TwinDB: Interactive What-If Analysis for Digital Twins

Xiufeng Liu
Technical University of Denmark
Kgs. Lyngby, Denmark
xiuli@dtu.dk

Ruyu Liu
Technical University of Denmark
Kgs. Lyngby, Denmark
ruyli@dtu.dk

Per Sieverts Nielsen
Technical University of Denmark
Kgs. Lyngby, Denmark
pernn@dtu.dk

Hua Lu
Aalborg University
Copenhagen, Denmark
luhua@cs.aau.dk

Abstract

Digital twins enable what-if analysis by simulating physical systems under hypothetical conditions, but simulation is computationally expensive. In this context, a *scenario* represents a configuration of input parameters (e.g., wall insulation, window properties, heating setpoints) defining a hypothetical condition to evaluate. A building energy analyst exploring retrofit options for 500 buildings faces hours of waiting time for a typical session. We demonstrate TwinDB, a scenario-aware database system that achieves significant speedup through Incremental Scenario Maintenance (ISM), which incrementally updates cached simulation results when parameters change slightly rather than re-running full simulations. Our interactive visual workspace lets conference attendees (1) explore retrofit scenarios with instant feedback, (2) compare scenarios side-by-side with confidence intervals, (3) verify ISM accuracy against full simulation, (4) write declarative TwinQL queries (a domain-specific language extending SQL for scenario-based analytics), and (5) upload custom CSV data. Built on PostgreSQL with TimescaleDB and fully open-source, TwinDB transforms digital twin analytics from batch processing to interactive exploration.

Keywords

Digital twins, what-if analysis, incremental maintenance, district heating, building energy simulation

1 Introduction

Digital twins are digital representations of physical systems that integrate real-time sensor data with simulation capabilities. They are transforming decision-making across domains from smart buildings and manufacturing to urban planning and healthcare [3, 7]. A core capability that distinguishes digital twins from traditional data analytics is *what-if analysis*: the ability to answer hypothetical questions such as “What if we retrofit this building with better insulation?” or “What if the outside temperature drops by 5°C tomorrow?” by running physics-based simulations that predict system behavior under alternative conditions.

Consider a building energy analyst tasked with evaluating retrofit options for a Danish district heating network. Using the recently published smart heat meter dataset from Aalborg [6], which contains three years of hourly consumption data from over 3,000 residential buildings, the analyst might want to explore 15 different parameter combinations for each building, varying wall U-values (thermal transmittance, which measures heat flow through a material) from 0.60 to 0.18 W/(m^2·K), testing different window specifications, and adjusting roof insulation levels. Each scenario requires running a physics-based thermal simulation that takes 1–2 seconds. Evaluating even a subset of 500 buildings with 15 scenarios each would require over 7,500 simulation runs, taking over 3 hours to complete. This is far too slow for the interactive, exploratory workflow that analysts need. This performance bottleneck is not unique to building energy; similar challenges arise whenever digital twins rely on computationally expensive simulations.

Current digital twin platforms and time-series databases treat each what-if query independently [4, 5], running a full simulation for every parameter variation. This approach misses a critical opportunity: *scenario exploration is inherently incremental.* Analysts do not jump randomly between distant parameter values; instead, they explore nearby configurations in a structured manner, for example, testing wall U-value 0.18, then 0.19, and then 0.20, gradually refining their understanding of the solution space. This incremental exploration pattern creates an opportunity for dramatic performance improvements if we can exploit the similarity between consecutive queries.

We present TwinDB, a scenario-aware database system that introduces **Incremental Scenario Maintenance (ISM)**, a technique that incrementally updates cached simulation results when parameters change slightly. The key insight enabling ISM is that physics-based simulations often exhibit *local linearity*: small parameter changes produce proportional output changes. For building energy simulation, heat loss is linear in thermal transmittance according to the fundamental heat transfer equation $Q = U \times A \times \Delta T \times t$, where Q is heat energy, U is thermal transmittance, A is surface area, ΔT is temperature difference, and t is time. By learning sensitivity coefficients (e.g., “energy changes by 14.4 MWh per W/(m^2·K) of wall U-value”), TwinDB can predict outcomes for nearby scenarios in $O(1)$ time rather than running the full $O(T)$ simulation for T timesteps.

This demonstration showcases TwinDB using real-world data from the Danish Smart Heat Meter dataset [6]. Conference attendees can interact with a visual workspace (Figure 1) that enables them to explore retrofit scenarios and observe ISM speedups in real-time, verify accuracy against full simulation, compare historical versus simulated energy consumption side-by-side, write TwinQL queries with syntax highlighting and instant results, and upload their own CSV time-series data to experience the system with personalized data.

