The Journal of Open Source Software

ETLForge: A unified framework for synthetic test-data generation and ETL validation

Kyriakos Kartas [1]

1 Independent Researcher

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Yuanqing Wang

Reviewers:

- @gwbischof
- @Nelly-Barret

Submitted: 18 June 2025
Published: unpublished

Summary

ETLForge is a Python package (Python >= 3.9) for tabular ETL testing workflows. It uses one external YAML/JSON schema to drive both synthetic data generation and validation for pandas DataFrames. The schema can describe field types, ranges, uniqueness, nullability and optional Faker templates (Faker Contributors, 2024). ETLForge provides `DataGenerator` (creates tabular test datasets) and `DataValidator` (checks datasets and returns row-level error reports). A Click-based CLI (The Pallets Projects, 2023) and a Python API expose the same core workflow, enabling local development and CI/CD automation (Fowler & Foemmel, 2013).

Statement of need

Extract-Transform-Load (ETL) processes are critical for data-driven organizations, but testing these pipelines remains challenging (Kimball & Ross, 2013; Kleppmann, 2017). Teams typically need both representative synthetic inputs and deterministic validation checks. Keeping those artifacts aligned over time can be labor-intensive and can introduce drift between test setup and quality checks (Dasu & Johnson, 2003; Loshin, 2010; Redman, 2016).

This space already has mature tools. Faker focuses on synthetic value generation (Faker Contributors, 2024). Great Expectations and Cerberus focus on data validation (Cerberus Contributors, 2024; The Great Expectations Team, 2023). pandera supports schema-based validation and can also synthesize example data from schemas (The pandera development team, 2023). ETLForge is positioned as a configuration-first tool for tabular workflows: one external schema file is consumed by both generator and validator, and the same workflow is available through CLI and Python interfaces.

State of the field

The data-quality ecosystem is mature and diverse, with different projects optimizing for different workflow styles. Tools such as Great Expectations, pandera and Cerberus offer strong validation functionality, while Faker is commonly used for synthetic value generation (Cerberus Contributors, 2024; Faker Contributors, 2024; The Great Expectations Team, 2023; The pandera development team, 2023).

ETLForge does not attempt to replace these frameworks. Its scope is narrower: tabular pandas DataFrames, declarative schema files (YAML/JSON), synthetic data generation and constraint-based validation through both CLI and Python entry points. The contribution is therefore a lightweight, configuration-first workflow that keeps generation and validation aligned under one external schema.

Software description

ETLForge implements a dual-purpose architecture where a single YAML/JSON schema drives both data generation and validation processes for **tabular data** (pandas DataFrames). The schema format supports common data types (integer, float, string, date, category), constraints (ranges, uniqueness, nullability) and realistic data generation via Faker integration.

Schema standards support: ETLForge includes adapters that detect and convert selected Frictionless Table Schema and JSON Schema definitions into ETLForge's internal tabular schema format:

- **Frictionless Table Schema**: The widely-adopted standard for describing tabular data (https://specs.frictionlessdata.io/table-schema/)
- **JSON Schema**: The popular standard for describing JSON data structures (https://json-schema.org/)

Adapter support is intentionally limited to flat/tabular field definitions. Nested `object` or `array` structures are not supported.

Core components: - `DataGenerator`: Creates synthetic tabular datasets (pandas DataFrames) - `DataValidator`: Validates tabular data (pandas DataFrames) against schema rules, returning detailed error reports - `SchemaAdapter`: Handles automatic detection and conversion of Frictionless Table Schema and JSON Schema formats - CLI interface: Enables command-line automation via Click (The Pallets Projects, 2023)

Typical workflow: ETLForge is designed to support the following pipeline:

1. **Schema definition**: Define data structure and constraints based on target system requirements
2. **Test data generation**: Generate synthetic datasets for initial ETL pipeline development and unit testing
3. **Pipeline validation**: Use the same schema to validate production or external data after ETL transformations
4. **Quality assurance**: Identify discrepancies between expected schema constraints and actual data quality

This workflow demonstrates that while ETLForge generates synthetic test data, its primary value proposition is in the validation phase where real production data is checked against expected constraints. The generation capability primarily serves to create controlled test datasets for unit testing ETL transformations before production data becomes available.

Integration approach: Rather than replacing existing tools, ETLForge complements ETL testing workflows by ensuring test data and validation rules remain synchronized. It integrates with pandas-based pipelines and exports to common formats (CSV, Excel via openpyxl). The framework targets tabular data structures.

