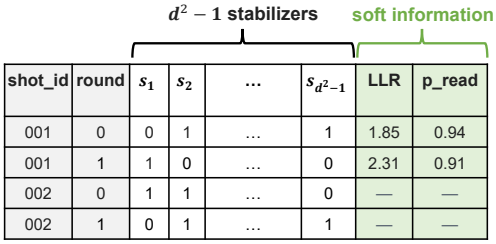
Quantum error correction (QEC) experiments generate large tabular datasets that researchers use to train AI-based decoders. However, QEC datasets are heterogeneous, scarce, and fragmented: schemas differ across code families and hardware platforms, datasets are scattered across incompatible repositories, hindering fair decoder comparison. We present *QEC lake*, a domain-specific data lake for quantum error correction. QEC lake continuously crawls and refreshes open QEC datasets, produces user-configurable simulated data, generates synthetic variants from real data, and manages data and metadata for discovery, query, and reuse. The system also offers AI-assisted dataset exploration and model-oriented data preparation pipeline that converts heterogeneous QEC data into training-ready inputs for tree-based models and tabular foundation models. In the demo, attendees ingest sources, generate new datasets, explore them through structured and natural-language interfaces, and export model-ready artifacts. This demonstrates how data management can accelerate data-driven QEC analysis, benchmarking, and model development. A video demonstrating QEC lake's workflow can be found at https://www.youtube.com/watch?v=kK4b-_tWhUA.

PVLDB Reference Format:
Aadi Patwardhan, Max de Groot, Andrei Ilinescu, Aditya Shankar, Yuandou Wang, Wenbo Sun, Floris Geerts, and Rihan Hai. QEC lake: Data Lake for AI-enhanced Quantum Error Correction. PVLDB, X(X): XXX-XXX, 2026.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

Quantum computers are error-prone. A *quantum bit* (qubit) might spontaneously flip, decohere, or accumulate noise during computation. To build reliable quantum systems, researchers use *quantum error correction* (QEC): encoding one protected logical qubit across many physical qubits, then continuously measuring parity checks, called *stabilizers*, to detect errors without disturbing the protected state [15]. Achieving practical quantum error correction is the critical bottleneck in turning today's noisy quantum processors into scalable, fault-tolerant quantum computers.



shot_id	round	s_1	s_2	...	s_{d^2-1}	LLR	p_read
001	0	0	1	...	1	1.85	0.94
001	1	1	0	...	0	2.31	0.91
002	0	1	1	...	0	—	—
002	1	0	1	...	1	—	—

Figure 1: Example of tabular QEC experiment data. The code distance d determines the number of columns.

Figure 1 shows a typical QEC dataset: each row is one experimental trial (*shot*), while columns record *stabilizer outcomes* (binary parity check results) across *measurement rounds*. The pattern of non-zero outcomes is called the *syndrome*, it is the input feature vector for decoding. A classical *decoder* processes this syndrome data to predict which qubits flipped. Recent work shows that learned decoders (LSTMs, GNNs, transformers) can match or outperform classical algorithms on real quantum hardware [4, 8, 16]. QEC decoding has now become a machine learning problem over structured data, but lacks the data infrastructure to support it. More concretely, QEC data presents three central challenges: *heterogeneity, scarcity, and fragmentation* as we explain below.

C1 Schema heterogeneity. QEC datasets do not conform to a single relational schema. The first source of variation is scale: the number of measurement columns grows with the target error tolerance. In surface-code experiments, the number of *stabilizer columns* is determined by the code distance d, yielding $d^2 - 1$ binary syndrome columns. In practice, this ranges from 8 columns in small instances to well over 120 in larger ones. The second source of variation is optional *soft-information attributes*. All datasets contain binary stabilizer outcomes, but some hardware platforms additionally expose analog readout metadata such as log-likelihood ratios (LLRs), which are scores indicating how strongly the analog measurement signal supports the binary outcome. Another soft information is the readout-confidence values p_{read}, which is the probability that the binary outcome is correct. These soft-information attributes are not universally available, so datasets generated for the same code distance may still differ in width and type composition. In summary, the schemas of QEC datasets are inherently heterogeneous and poorly suited to fixed-table assumptions.

Proceedings of the VLDB Endowment, Vol. X, No. X ISSN 2150-8097.
doi:XX.XX/XXX.XX

C2 Data scarcity. Large-scale real multi-round QEC datasets are currently available only from organizations (e.g., Google [2]) or research labs [3] with advanced quantum hardware, limiting decoder training, validation, and fair benchmarking.