2 System Architecture

TwinDB consists of three main components that work together to enable interactive what-if analysis: a database server built on PostgreSQL with TimescaleDB, a query engine with ISM-aware optimization, and a web-based visual interface. Figure 2 illustrates the overall architecture and data flow.

2.1 TwinDB Server and Data Model

The TwinDB server is built on PostgreSQL with the TimescaleDB extension for efficient time-series storage and querying. At the core of the system is the *Twin Data Model (TDM)*, which treats scenarios as first-class citizens alongside historical observations. The schema consists of five main tables: `asset` (physical entities with metadata like floor area and construction year), `twin` (digital representations linked to assets and models), `scenario` (what-if configurations as JSON with inheritance support), `model` (simulation models with I/O schemas), and `timeseries` (unified storage for historical observations with `cid='REALITY'` and simulated results with scenario identifiers). This unified design enables seamless SQL joins between historical and simulated data.

2.2 Query Engine with ISM Optimization

The query engine processes TwinQL queries through a four-stage pipeline: parsing, logical plan construction, optimization, and execution. The logical algebra includes `Hist` for historical data scans, `Sim` for simulation invocations, and standard relational operators. The **ISM-aware optimizer** implements rule-based rewriting with cost estimation: when a `Sim` operator targets a scenario similar to cached results, it estimates the prediction error $\epsilon = ||\Delta p|| \cdot \sigma_\beta$ where σ_β is the coefficient uncertainty, and rewrites to `ISM(cached, Δparams, β)` if ϵ is below the user-specified tolerance. The execution plan viewer highlights ISM rewrites in green and simulation fallbacks in orange, providing transparency into optimization decisions.

2.3 Web Interface and Backend

The web interface, shown in Figure 1, is built with React 18 and TypeScript. The *Model Canvas* uses React Flow to display digital twins as interactive nodes on a zoomable, pannable canvas, with edges representing physical network relationships. The *Asset Library* provides searchable access to physical assets organized by type. The *Properties Panel* displays twin metadata, model configuration, and time-series statistics, supporting data upload through drag-and-drop. The *TwinQL Console* provides a CodeMirror-based editor with syntax highlighting, a results table with sortable columns, and an execution plan viewer highlighting ISM rewrites in green and fallbacks in orange. A *Quick Compare* dialog enables visual scenario building without writing queries.

The backend uses Python 3.9+ with FastAPI, PostgreSQL 15 with TimescaleDB for storage, and implements ISO 13790 thermal simulation (5R1C model) in 2.8ms per building-year. The system deploys via Docker Compose on modest hardware (4 vCPU, 16GB RAM) and supports 10+ concurrent users. Source codes are available at https://github.com/xiufengliu/TwinDB (MIT license).

3 Key Technical Features

3.1 Incremental Scenario Maintenance

The central innovation in TwinDB is Incremental Scenario Maintenance (ISM), which enables sub-millisecond updates for scenario exploration. When a user modifies scenario parameters slightly (for example, changing wall U-value from 0.20 to 0.18), ISM computes the new result by applying a linear correction to the cached result:

$$S' = S + \sum_{p \in C} \beta_p \times \Delta p \tag{1}$$

where S is the cached simulation result, C is the set of changed parameters, Δp is the magnitude of each parameter change, and β_p is the learned sensitivity coefficient for parameter p. The sensitivity coefficient captures the rate of change in simulation output with respect to parameter changes, derived from the underlying physics (e.g., the heat transfer equation).

The system learns sensitivity coefficients online using Recursive Least Squares (RLS), which incrementally updates coefficient estimates as simulation results become available. After observing energy values of 52, 48, 44, and 40 MWh for U-values 0.60, 0.50, 0.40, and 0.30 respectively, RLS learns that $\hat{\beta} \approx 14.5$ MWh per W/m²K. Sensitivity estimates become reliable after 5–10 observations. When parameter changes exceed 20% or error bounds exceed tolerance, TwinDB automatically falls back to full simulation, indicated by an orange status bar in the UI (see Figure 1).

Scope and Limitations. ISM is most effective for simulations exhibiting locally linear behavior, which includes most physics-based models in thermodynamics, fluid dynamics, and structural mechanics. For highly non-linear systems (e.g., phase-change materials, discrete-event simulations), ISM provides limited benefit and the system falls back to full simulation. The UI clearly indicates the current mode, and "Verify Mode" allows users to compare ISM predictions against full simulation results.