Software methodology

Data generation algorithm: The `DataGenerator` component parses the schema specification and creates pandas DataFrames by iterating through field definitions. For each field type, it applies the appropriate generation strategy:

- **Numeric fields** (int, float): Uses Python's `random` module with specified ranges and precision constraints
- **String fields**: Generates random strings or invokes Faker methods when `faker_template` is specified
- **Date fields**: Samples dates uniformly within specified ranges using Python `datetime` utilities
- **Category fields**: Samples from predefined value sets with uniform distribution
- **Uniqueness constraints**: Maintains sets of generated values to ensure uniqueness when required
- **Nullability**: Applies configurable null rates to nullable fields using random sampling

Validation algorithm: The `DataValidator` component performs multi-pass validation on input datasets:

1. **Schema conformance**: Verifies all required columns exist and reports extra columns
2. **Type checking**: Validates each cell's data type matches schema specifications
3. **Constraint validation**: Checks range constraints, uniqueness requirements and categorical value memberships
4. **Null validation**: Ensures null values only appear in nullable fields
5. **Error aggregation**: Collects all validation failures with row and column identifiers for detailed reporting

The validation process runs all checks and aggregates all detected issues into a single report, prioritizing complete feedback over early termination.

Quality control: GitHub Actions run checks on Python 3.9-3.11, including unit tests, static analysis via flake8 and mypy, and integration tests through end-to-end workflows.

Performance characteristics

Benchmark results included in the repository (`benchmark_results.csv`) show near-linear scaling across the tested range (1,000 to 5,000,000 rows). These benchmarks were conducted using a representative schema containing 8 fields with varying complexity levels:

- 2 integer fields with range constraints (id: 1-10000000, age: 18-80)
- 1 float field with range constraints (30000.0-150000.0) and precision specifications
- 3 string fields, including two with Faker template integration (name, email) and one with length constraints
- 1 categorical field with 5 predefined values (Engineering, Marketing, Sales, HR and Finance)
- 1 date field with range constraints (2020-01-01 to 2024-12-31)

Performance scales approximately linearly with the number of rows and fields. Complex constraints such as uniqueness checking and Faker integration introduce additional overhead. The complete benchmark schema is available in the repository as `benchmark_schema.yaml` for reproducibility.

These results indicate that ETLForge can be integrated into CI/CD checks for tabular datasets while focusing on constraint-based validation rather than advanced statistical profiling.

Discussion

ETLForge unifies data generation and validation under one schema for tabular workflows, but it makes deliberate trade-offs compared with broader validation platforms. Compared with tools such as Great Expectations and pandera (The Great Expectations Team, 2023; The pandera development team, 2023), ETLForge emphasizes a smaller feature set centered on schema conformance, type checks, basic constraints and row-level error reporting. This narrower scope prioritizes straightforward configuration and reproducible test workflows over advanced profiling and statistical quality analysis.

The framework currently has several technical limitations that constrain its applicability:

- **Dataset size**: Large datasets exceeding one million rows may require memory optimization strategies, as the current implementation loads entire datasets into pandas DataFrames during validation.
- **Nested structures**: Complex nested data structures are not supported. This limitation exists because ETLForge specifically targets tabular data formats (CSV and Excel) which are inherently flat.

- **Statistical validation**: Advanced statistical validations (distribution testing, anomaly detection and correlation analysis) require integration with specialized tools. ETLForge provides constraint-based validation rather than statistical analysis.
- **Custom validation logic**: While the framework validates against schema-defined constraints, it does not currently support user-defined validation functions, limiting extensibility for domain-specific validation rules.

Availability

The ETLForge source code is available on GitHub at https://github.com/kkartas/ETLForge under the MIT license. The latest release (v1.1.0) can be installed from the Python Package Index using `pip install etl-forge`, with optional extras `etl-forge[faker]` and `etl-forge[excel]`. Complete documentation is hosted at https://etlforge.readthedocs.io/. The software supports Linux, macOS and Windows operating systems and is compatible with Python versions 3.9 through 3.11.

References

Cerberus Contributors. (2024). *Cerberus: Lightweight, extensible data validation library for python* (Version 1.3.5). https://docs.python-cerberus.org/

Dasu, T., & Johnson, T. (2003). *Exploratory data mining and data cleaning*. John Wiley & Sons.

Faker Contributors. (2024). *Faker: A python package that generates fake data for you* (Version 24.4.0). https://doi.org/10.5281/zenodo.3823398

Fowler, M., & Foemmel, M. (2013). Continuous integration. *Thought-Works*. https://martinfowler.com/articles/continuousIntegration.html

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3rd ed.). John wiley & sons.

Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. O'Reilly Media.

Loshin, D. (2010). *The practitioner's guide to data quality improvement*. Morgan Kaufmann.

Redman, T. C. (2016). *Getting in front on data: Who does what*. Harvard Business Review Press.

The Great Expectations Team. (2023). *Great Expectations: Always know what to expect from your data* (Version 0.18.0). https://doi.org/10.5281/zenodo.14976223

The Pallets Projects. (2023). *Click: Command Line Interface Creation Kit* (Version 8.1.3). https://doi.org/10.5281/zenodo.5123732

The pandera development team. (2023). *pandera: Data validation for scientists, engineers, and analysts* (Version 0.17.0). https://doi.org/10.25080/gerudo-f2bc6f59-010