C3 Format fragmentation. While QEC data is fundamentally tabular (shots × rounds × stabilizers), available datasets are published in incompatible file formats that prevent direct use. Some appear as compressed archives bundling circuits, syndrome samples, and decoder outputs[1], others as multidimensional scientific arrays (HDF5, NetCDF) with domain-specific indexing[2], and others as custom binary encodings. Worse, after extracting raw syndrome tables, different AI models need different derived representations: LSTMs expect padded time sequences (rounds as timesteps), GNNs expect graphs with syndrome defects as nodes, XGBoost expects flattened feature tables with engineered statistics. Researchers currently write custom extraction and transformation scripts for each dataset-model combination, making reproducible comparisons nearly impossible.

These challenges, various schemas, scarce and scattered data, fragmented formats, align naturally with data lake design principles. Data lakes support *schema-on-read*: raw data is ingested flexibly, with structure imposed only when queried or processed [6]. This matches QEC's needs: store heterogeneous syndrome datasets in their native formats, then apply code-family-specific schemas and model-specific transformations on demand.

Contributions. We present *QEC lake*, a domain-specific data lake for quantum error correction. To tackle data scarcity, QEC lake continuously crawls open QEC datasets, providing centralized access to scattered experimental data. It further enriches QEC datasets by supporting user-configurable simulation and generating synthetic data from real experiments using generative AI models. It provides flexible data and metadata management for QEC artifacts, enabling search, query, export, and reuse. It also offers an AI-assisted dataset exploration and model-oriented data preparation pipeline that transforms heterogeneous QEC datasets into training-ready inputs for downstream AI models, including tree-based models and tabular foundation models that remain underexplored in QEC, enabling fair, reproducible benchmarking.

Data lakes have been widely studied for enterprise analytics; see recent tutorials [1, 11] and our survey [6]. Rather than proposing another generic data lake, this work demonstrates that domain-specific data lake design is essential for closing critical infrastructure gaps in AI-enhanced scientific computing. Our technical contributions unify heterogeneous QEC data management, synthetic data generation, metadata management, and end-to-end preprocessing, facilitating reproducible benchmarking protocols that prior QEC research lacked. Overall, this work establishes data lakes as the core infrastructure for AI-enhanced QEC.

2 DATA AND METADATA MANAGEMENT

QEC lake addresses a new challenge that conventional data lakes do not directly target, namely the limited availability of QEC data. To this end, QEC lake enriches QEC data availability by crawling real experimental datasets and supporting simulated and synthetic data generation, while preserving the semantics required for downstream AI-based decoder training and evaluation.

2.1 Data Enrichment

2.1.1 Crawling real data. QEC scientists should be able to upload their own datasets directly into QEC lake, since many valuable experiment traces are generated locally and may not yet be available through public data repositories. However, relying only on manual uploads would make data discovery fragmented and labor-intensive. QEC lake therefore complements user uploads with automatic dataset discovery. Zenodo[3] is a widely used public repository for QEC-related datasets and already hosts releases accompanying major surface-code and logical-qubit experiments. Accordingly, QEC lake periodically, e.g., every day, crawls Zenodo, identifies QEC-relevant dataset records from titles, abstracts, and metadata, and ingests them into the lake while preserving provenance. This gives QEC researchers a central place to see what data are already available and which datasets can be reused for decoder training, benchmarking, and comparison.

2.1.2 Simulated data. Simulated data are also important for enriching the lake, because real QEC experiments remain expensive and limited in scale. In contrast, simulation can cover a wider range of noise regimes, code distances, and measurement conditions. Prior work already combines simulated and experimental data for decoder development. For instance, Varbanov et al. evaluate a neural decoder on both simulated and experimental data, and further show that incorporating soft information from analog readout can reduce logical error rate by approximately 10% in simulation [16]. Google's transformer-based decoder [4] follows a similar strategy, using simulated samples for large-scale pretraining before adapting the decoder with a limited amount of experimental data. Our contribution in QEC lake is that simulated data generation does not stop at binary syndrome bits. Consistent with the schema in Figure 1, when the underlying measurement model permits it, the system also materializes soft-information attributes. This makes the generated tables better aligned with real experimental data, and modern AI-based QEC decoders.

