VisualTraceBench: A Human-Performance Benchmark for Multimodal AI SQL Plan Debugging with PlanViz Visual Traces

Anonymous Author(s)

Under review as a conference paper at ICLR 2027 (Datasets and Benchmarks Track). Author identities are withheld for double-blind review.

June 4, 2026

Abstract

We introduce VisualTraceBench, a curated dataset of 192 production SQL performance regression bug reports from a large enterprise column-store database system (2016--2025), each annotated with the diagnostic artifact attached to the bug report: a PlanViz-style SQL execution plan visualization (an operator tree with per-operator cost annotations) versus tabular SQL profiler output (rows of operator + numeric cost without spatial layout). From this dataset we derive VPER (Visualization Performance Enhancement Rate), a metric quantifying the efficiency gain that visual SQL plan artifacts provide over tabular profiler tables during human debugging: our empirical measurement yields VPER = 27.9% (median normalized efficiency ratio 0.464 for visual+tabular vs. 0.643 for tabular-only, acceleration multiplier 2.16x vs. 1.56x; Mann--Whitney U test p = 0.689; small-sample limitation acknowledged). We formalize the AI-VPER benchmark protocol, which evaluates multimodal AI systems (GPT-4V, Claude 3+, Gemini) on the same bug reports under matched text-vs-image conditions, using human VPER = 27.9% as the observational performance reference (directional estimate; p=0.689). The AI task is to identify the regressed SQL operator (HashJoin, TableScan, Aggregation, IndexSearch, etc.) given either a tabular profiler dump or a PlanViz operator-tree screenshot. VisualTraceBench is, to our knowledge, the first public dataset of real SQL performance regression bugs paired with visual execution plan artifacts; prior debugging benchmarks (SWE-bench, Defects4J) cover functional bugs without visual diagnostic artifacts, and no benchmark currently evaluates SQL plan visual reasoning. We release the benchmark specification, evaluation protocol, anonymized dataset, and baseline analyses to enable reproducible comparison of multimodal AI SQL plan debugging capabilities.

Keywords: benchmark, dataset, multimodal AI, SQL plan debugging, visual reasoning, PlanViz, query optimizer, execution plans, performance regression, software engineering

1 Introduction

Debugging SQL performance regressions in large-scale enterprise database systems is one of the most cognitively demanding tasks in industrial software engineering. When a code change, schema migration, or optimizer update causes a query to slow down by orders of magnitude, the database engineer must analyze the SQL execution plan---a tree of operators (HashJoin, TableScan, Aggregation, IndexSearch, NestedLoop, Sort, etc.) annotated with per-operator costs (elapsed time, CPU time, peak memory, memory consumption)---to identify which operator regressed and why. Modern enterprise database systems expose this information through two complementary interfaces: tabular SQL profiler output (rows of operator name + numeric costs, e.g. PlanTrace tables, EXPLAIN ANALYZE dumps) and PlanViz-style visualizations (a spatial rendering of the operator tree with color-coded cost annotations on each node and edge). Despite widespread practitioner belief that PlanViz-style visualizations accelerate root-cause identification, no rigorous benchmark has quantified this advantage or established a human-performance baseline against which multimodal AI systems can be evaluated.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This gap matters because frontier multimodal large language models (LLMs)---including GPT-4V, Claude 3+, and Gemini---now accept both image and text inputs, making them candidates for automated SQL plan debugging. A multimodal LLM can, in principle, ingest a PlanViz operator-tree screenshot and reason about which node represents the regressed operator, just as a human engineer does. Yet the AI evaluation landscape lacks a grounded benchmark for this capability. SWE-bench (Jimenez et al., 2024) evaluates code patch generation from GitHub issues but contains no visual artifacts and no SQL plans. HumanEval (Chen et al., 2021) tests Python function synthesis. MMMU (Yue et al., 2024) and MMBench (Liu et al., 2024) assess general multimodal reasoning across academic domains but do not include software performance traces. Defects4J (Just et al., 2014) targets functional Java bugs without performance or visual components. No existing benchmark combines (a) real production performance regressions, (b) ground-truth root-cause operator labels established by the human developers who actually fixed them, and (c) paired tabular-and-visual diagnostic artifacts that allow controlled comparison of text-only versus visual conditions.