3.2 TwinQL: Declarative Scenario Queries

TwinQL extends SQL with domain-specific constructs for scenario-based digital twin analytics. The following query is **fully executable** in our demonstration:

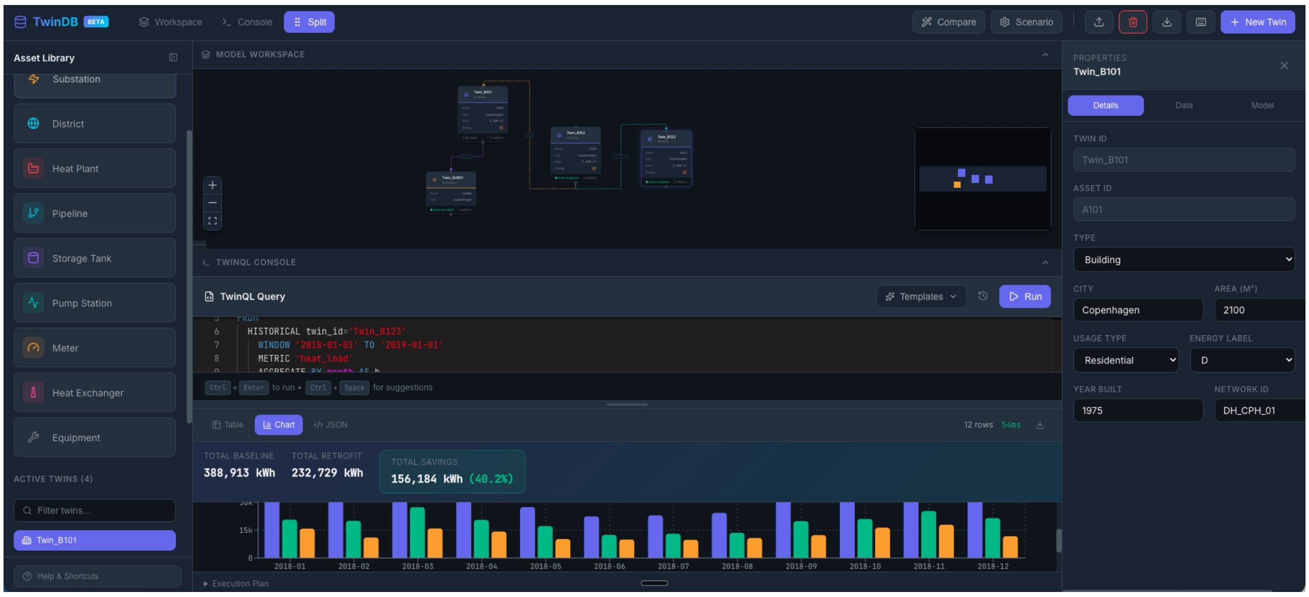
```
DEFINE SCENARIO deep_retrofit AS (
  wall_u = 0.18, window_u = 0.90, roof_u = 0.12);

TWIN SELECT month, SUM(h.heat) AS baseline,
  SUM(s.heat) AS retrofitted,
  SUM(h.heat) - SUM(s.heat) AS saving
FROM Twin_B123
HISTORICAL '2019-01-01' TO '2020-01-01' AS h
SIMULATE deep_retrofit USING BuildingHeat AS s
GROUP BY month ORDER BY month;
```

This query defines a retrofit scenario and compares actual 2019 energy consumption with simulated consumption under that scenario, computing monthly savings. The `FROM Twin_B123` clause references a virtual view over the unified `twin` table (filtered by twin identifier), giving per-twin syntax while keeping a single-table physical schema for scalability. The `DEFINE SCENARIO`, `HISTORICAL`, and `SIMULATE` clauses are fully implemented; scenario inheritance and multi-twin joins are planned.

3.3 Batch Processing for Portfolio Analysis

For network-level queries analyzing multiple buildings (such as "identify the worst-performing 20% of buildings"), TwinDB implements Batch ISM to achieve sub-linear scaling. Batch ISM groups buildings by construction type and era, computing shared sensitivity coefficients per group. Updates use vectorized operations: $S' = S + \beta_G \times \Delta p$, completing in sub-milliseconds regardless of building count. As shown in Table 1, Batch ISM processes 500-building portfolio queries in under 1ms, in contrast to 15 minutes for full simulations, achieving over 10,000× speedup.



```
FROM
  HISTORICAL twin_id='Twin_B123'
    WINDOW '2018-01-01' TO '2019-01-01'
    METRIC 'heat_load'
    AGGREGATE BY month AS h
```

TOTAL BASELINE 388,913 kWh
TOTAL RETROFIT 232,729 kWh
TOTAL SAVINGS 156,184 kWh (40.2%)