2.1.3 Generating synthetic data. Real QEC experimental data are scarce, motivating the use of generative models to augment the data lake with synthetic samples. Common candidates include generative adversarial networks (GANs) [5], diffusion models [7], and large language models (LLMs). In our setting, the target is *tabular* QEC data rather than free-form text, as illustrated in Figure 1. Intuitively, a diffusion model learns how to reconstruct a clean table row from a corrupted one. During training, noise is gradually added to real samples, and the model learns to remove that noise step by step. At generation time, it starts from random noise and iteratively denoises it into a new sample that follows the training distribution. Compared with GANs, which require balancing a generator and a discriminator throughout training, diffusion models are generally easier to optimize. In our prior work, tabular diffusion models also achieved better quality and robustness than GAN baselines [12, 13]. Moreover, the QEC schema in Figure 1 is dominated by discrete fields such as shot identifiers, round indices, and stabilizer outcomes,

[1] https://zenodo.org/records/6804040
[2] https://doi.org/10.4121/902d7f9e-38bf-48a2-a2f7-52bbf7aeeedf.v1
[3] https://zenodo.org/

Table 1: Representative metadata in QEC lake's metadata catalog

Category	Examples
Dataset	Source location, syndrome/data format
Code/experiment	Code family and distance, number of rounds
QEC Decoder	Decoder type, checkpoint or hyperparameters
Hardware	Backend or chip ID, calibration snapshot

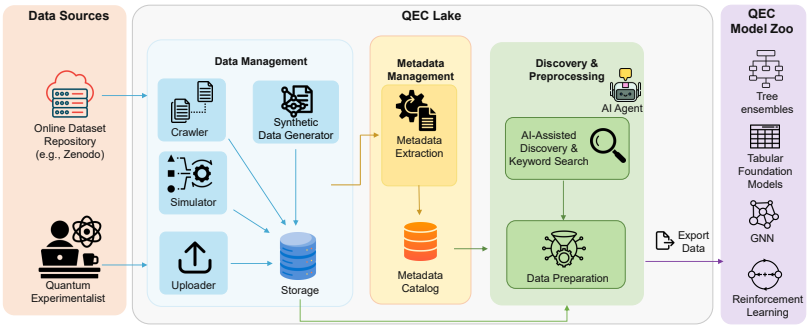


Figure 2: System architecture

while only soft-information attributes, e.g., log-likelihood ratios or readout confidences, are continuous. With generic tabular datasets containing both discrete and continuous features, we also observed that our diffusion model [13] performed better than LLM-based generators. Therefore, in QEC lake we employ our recent diffusion models for synthetic QEC data generation, conjecturing that they may also perform well in this setting.

2.2 Metadata Management

Metadata in a data lake describes the schema, provenance, and characteristics of stored datasets. Our prior work [9, 10] shows that such metadata is valuable not only for dataset discovery and querying, but also for downstream ML tasks, where it informs model selection, training, and inference. Accordingly, QEC lake adopts a structured metadata design following [9]. Table 1 summarizes the core QEC artifacts captured in the catalog: datasets, code/experiment settings, QEC decoders, and hardware context. Metadata management is essential for QEC lake because it enables the discovery and preprocessing workflow in Section 3, making experimental, simulated, and synthetic QEC datasets searchable, comparable, and reproducible across heterogeneous runs and hardware backends.

3 SYSTEM OVERVIEW

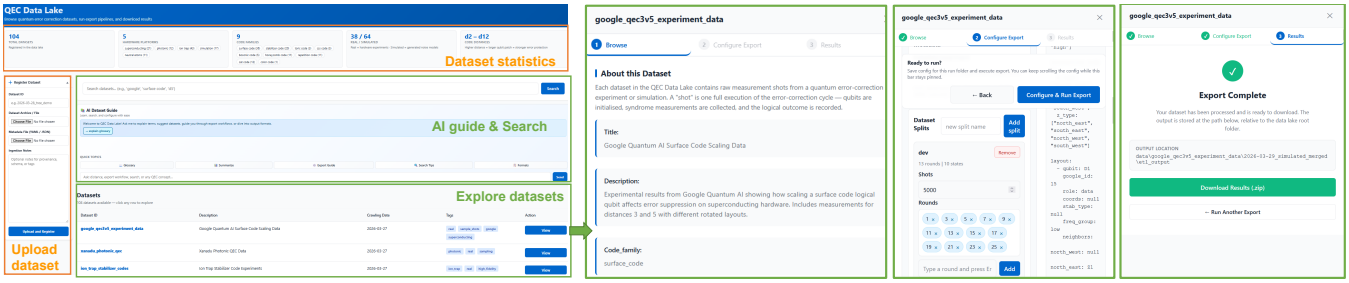
Figure 2 shows the architecture of QEC Lake. The system is organized into three internal layers: (i) data management; (ii) metadata management; and (iii) discovery & preprocessing, which connect external QEC data sources to a downstream *QEC model zoo*. This organization directly follows the challenges identified in Section 1: fragmented data sources require unified ingestion, scarce experimental data calls for simulation and synthetic generation, heterogeneous schemas require explicit metadata, and model-dependent downstream use calls for on-demand preparation rather than a fixed schema at ingest time. QEC Lake is implemented as a web-based prototype with a clean separation between storage and catalog services, and the user interface (UI). Figure 3 shows the UI.