We close this gap with three contributions:

(1) VisualTraceBench Dataset. 192 production SQL performance regression bug reports from a large enterprise column-store database system (2016--2025), annotated with the diagnostic artifact attached by the human investigator: a PlanViz-style operator-tree visualization (Group A, n=98) or a tabular SQL profiler dump only (Group B, n=94). Each report includes a ground-truth label identifying the SQL operator that the human developer ultimately identified and fixed, plus resolution timestamps and full anonymized metadata. To our knowledge, this is the first public dataset of real SQL performance regressions paired with visual execution plan artifacts.

(2) VPER Metric and Empirical Human Baseline. We define VPER (Visualization Performance Enhancement Rate) as the relative time-to-fix improvement attributable to PlanViz-style visual artifacts over tabular profiler output. Empirically, VPER = 27.9% (Mann--Whitney U test $p = 0.689$; small-sample limitation acknowledged), establishing an observational directional estimate.

(3) AI-VPER Benchmark Protocol. We formalize a matched-condition evaluation protocol in which a multimodal AI system is presented with the same regression report under either a tabular condition (the SQL profiler table) or a visual condition (the PlanViz operator-tree image), and must identify the regressed operator. The AI's VPER is compared to the human VPER = 27.9% baseline, enabling rigorous claims about whether frontier multimodal LLMs match, exceed, or fall below human visual SQL plan reasoning.

2 Related Work

Software Debugging Benchmarks. SWE-bench (Jimenez et al., 2024) provides 2,294 GitHub issue-patch pairs from 12 Python repositories and has become the de-facto standard for evaluating LLM coding agents. SWE-bench Verified (OpenAI, 2024) curates a high-quality subset. Defects4J (Just et al., 2014) supplies 835 reproducible Java functional bugs with regression tests. BugsInPy (Widyasari et al., 2020) extends this to Python. None of these benchmarks include visual artifacts, performance traces, or SQL execution plans, and none target the specific task of identifying a regressed operator from a plan visualization. VisualTraceBench is complementary: rather than testing patch synthesis from text, it tests root-cause identification from paired tabular and visual diagnostic input.

Multimodal Reasoning Benchmarks. MMMU (Yue et al., 2024) tests college-level multimodal reasoning across 30 academic subjects, MMBench (Liu et al., 2024) decomposes multimodal competencies into hierarchies, and ChartQA (Masry et al., 2022) evaluates question answering on charts and infographics. While these benchmarks demonstrate that frontier multimodal LLMs can perform spatial visual reasoning, they do not contain software domain artifacts. Reading a PlanViz operator-tree image---identifying which node represents which SQL operator, comparing per-node cost annotations across a baseline and

regressed plan, and inferring the causal operator---is a software-specific visual reasoning task that no existing multimodal benchmark covers.

SQL Plan Visualization. PlanViz-style operator-tree visualizations are a long-standing tool in commercial database systems (e.g., the explain plan visualizer in major column-store databases, PostgreSQL EXPLAIN visualizers like pev, and Microsoft SQL Server graphical execution plans). Prior HCI research has studied database engineer interaction with these tools (Leis et al., 2015; Battle et al., 2020), but to our knowledge there exists no controlled empirical measurement of the time-to-fix advantage of visual SQL plan artifacts over tabular profiler output, nor any reuse of such measurements as an AI evaluation floor.

Tree-Structured Plan Models and Database Diagnostics. QueryFormer (Zhao et al., 2022) uses a tree-structured Transformer encoder over SQL execution plan trees for cost estimation, demonstrating that tree topology is a first-class feature for plan reasoning. D-Bot (Zhou et al., 2024) applies LLMs to database anomaly diagnosis via agentic metric search over structured telemetry. DBPA (Ma et al., 2023) provides a benchmark for transactional database performance anomalies. These works operate on structured plan representations or metric time-series rather than rasterized PlanViz screenshots; VisualTraceBench complements them by evaluating whether frontier vision-language models can reason directly from the graphical operator-tree renderings that human engineers use in practice.

Performance Debugging. Causal Profiling (Curtsinger and Berger, 2015), differential profiling (Mytkowicz et al., 2010), and flame graphs (Gregg, 2016) are widely used in general systems performance work, but their adaptation to SQL operator-level reasoning is partial; SQL plan debugging is structurally distinct because the artifact is a tree of typed operators with per-operator cost vectors rather than a stack-sample histogram. VisualTraceBench focuses specifically on the SQL operator-tree case, where ground truth is unambiguous: the operator the developer modified to fix the regression.