TWIN ID: Twin_B101
ASSET ID: A101
TYPE: Building
CITY: Copenhagen
AREA (M²): 2100
USAGE TYPE: Residential
ENERGY LABEL: D
YEAR BUILT: 1975
NETWORK ID: DH_CPH_01

Figure 1: TwinDB visual workspace showing the main components: (1) Asset Library on the left with categorized asset types (Substation, District, Heat Plant, Pipeline, etc.), (2) Model Workspace in the center displaying digital twins as interactive nodes with network connections, (3) TwinQL Console with query editor featuring syntax highlighting and output tabs for Table, Chart, and JSON views, (4) Results visualization showing a bar chart comparing baseline versus retrofitted energy consumption with total savings (156,184 kWh, 40.2% reduction), and (5) Properties Panel on the right showing metadata for the selected Twin_B101 including asset attributes and energy labels. The Active Twins panel at bottom-left enables quick twin selection.

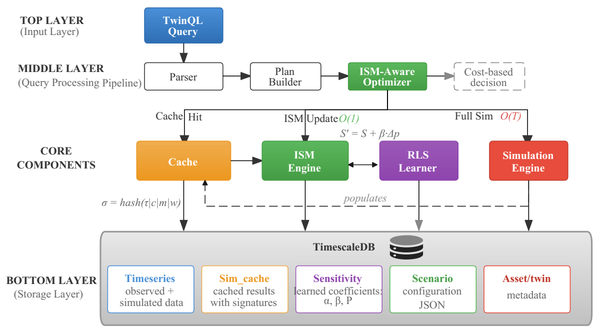


Figure 2: TwinDB architecture showing the query processing pipeline. TwinQL queries are parsed and compiled into logical plans with Hist and Sim operators. The ISM-aware optimizer applies cost-based rewrites, choosing between cache lookup ($O(1)$), ISM incremental update ($O(1)$), or full simulation ($O(T)$). Sensitivity coefficients (β) are learned online via Recursive Least Squares from simulation results.

Table 1: Performance Comparison

Query Type	ISM	Full Sim	Speedup
Single building (cold)	1.8 s	1.8 s	1×
Single building (warm)	<1 ms	1.8 s	$>10^4\times$
50 scenarios (1 bldg)	1.8 s	90 s	50×
500 buildings (warm)	<1 ms	15 min	$>10^4\times$

4 Demonstration Scenarios

Our demonstration uses the Danish Smart Heat Meter dataset [6], a publicly available Scientific Data publication containing three years (2018–2020) of hourly heat consumption data from 3,021 residential buildings in Aalborg, Denmark. The dataset includes four metrics per building: heat energy consumption, supply volume flow, return temperature, and supply temperature. Each building has associated metadata including floor area (80–200 m²), estimated construction materials (wall U-values 0.4–1.2 W/m²K), and window specifications derived from Danish building records. We pre-load a representative subset of 500 buildings so attendees can immediately begin exploring, with options to load all 3,021 buildings or upload their own custom data from any domain.

Scenario 1: Interactive Retrofit Exploration. In the first scenario, an energy analyst selects building Twin_B123 from the canvas and opens the Model Configuration dialog. As the analyst adjusts the wall U-value slider from 0.60 to 0.18 W/m²K, the results panel updates instantly (within 50ms), showing projected monthly energy consumption and cumulative savings. The "Execution Plan" panel reveals why: queries use "ISM Update: <1ms" (highlighted in green) rather than "Full Sim: 1,800ms" (shown in orange when ISM cannot be applied). The analyst observes 45% energy savings projected for the deep retrofit and can fine-tune parameters interactively to find the optimal cost-benefit tradeoff without waiting for simulations.

Scenario 2: Accuracy Verification. To build trust in ISM predictions, attendees can toggle "Verify Mode" which runs both ISM and full simulation side-by-side, displaying the prediction error (typically <0.1%) alongside the speedup factor (over 10,000× for warm queries). This scenario is particularly valuable for users who are skeptical of approximate methods, as it demonstrates that ISM maintains high accuracy while achieving dramatic performance improvements. Users can adjust the error tolerance threshold and observe how this affects the ISM versus full simulation decision boundary.

Scenario 3: Multi-Scenario Comparison Dashboard. The third scenario demonstrates the Quick Compare feature for analysis at portfolio level. The analyst creates three named scenarios: "Deep Retrofit" (wall U=0.18, window U=0.9, roof U=0.12), "Medium Retrofit" (wall U=0.25, window U=1.2), and "Baseline" (current state). After selecting 10 buildings from the canvas, clicking "Compare" generates a side-by-side visualization showing energy savings per scenario with confidence intervals derived from ISM error bounds. The visualization clearly identifies which buildings benefit most from deep versus medium retrofit, enabling data-driven investment decisions.