Data management. The data-management layer is the ingestion and enrichment entry point, as described in Section 2. It includes a crawler that periodically discovers public QEC datasets (e.g., from Zenodo), an uploader for user-provided experimental data, a simulator, and a synthetic data generator. All raw and generated datasets are kept in their native formats in an object store. The current prototype uses Amazon S3. This preserves provenance and supports a schema-on-read workflow.

Metadata management. For each stored object, the metadata extraction component derives a standardized description and registers it in the metadata catalog. This layer realizes the metadata categories introduced earlier (Table 1). In the current prototype, catalog entries are stored as YAML files linked to object identifiers in Amazon S3. This representation naturally accommodates heterogeneous datasets in QEC Lake without forcing a fixed relational schema.

Discovery & preprocessing. The discovery layer operates over the metadata catalog and offers both keyword search and AI-assisted discovery. An AI agent grounds user requests in the extracted metadata, identifies candidate datasets, and passes the selected dataset and requested output representation to the data preparation component. In the current prototype, we implement the AI agent using Google's Gemini API. Once a dataset has been selected, QEC Lake performs preprocessing on demand. The data-preparation component reads the raw dataset from the object store, cross-checks its metadata, and materializes the representation required by the requested downstream task. This avoids transforming all datasets upfront and is essential for QEC, where the same syndrome data may later be consumed by different model families.

Downstream AI tasks. QEC lake powers downstream AI tasks through a unified data preparation pipeline and exports prepared datasets to downstream consumers, including our *QEC model zoo*. A model zoo is a curated set of model implementations, configurations, and metadata that allows users to run, compare, and reuse multiple models through a uniform interface. Model zoo is especially useful in AI-enhanced QEC, where the same syndrome dataset may need to be materialized as a flat table, a sequence, or a graph depending on the target model family. The model zoo is placed outside the lake in Figure 2 because QEC lake manages raw datasets, metadata, and provenance, while downstream consumers operate on prepared data. With QEC lake, users no longer need dataset-specific conversion scripts for each dataset–model pair. Instead, they select a dataset once and invoke a unified preparation pipeline that reads heterogeneous QEC datasets, validates them against metadata, and exports the model-ready structure and format required by the target AI method. This removes a key bottleneck in AI-enhanced QEC: the repeated manual effort of turning incompatible raw data into comparable inputs. As shown in Table 2, QEC lake turns fragmented QEC data into a practical foundation for applying both established and state-of-the-art models and for fair, reproducible benchmarking.



(a) View QEC lake statistics, upload a dataset, and search datasets (b) View dataset information (c) Configure data preparation (d) Export data

Figure 3: User interface of QEC lake

Table 2: Representative models in our QEC model zoo. Models newly supported through QEC lake are in purple.

Model Family	Representative models
Tree ensemble	**XGBoost, LightGBM, CatBoost**
Tabular deep learning	**TabR, TabM**
Tabular foundation model	**TabPFNv2, TabICLv2**
Feedforward neural network	Multilayer perceptron (MLP) [17]
Sequence	Long Short-Term Memory (LSTM) [16]
Graph neural networks (GNN)	Graph-convolutional message-passing GNN [8]
Transformer	Recurrent, transformer-based NN [4]
Reinforcement learning (RL)	Multi-objective policy-gradient RL [14]

4 DEMONSTRATION SCENARIOS

We structure the demonstration around three scenarios that showcase QEC lake as both a QEC data management platform and an interface for AI-driven decoder exploration.

QEC dataset searching. We first demonstrate how QEC lake supports dataset discovery. As shown in Figure 3(a), the interface presents data lake statistics, such as the number of datasets currently available and their distribution across code families, hardware platforms, and other metadata. VLDB attendees will ingest datasets, query for specific criteria (e.g., "distance-5 surface codes from superconducting platforms with soft information"), invoke data preparation pipeline to prepare data for multiple model families, compare decoder performance on a unified test set, and explore syndrome patterns through interactive visualizations. Attendees, acting in the role of a quantum experimentalist, can issue queries over dataset attributes and identify QEC datasets relevant to their setting. This scenario highlights how QEC lake turns scattered QEC data into a centralized and searchable repository, allowing researchers to quickly determine what data are available and which datasets are suitable for downstream analysis.