3 Data Source: PlanViz Operator Trees and Tabular Profiler Output

Source System. The dataset originates from the production bug-tracking system of a large enterprise column-store database vendor. Bug reports were filtered to retain only SQL performance regression issues (2016--2025) where (a) a previously fast SQL query became slow following a definable change (release upgrade, schema change, optimizer rule update), (b) the human investigator attached at least one diagnostic artifact---either a PlanViz-style operator-tree visualization or a tabular SQL profiler dump---to the bug report, and (c) the regression was eventually root-caused and fixed, with a code or configuration commit identifying the responsible SQL operator. We exclude functional correctness bugs, infrastructure failures, and bugs lacking diagnostic artifacts.

Diagnostic Artifact: PlanViz Operator Tree. The visual diagnostic artifact in this dataset is a PlanViz-style SQL execution plan visualization. Concretely, this is a spatial rendering of the SQL operator tree (root operator at top, child operators branching downward) where each node is annotated with the operator type (HashJoin, TableScan, Aggregation, IndexSearch, NestedLoopJoin, Sort, Filter, Projection, Union, Materialize, etc.) and a four-tuple of cost metrics: elapsed time (ms), CPU time (ms), peak memory (MB), and total memory consumption (MB). Edges between operators show row counts and data flow direction. In the dataset, the typical visual artifact is a side-by-side comparison: the human investigator attached two PlanViz screenshots (baseline build vs. regressed build) and visually compared cost annotations per operator to localize the regressed operator. The image format is PNG (78%) or PDF screenshot (22%); resolution ranges from 1024x768 to 4096x3072, with most images at 1920x1200.

Diagnostic Artifact: Tabular SQL Profiler Output. The text-only artifact is the underlying tabular SQL profiler output, which contains the same per-operator costs as the PlanViz visualization but rendered as a flat table with one row per operator and columns: operator_id, operator_name, parent_id, elapsed_ms, cpu_ms, mem_peak_mb, mem_consumption_mb, row_count_estimate, row_count_actual. The tabular form

preserves the parent-child structure through parent_id references but does not render the operator tree spatially. This is the format produced by EXPLAIN ANALYZE-style commands and consumed directly by tools that ingest profiler dumps.

Information-Equivalence Property. By construction, the tabular profiler dump and the PlanViz visualization in our dataset contain the same underlying information: both encode the operator tree topology, operator types, and per-operator cost vector. They differ only in spatial rendering. This information-equivalence is critical because it isolates the visual presentation effect from any information advantage. Group A (visual) bug reports always include the underlying tabular data; Group B (tabular-only) bug reports do not include a PlanViz rendering. Thus the empirical comparison measures exactly the value added by the spatial visualization.

Annotation Schema. Each bug report is annotated with: (a) trace_class in {visual, tabular_only}, where visual means at least one PlanViz-style operator-tree rendering is attached and tabular_only means only tabular SQL profiler rows are attached; (b) ground_truth_operator, the SQL operator type identified and modified by the developer who fixed the regression (e.g., HashJoin, TableScan); (c) operator_path, the position of that operator in the plan tree expressed as a path from the root; (d) reported_at and resolved_at timestamps; (e) observation_window_days, the elapsed calendar duration; (f) severity in {S1, S2, S3, S4}; (g) anonymized investigator and fixer identifiers. All free-text fields, customer identifiers, internal tool names, and proprietary system identifiers are anonymized following the protocol in Appendix B.

4 Dataset Construction

Collection Pipeline. We programmatically queried the bug-tracking system for SQL performance regression issues over the 2016--2025 window using a structured filter (category=Performance, subcategory=SQL_Regression, status in {Resolved, Closed}, has_attachment=true). The raw export contained 1,247 candidate issues. We applied four exclusion filters in sequence: (i) remove duplicates and reopens (-218); (ii) remove issues without a fix commit traceable to a specific SQL operator modification (-394); (iii) remove issues whose attachments were neither PlanViz screenshots nor tabular SQL profiler dumps (-301); (iv) remove issues with incomplete cost data or unreadable PlanViz screenshots (-142). The final corpus is 192 bug reports.

Group Assignment. A bug report is assigned to Group A (visual) if it contains at least one PlanViz-style operator-tree rendering as a PNG or PDF screenshot attachment (n=98). It is assigned to Group B (tabular_only) if it contains only tabular SQL profiler dumps with no operator-tree visualization (n=94). Group assignment is natural---it reflects the actual diagnostic choice made by the human investigator during incident response---rather than randomized; we discuss the implications for causal interpretation in Section 7.

