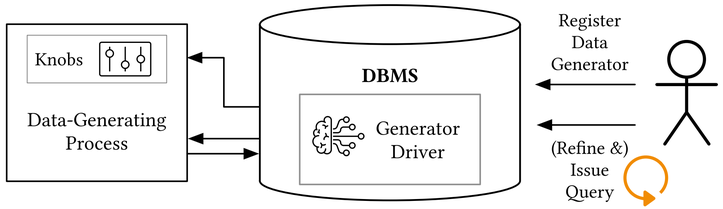
GenIE: Generator-Driven Iterative Data Exploration

Sriram Rao, Sharad Mehrotra (UC Irvine), Martin Boissier, Tilmann Rabl (HPI)

Abstract

Data generators such as physics-based simulators, benchmark generators such as TPC-H's DBGEN, and machine learning models play an important role in data analysis. Today, databases treat generators as external pre-processing steps, leading to fundamental inefficiencies and increased latency, especially when the generation of data itself depends on evolving analysis needs. We envision a new type of database technology, GenIE, that tightly integrates the generation process with analytical query processing to enable iterative and incremental analysis.



Envisioned workflow

Linear Analytical Workflows

Despite significant developments in database and hardware technologies over the past decade, many compute-intensive applications that work on generated data (e.g., from simulations or benchmark data generators) continue to be linear. We envision a system, GenIE, that overcomes these issues by tightly integrating data-generating processes into database systems. This allows progressive computation for interactive exploratory analysis.

GenIE increases the efficiency of analyzing generated data and improves the user experience. Users no longer need to tediously interact with data generators. Instead, they state their intent via SQL.

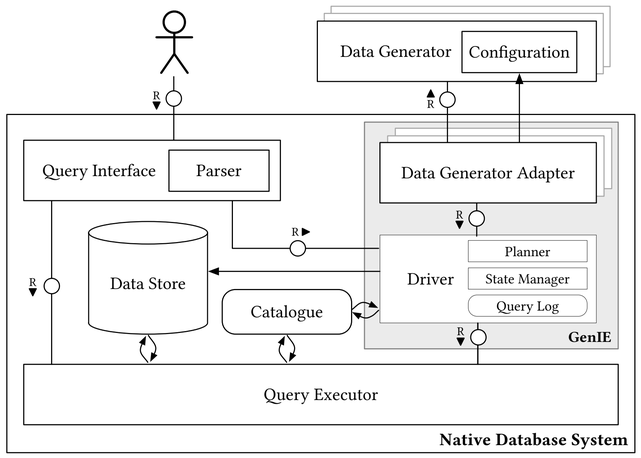
Design and Architecture of GenIE

One of our design goals is to simplify working with data-generating processes. To achieve this, GenIE minimizes manual and repetitive tasks when interacting with them. GenIE is an integrative extension to existing database systems that allows various data generators to be integrated and invoked from within the database system.

The data generator can be accessed via SQL and GenIE minimizes the amount of data that needs to be generated to answer user queries efficiently. GenIE comprises a driver module and data generator adapters. The driver initiates generator execution through corresponding adapters.

Iterative data generation is done in epochs. Thedriver first initiates the adapter with parameters determined by the current workload. During subsequent epochs, the driver reuses any data that can be reused.

The planner creates efficient generation plans based on the query log and the state of present data using knowledge of the accuracy and runtime behavior.

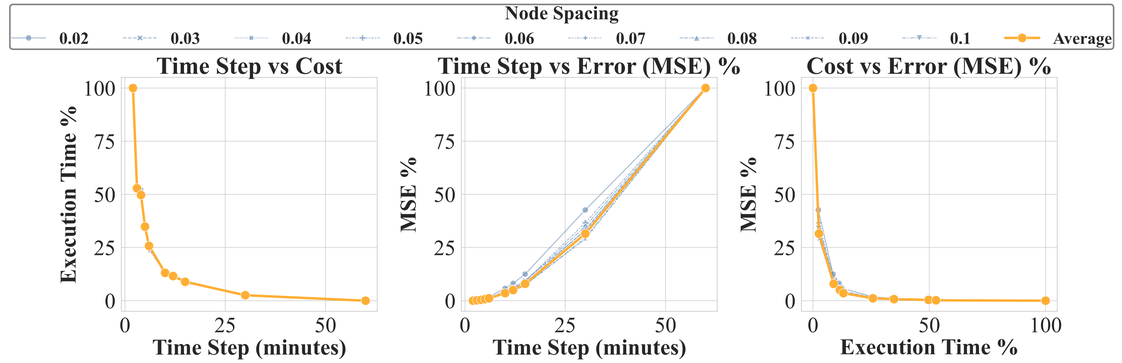


GenIE Architecture (FMC Notation)

Case Studies

Simulation-Driven Consequence Analysis

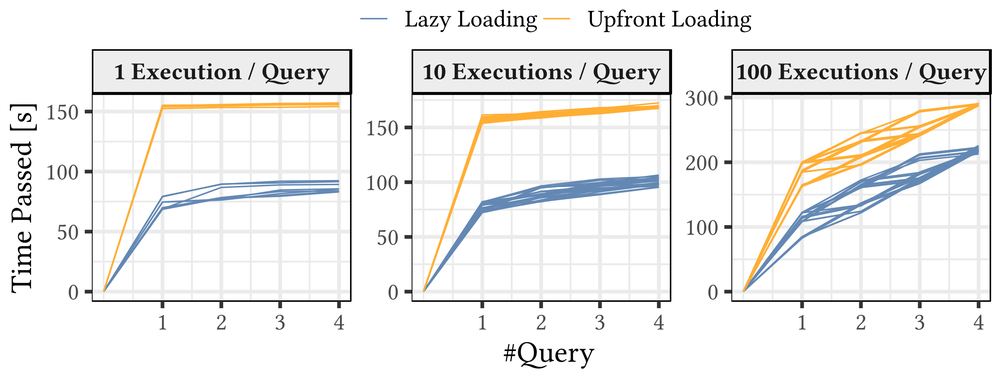
To support simulation queries on EnrichDB, the driver fetches and implements an execution plan to satisfy each query in the workload. The GenIE planner for this purpose creates a list of suggested execution parameters. The planner capitalizes on coarse results being produced sooner. We visualize the trade-off using measures of cost (execution time) and quality (Mean Square Error) of results produced by HYSPLIT. We measure the effect of the parameters through a grid search.



HYSPLIT: Error-Cost Trade-off

Database Testing and Evaluation

We integrated variations of data generators in TPC-H with Hyrise using GenIE to explore its capabilities in enabling a testing and evaluation framework to benchmark databases. Loading data partially still allows benchmarking as if all data were loaded, thus, reducing the runtime of experiments. We compare the runtimes of lazy loading (using GenIE) and Hyrise's default approach of upfront loading (TPC-H's DBGEN as a library to create and load data sets). We analyze all permutations of the evaluation query set.



Hyrise: Lazy Loading v Upfront Loading