Support QEC model zoo. We next demonstrate how QEC lake effectively supports AI-enhanced QEC via transforming the datasets into a suitable structure and format for downstream AI tasks in the QEC model zoo. Table 2 summarizes the models currently supported by the system. The models shown in black have already been studied in prior QEC studies, whereas the models highlighted in purple, to the best of our knowledge, have not yet been systematically explored in QEC but are now enabled by QEC lake. In this scenario, attendees again play the role of a quantum experimentalist: after selecting a dataset, they invoke the data preparation pipeline in Fig. 3(b)-(d) to prepare the data, choose a decoder model from the model zoo, and inspect its performance. This scenario demonstrates how QEC lake lowers the barrier to applying both established and state-of-the-art AI models to QEC workloads, making cross-model experimentation and benchmarking in AI-enhanced QEC practical and reproducible.

Educational exploration of quantum computing. QEC lake also helps VLDB attendees understand QEC through a data-centric view. Starting from a query, they inspect a dataset together with its code family, experimental setting, hardware backend, and decoder options. This makes clear how QEC experiments are organized, how quantum errors are represented in data, and why AI-based decoding raises new data management challenges.

REFERENCES

[1] Ziawasch Abedjan et al. 2025. Data Discovery in Data Lakes: Operations, Indexes, Systems. *PVLDB* 18, 12 (2025), 5455–5459.
[2] Google Quantum AI. 2023. Suppressing quantum errors by scaling a surface code logical qubit. *Nature* 614, 7949 (2023), 676–681.
[3] Hany Ali et al. 2024. Reducing the error rate of a superconducting logical qubit using analog readout information. *Physical Review Applied* 22, 4 (2024), 044031.
[4] Johannes Bausch et al. 2024. Learning high-accuracy error decoding for quantum processors. *Nature* 635, 8040 (2024), 834–840.
[5] Ian J. Goodfellow et al. 2014. Generative Adversarial Nets. In *NeurIPS*, Vol. 27. 2672–2680.
[6] Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. 2023. Data Lakes: A Survey of Functions and Systems. *TKDE* 35, 12 (2023), 12571–12590.
[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *NeurIPS*, Vol. 33. 6840–6851.
[8] Moritz Lange et al. 2025. Data-driven decoding of quantum error correcting codes using graph neural networks. *Physical Review Research* 7, 2 (2025), 023181.
[9] Ziyu Li, Wenbo Sun, Danning Zhan, Yan Kang, Lydia Chen, Alessandro Bozzon, and Rihan Hai. 2024. Amalur: The Convergence of Data Integration and Machine Learning. *TKDE* 36, 12 (2024), 7353–7367.
[10] Ziyu Li, Hilco Van Der Wilk, Danning Zhan, Megha Khosla, Alessandro Bozzon, and Rihan Hai. 2024. Model selection with model zoo via graph learning. In *ICDE*. IEEE, 1296–1309.
[11] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. 2019. Data Lake Management: Challenges and Opportunities. *PVLDB* 12, 12 (2019), 1986–1989.
[12] Aditya Shankar, Lydia Chen, Arie van Deursen, and Rihan Hai. 2025. WaveStitch: Flexible and Fast Conditional Time Series Generation with Diffusion Models. *SIGMOD* 3, 6, Article 377 (Dec. 2025), 25 pages.
[13] Aditya Shankar, Yuandou Wang, Rihan Hai, and Lydia Y. Chen. 2026. Harpoon: Generalised Manifold Guidance for Conditional Tabular Diffusion. In *ICLR*. To appear.
[14] Volodymyr Sivak et al. 2026. Reinforcement Learning Control of Quantum Error Correction. arXiv:2511.08493 [quant-ph] https://arxiv.org/abs/2511.08493
[15] Barbara M Terhal. 2015. Quantum error correction for quantum memories. *Reviews of Modern Physics* 87, 2 (2015), 307–346.
[16] Boris M Varbanov, Marc Serra-Peralta, David Byfield, and Barbara M Terhal. 2025. Neural network decoder for near-term surface-code experiments. *Physical Review Research* 7, 1 (2025), 013029.
[17] Savvas Varsamopoulos et al. 2017. Decoding small surface codes with feedforward neural networks. *Quantum Science and Technology* 3, 1 (2017), 015004.