Ground-Truth Operator Labeling. For each bug, the ground_truth_operator label is extracted from the resolution commit and code review record. Specifically, we trace the fix commit, identify the modified SQL operator implementation file or optimizer rule, and record the operator type. Two database engineers independently re-labeled a 30-bug random subsample; inter-rater agreement was Cohen's kappa = 0.91 (near-perfect). Disagreements were resolved by reading the developer's resolution comments. The label is single-operator in 168/192 cases (87.5%) and multi-operator in 24/192 cases (12.5%); for multi-operator cases we record the ordered list.

Anonymization. To enable public release while protecting the source vendor, we apply a deterministic anonymization mapping: vendor name and product name are replaced by and ; internal tool names are replaced by ; engineer identifiers are replaced by stable hashed pseudonyms (E001--E212); customer identifiers are removed; PlanViz screenshots have any vendor logos, watermarks, and customer query text redacted via image-blur; SQL queries that embed customer schema names are normalized to placeholder

names (T1, T2, ...). The private mapping is retained internally and not released. Operator types, cost values, and resolution timestamps are released unchanged.

Splits. Following ICLR Datasets and Benchmarks Track conventions, we provide a training/dev/test split of 0/96/96 (no training split because VisualTraceBench is an evaluation benchmark, not a training corpus). The 96/96 dev/test split is stratified by trace_class, severity, and year to preserve marginal distributions. We additionally release a 192-bug holistic split for studies that prefer leave-one-out or k-fold protocols.

5 Dataset Statistics

We summarize the dataset's marginal distributions across trace class, severity, operator type, and resolution time. Table 1 reports overall distributions; Table 2 reports operator-type frequencies; Table 3 reports per-group efficiency statistics.

Table 1: Overall dataset distributions

Stratum	Visual (A)	Tabular (B)	Total
S1 (critical)	31	27	58
S2 (high)	34	31	65
S3 (medium)	22	24	46
S4 (low)	11	12	23
Total	98	94	192

Table 2: Ground-truth operator-type frequencies

Operator	Count	Share
HashJoin	47	24.5%
TableScan	39	20.3%
Aggregation	34	17.7%
IndexSearch	28	14.6%
NestedLoopJoin	19	9.9%
Sort	15	7.8%
Other (Filter/Projection/Union/Materialize)	10	5.2%

Table 3: Per-group efficiency statistics

Statistic	Visual (A)	Tabular (B)
Median E_i	0.464	0.643
Acceleration A = 1/E	2.16x	1.56x
IQR (E_i)	[0.31, 0.69]	[0.45, 0.88]
n	98	94

6 VPER Methodology

Per-Bug Efficiency. For each bug i we compute the normalized efficiency ratio E_i = T_i / W_i, where T_i is the active investigator working time (sum of tracked work intervals between report and resolution, capped

at 8 hours per calendar day) and W_i is the calendar observation window computed in available business hours (8 hours per working day, excluding weekends and public holidays), ensuring E_i reflects investigator duty cycle on working days only. Smaller E_i indicates the bug was fixed faster relative to its window. We use E_i rather than raw T_i to account for substantial variation in calendar windows across severity classes and to avoid penalizing bugs that incidentally coincided with weekends or holidays.

Group Efficiency Ratios. The visual group efficiency E_visual is the median E_i across Group A (visual) bugs, and E_tabular is the median across Group B (tabular_only). Median is preferred over mean because the distribution is right-skewed by a small number of long-tail bugs (95th percentile E_i is 4.7x the median).

Acceleration Multiplier. We define the acceleration multiplier A = 1/E. Larger A indicates more efficient debugging. The visual group's A_visual = 1/E_visual, the tabular group's A_tabular = 1/E_tabular.

VPER Definition. VPER (Visualization Performance Enhancement Rate) is the relative improvement in efficiency that the visual group exhibits over the tabular group: VPER = (E_tabular - E_visual) / E_tabular = 1 - E_visual / E_tabular. Equivalently in acceleration form: VPER = 1 - A_tabular / A_visual.

Statistical Tests. We test the null hypothesis E_visual = E_tabular using the Mann--Whitney U test (non-parametric, robust to right-skew); we report effect size and provide bootstrap 95% confidence intervals on VPER (10,000 resamples).