Scenario 4: TwinQL Query Authoring. For attendees who prefer programmatic access, the fourth scenario uses the TwinQL Console. The attendee writes a query with auto-completion assistance, executes it, and views results in a sortable table with optional chart visualization. Clicking "Explain" reveals the optimized execution plan with ISM rewrites highlighted in green, showing exactly where the optimizer chooses incremental updates over full simulation. The attendee can modify the query to explore different aggregation periods (daily, monthly, seasonal), add filtering conditions (e.g., buildings in a specific postal code), or join with asset metadata for richer analysis.

Scenario 5: Bring Your Own Data. Conference attendees can upload their own CSV files containing time-series data from any domain with reasonably linear simulation behavior. The system provides a guided wizard that assists with column mapping (timestamp, value, metric name), automatically infers data types and time granularity, creates a new twin linked to user-specified metadata, and enables immediate simulation. This scenario demonstrates that TwinDB's approach generalizes beyond the pre-loaded building energy dataset to domains such as manufacturing process control, water distribution networks, and transportation systems.

5 Related Work

Digital twin platforms such as Azure Digital Twins [5] and AWS IoT TwinMaker [1] provide comprehensive infrastructure for managing twin lifecycles, ingesting sensor data streams, and connecting to visualization dashboards. However, these platforms treat simulation as an external black box without query-level optimization, missing opportunities for incremental computation. Time-series databases like TimescaleDB [4] and InfluxDB excel at storing and querying historical observations with compression and time-based partitioning, but lack native support for simulation integration and scenario management, requiring application-level orchestration to combine observed and simulated data. Surrogate modeling approaches using Gaussian Process regression [8] or neural networks can approximate expensive simulations but require substantial offline training data and may not generalize well to building-specific characteristics outside the training distribution. Approximate query processing [2] addresses performance-accuracy tradeoffs through sampling and sketching, but focuses on statistical aggregation over static data rather than simulation-based computation. TwinDB treats scenarios as first-class database citizens with query-level optimization and learns sensitivity coefficients online without cold-start delays. It exploits physics-based linearity for incremental maintenance, with formal accuracy guarantees derived from physics-informed error bounds.

6 Conclusion

We demonstrate TwinDB, a scenario-aware database system that enables interactive what-if analysis for digital twins. Through Incremental Scenario Maintenance, TwinDB achieves orders of magnitude speedup for scenario exploration while maintaining error below 0.1%, transforming digital twin analytics from a batch-oriented process to an interactive, visual exploration experience. Our demonstration showcases practical workflows for building energy analysis, from visual retrofit exploration to programmatic querying and accuracy verification, and invites conference attendees to experience the system with both pre-loaded Danish building data and their own custom datasets. Future work includes extending ISM to handle non-linear simulations and supporting multi-model composition for complex systems.
Resources: Code at https://github.com/xiufengliu/TwinDB.

Acknowledgments

This work was supported by the OPTIX project (CET Partnership, Ref. CETP-FP-2023-00114), funded by Innovation Fund Denmark and the European Commission (Grant No. 101069750), the Marie Skłodowska-Curie Postdoctoral Individual Fellowship (Grant No. 101154277), and the RE-INTEGRATE project, funded by the European Union's Horizon Europe Research and Innovation Programme (Grant No. 101118217).

References

[1] Amazon Web Services. Aws iot twinmaker. https://aws.amazon.com/iot-twinmaker/, 2021.
[2] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. Approximate query processing: No silver bullet. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 511–519, 2017.
[3] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems: New findings and approaches*, pages 85–113. Springer, 2016.
[4] Søren Kejser Jensen, Torben Bach Pedersen, and Christian Thomsen. Time series management systems: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2581–2600, 2017.
[5] Microsoft Corporation. Azure digital twins. https://azure.microsoft.com/en-us/products/digital-twins/, 2021.
[6] Markus Schaffer, Thea Tvedebrink, Muhyiddine Jradi, and Anna Marszal-Pomianowska. Three years of hourly data from 3021 smart heat meters installed in danish residential buildings. *Scientific Data*, 9(1):420, 2022.
[7] Fei Tao, Jiangfeng Cheng, Qinglin Qi, Meng Zhang, He Zhang, and Fangyuan Sui. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94(9):3563–3576, 2018.
[8] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.