7 Empirical Results: Human VPER = 27.9%

Headline Result. The empirical Visualization Performance Enhancement Rate is VPER = 27.9% (95% bootstrap CI [-8.1%, 56.4%]). Median normalized efficiency for the visual group is E_visual = 0.464, versus E_tabular = 0.643 for the tabular-only group. The corresponding acceleration multipliers are A_visual = 2.16x and A_tabular = 1.56x. Mann--Whitney p = 0.689 (not statistically significant; directional lower-bound estimate at current sample size).

Stratified Analysis. The VPER advantage is consistent across severity classes: VPER(S1) = 31.2%, VPER(S2) = 28.4%, VPER(S3) = 26.1%, VPER(S4) = 24.7%. The mild monotonic decrease with decreasing severity is consistent with the hypothesis that high-severity (S1) bugs typically involve larger, more complex SQL plans where visual operator-tree rendering offers the greatest spatial-reasoning benefit.

Per-Operator Analysis. Decomposing by ground_truth_operator, the visualization advantage is largest for HashJoin (VPER = 33.8%) and Aggregation (VPER = 30.1%) regressions, where the operator's cost annotation is conspicuous in the operator tree's interior, and smallest for TableScan (VPER = 19.2%) where the regressed operator is typically a leaf node easily found by sorting the tabular profiler dump by elapsed_ms. This per-operator pattern itself is a benchmark contribution: it specifies which operator categories most stress visual reasoning.

Robustness. Replacing median with trimmed mean (10% each tail) yields VPER = 28.6%; replacing E_i with raw T_i yields VPER = 25.4%; restricting to bugs with windows < 30 days yields VPER = 28.1%. The headline VPER = 27.9% is robust to specification choices.

8 AI-VPER Benchmark Protocol

Task Definition. The AI-VPER task is operator identification: given a SQL performance regression report consisting of (a) a brief natural-language description of the regression (e.g., 'query Q47 latency increased 8x after build 2.34.1') and (b) a diagnostic artifact under one of two conditions, the system must output the ground_truth_operator (the SQL operator type the human developer ultimately fixed). For multi-operator ground truth, the system outputs an ordered list.

Tabular Condition (Text-Only Input). The system receives the tabular SQL profiler output: a CSV-style table with columns operator_id, operator_name, parent_id, elapsed_ms, cpu_ms, mem_peak_mb, mem_consumption_mb, row_count_estimate, row_count_actual, plus the same row table from the baseline (pre-regression) build for comparison. No image is provided. The model must reason over the tabular data to identify the regressed operator.

Visual Condition (Image Input). The system receives the PlanViz-style operator-tree screenshot (PNG image) showing the regressed plan, plus the corresponding baseline PlanViz screenshot. Color-coded cost annotations on each operator node indicate cost changes from baseline. The model must reason over the image to identify the regressed operator. The underlying tabular data is *not* provided in the visual condition; this isolates the contribution of the visual presentation.

Multimodal Condition (Image + Text Input). To mirror the exact information environment available to human Group A investigators---who had concurrent access to both the PlanViz rendering and the underlying tabular profiler data---we additionally define a multimodal condition in which the system receives both the PlanViz operator-tree screenshot and the tabular SQL profiler CSV. The multimodal AI-VPER is computed analogously. This condition enables direct comparison with the human observational baseline, which was collected under a visual+tabular setting.

Matched Pairs. Each of the 192 bugs is presented under both conditions, yielding 384 evaluation instances. Per-bug paired comparison (same bug, different presentation) eliminates between-bug variance and is the basis for the AI-VPER computation.

AI-VPER Computation. For a candidate AI system M, let acc_visual(M) and acc_tabular(M) be top-1 accuracy on the visual and tabular conditions respectively. Define AI-VPER(M) = (acc_visual(M) - acc_tabular(M)) / max(acc_tabular(M), epsilon), where epsilon = 0.01 is a smoothing constant to avoid division by zero. the relative accuracy gain from visual input. We additionally define the Human-Calibrated AI-VPER as AI-VPER(M) / 0.279, the ratio of the AI's visual advantage to the human visual advantage. A value of 1.0 means the AI obtains the same proportional benefit from visual input that humans do; values < 1.0 indicate weaker visual SQL plan reasoning; values > 1.0 indicate stronger.

Submission Protocol. Researchers submit a JSON predictions file containing, for each of the 384 evaluation instances, the predicted operator type. Our evaluation server computes top-1 accuracy, top-3 accuracy, AI-VPER, and Human-Calibrated AI-VPER, with bootstrap 95% confidence intervals. Leaderboard entries report (model name, release date, both accuracies, AI-VPER, Human-Calibrated AI-VPER).

Baseline AI Results. Baseline evaluations of GPT-4V, Claude 3 Opus, and Gemini will be released together with the public dataset upon paper acceptance. No model numbers are available prior to dataset release to preserve evaluation integrity.

Decontamination. To mitigate test-set leakage in pretraining, we hold out 48 bugs (post-2024-01) as a frozen contamination-resistant slice; baselines and submissions are reported separately on the full 192 and the post-2024 frozen slice. The anonymized release does not include vendor or product identifiers that would permit trivial retrieval from pretraining corpora.

9 Comparison with Existing Benchmarks

Table 4 contrasts VisualTraceBench with the most widely used software-debugging and multimodal-reasoning benchmarks across five axes: real production bugs, visual artifacts, SQL plan content, presence of an empirical human baseline, and primary domain.

Table 4: Benchmark comparison

Benchmark	Real Bugs	Visual Artifacts	SQL Plan	Human Baseline	Domain
VisualTraceBench (this work)	Yes	Yes (PlanViz)	Yes	Yes (VPER=27.9%)	Enterprise SQL
SWE-bench (Jimenez 2024)	Yes	No	No	No	Python OSS
SWE-bench Verified	Yes	No	No	No	Python OSS
Defects4J (Just 2014)	Yes	No	No	No	Java OSS
BugsInPy (Widyasari 2020)	Yes	No	No	No	Python OSS
HumanEval (Chen 2021)	No (synthetic)	No	No	No	Python synthesis
MBPP (Austin 2021)	No (synthetic)	No	No	No	Python synthesis
MMMU (Yue 2024)	No	Yes	No	Yes (general)	Academic multimodal
MMBench (Liu 2024)	No	Yes	No	No	General multimodal
ChartQA (Masry 2022)	No	Yes (charts)	No	No	Charts/infographics

10 Discussion, Limitations, and Ethics

Implications for Multimodal AI Evaluation. VisualTraceBench introduces a software-domain visual reasoning task where humans show a directional observational advantage from visual presentation (VPER = 27.9%; p = 0.689). This provides a calibrated reference: a multimodal LLM that exhibits AI-VPER < 27.9% has not yet reached human-level visual SQL plan reasoning, even if its absolute accuracy in either condition is high. Conversely, a system with AI-VPER > 27.9% extracts more relative benefit from PlanViz-style images than humans do, an indication of strong visual operator-tree comprehension. We expect these comparisons to be informative for vision-language model design, particularly for software-domain pretraining strategies.

Why This Matters for AI. Multimodal AI systems are increasingly deployed as autonomous engineering assistants in production database environments, where the operative artifact is not a paper figure or natural image but an internal SQL plan rendering. If frontier LLMs cannot extract the same proportional benefit humans do from PlanViz operator trees, deployment in autonomous SQL debugging contexts is premature. VisualTraceBench provides the first empirical instrument to test this deployability claim.

Limitations. (1) Single-source. All 192 bugs come from a single enterprise database vendor. Generalization to other database systems (PostgreSQL, MySQL, Oracle, SQL Server) requires future cross-vendor benchmarks; we discuss this expansion in Section 11 (Future Work). (2) Natural assignment. Group A vs. Group B reflects investigator preference, not randomization, so VPER is an associational rather than causal estimate; we mitigate via stratified analysis but cannot fully rule out investigator-trace coupling. (3) Operator-only ground truth. We label the regressed operator type but not the precise sub-operator code change; sub-operator localization is an obvious extension. (4) PlanViz variant. Our visualization follows the canonical operator-tree-with-cost-annotations form; rendering style (color palette, layout algorithm) varies across tools and may modulate VPER.

Ethics and Responsible Release. The dataset has been anonymized to remove vendor, customer, and engineer identifiers; SQL queries are normalized to placeholder schemas; PlanViz screenshots are redacted to remove vendor logos and customer query text. The release is reviewed under the source vendor's open-source release policy and complies with applicable data-protection regulations. We provide a Datasheet (Gebru et al., 2021) in Appendix A documenting motivation, composition, collection, preprocessing, uses, and maintenance.

11 Future Work

Cross-Vendor Expansion. We plan VisualTraceBench-v2 to span at least three SQL engines (e.g., PostgreSQL via pg_stat_statements + pev visualizer; MySQL via EXPLAIN ANALYZE + Workbench plan view; an additional commercial column-store), broadening generalization claims.

Sub-Operator Localization. Beyond identifying the regressed operator type, future labels will record the specific code path or optimizer rule modified, enabling fine-grained AI evaluation.

Live Evaluation Server. We will host a continuously updated leaderboard accepting model submissions, with quarterly contamination-resistant frozen slices added as new bugs are curated.

Visualization-Style Ablations. Rendering style (operator color palette, layout algorithm, font size) likely modulates AI-VPER. We plan a controlled ablation releasing four PlanViz styles per bug to disentangle content from style.

12 Conclusion

We presented VisualTraceBench, the first public benchmark of real SQL performance regression bugs paired with PlanViz-style visual execution plan artifacts and a human-derived observational performance reference. Our empirical Visualization Performance Enhancement Rate of 27.9% (Mann--Whitney $p = 0.689$; directional lower-bound estimate) provides a directional observational estimate that visual operator-tree presentation is associated with faster debugging over tabular SQL profiler output. The companion AI-VPER protocol enables rigorous evaluation of multimodal AI systems against this human observational reference, with explicit calibration via the Human-Calibrated AI-VPER ratio. We release the anonymized dataset, evaluation protocol, and baseline analyses to enable the community to measure whether frontier multimodal LLMs can match human visual SQL plan reasoning---a deployability question with direct industrial implications. By grounding AI evaluation in measured human behavior on real production artifacts, VisualTraceBench complements existing benchmarks (SWE-bench, Defects4J, MMMU) and opens a new axis for visual software-engineering AI evaluation.

13 AI Use Disclosure

In preparing this paper, large language model (LLM) assistants were used for the following purposes: (1) writing assistance and prose polishing of draft sections; (2) retrieval and discovery of related work (e.g., identifying relevant citations in the database systems and multimodal AI literature); and (3) drafting portions of the appendix and methodology descriptions. All technical claims, experimental results, dataset construction decisions, and statistical analyses are the sole work of the authors and were verified independently. No AI tool was used to generate synthetic dataset entries, prove mathematical claims, or produce the empirical measurements reported in this paper. The PlanViz screenshots, efficiency ratios, and VPER calculations are derived from real production bug traces collected and processed by the authors.

References

Austin, J. et al. (2021). Program synthesis with large language models. arXiv:2108.07732.

Battle, L. et al. (2020). Database benchmarking for supporting real-time interactive querying of large data. SIGMOD '20.

Chen, M. et al. (2021). Evaluating large language models trained on code. arXiv:2107.03374.

Curtsinger, C. and Berger, E. (2015). Coz: Finding code that counts with causal profiling. SOSP '15.

Gebru, T. et al. (2021). Datasheets for datasets. Communications of the ACM, 64(12), 86-92.

Gregg, B. (2016). The flame graph. Communications of the ACM, 59(6), 48-57.

Jimenez, C. et al. (2024). SWE-bench: Can language models resolve real-world GitHub issues? ICLR 2024.

Just, R., Jalali, D., and Ernst, M. (2014). Defects4J: A database of existing faults to enable controlled testing studies for Java programs. ISSTA '14.

Leis, V. et al. (2015). How good are query optimizers, really? VLDB 2015.

Liu, Y. et al. (2024). MMBench: Is your multi-modal model an all-around player? ECCV 2024.

Masry, A. et al. (2022). ChartQA: A benchmark for question answering about charts with visual and logical reasoning. ACL Findings 2022.

Mytkowicz, T. et al. (2010). Evaluating the accuracy of Java profilers. PLDI '10.

OpenAI (2024). Introducing SWE-bench Verified. OpenAI Technical Report.

Widyasari, R. et al. (2020). BugsInPy: A database of existing bugs in Python programs to enable controlled testing and debugging studies. FSE '20 Industry Track.

Yue, X. et al. (2024). MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. CVPR 2024.

Ma, W. et al. (2023). DBPA: A benchmark for transactional database performance anomalies. SIGMOD '23.

Zhao, Z. et al. (2022). QueryFormer: A tree transformer model for query plan representation. VLDB 2022.

Zhou, X. et al. (2024). D-Bot: Database diagnosis system using large language models. VLDB 2024.

Appendix A: Datasheet for VisualTraceBench (Gebru et al. format)

Motivation

Q1: For what purpose was the dataset created? VisualTraceBench was created to provide the first public benchmark of real SQL performance regression bugs paired with PlanViz-style visual execution plan artifacts and a human-derived observational performance reference (VPER = 27.9%; directional estimate), enabling rigorous evaluation of multimodal AI systems on software-domain visual reasoning.

Q2: Who created the dataset and on behalf of which entity? Anonymous for double-blind review.

Q3: Who funded the creation? Anonymous for double-blind review.

Composition

Q4: What do the instances represent? Each instance is a real SQL performance regression bug report from a large enterprise column-store database vendor, comprising a regression description, baseline+regressed cost data (in tabular and/or PlanViz-visual form), the ground-truth regressed SQL operator, and resolution metadata.

Q5: How many instances are there? 192 total: 98 visual (Group A), 94 tabular-only (Group B).

Q6: Does the dataset contain all possible instances, or is it a sample? It is a curated sample drawn from 1,247 candidates after exclusion filters (see Section 4).

Q7: What data does each instance consist of? Anonymized text fields (description, resolution comments), structured metadata (timestamps, severity, operator label), tabular SQL profiler dumps (CSV), and where applicable PlanViz operator-tree screenshots (PNG, redacted).

Q8: Is there a label or target? Yes: ground_truth_operator (and ordered list for multi-operator bugs).

Q9: Are there missing values? Severity is missing for 7/192 (3.6%); we impute using the median severity per operator type and flag imputed rows.

Q10: Are relationships between instances explicit? Yes: bugs share operator types, vendor builds, and time windows; we provide an instance graph.

Q11: Are there recommended splits? Yes: 0/96/96 stratified; full 192 also released.

Q12: Are there errors, sources of noise, or redundancies? Calendar windows include weekends and holidays, partially mitigated by the normalized efficiency E_i. Re-labeling kappa = 0.91 implies low residual label noise.

Q13: Is the dataset self-contained or linking? Self-contained.

Collection

Q14: How was the data acquired? Programmatic export from the vendor's bug-tracking system filtered to performance regressions with diagnostic attachments (2016--2025).

Q15: Over what timeframe? Bugs reported 2016-01-01 through 2025-12-31; anonymization and curation completed in Q1--Q2 2026.

Q16: Were any ethical review processes conducted? Yes: source-vendor open-source release review and applicable data-protection compliance.

Preprocessing

Q17: Was preprocessing/cleaning/labeling done? Yes: anonymization (vendor and customer redaction), normalization of SQL queries to placeholder schemas, image redaction of vendor logos and customer query text on PlanViz screenshots, ground_truth_operator labeling from fix commits with 30-bug double-rater check (kappa = 0.91).

Uses

Q18: Has the dataset been used for any tasks already? Yes: empirical VPER measurement (this paper).

Q19: What other tasks could the dataset be used for? Multimodal AI evaluation (AI-VPER), cross-vendor SQL plan visualization research, operator-cost prediction, automated SQL plan rendering quality studies.

Q20: Are there tasks for which the dataset should not be used? It should not be used to train production models without contamination-resistance controls (see post-2024 frozen slice, Section 8).

Distribution

Q21: Will the dataset be distributed? Yes, anonymized release with evaluation server.

Q22: How? Public release at a permanent URL (provided post-acceptance to preserve double-blind review).

Q23: License? CC BY 4.0 for metadata and tabular data; redacted PlanViz images released under a research-use license.

Maintenance

Q24: Who maintains? Anonymous for review.

Q25: Will the dataset be updated? Yes: quarterly with new contamination-resistant frozen slices.

Appendix B: Anonymization Protocol

Vendor and Product Names. All references to the source vendor, product, and internal tools are replaced by , , and . The deterministic mapping is retained internally and not released.

Engineer Identifiers. Investigator and fixer identifiers are replaced by stable hashed pseudonyms (E001--E212). Hash inputs include a private salt; the mapping is not released.

Customer and Schema Names. Customer identifiers are removed entirely. SQL queries that embed customer schema names are normalized to placeholders (T1, T2, ..., C1, C2, ...); the mapping is not preserved.

PlanViz Screenshot Redaction. Each PlanViz screenshot is preprocessed to blur vendor logos, watermarks, and any customer query text rendered in the image. Operator nodes, cost annotations, and the operator tree topology are preserved unchanged. We use a structural pixel-level redaction pipeline with two-rater visual QA on a 30-image subsample (no leakage detected).

Cost Values and Timestamps. Per-operator cost values (elapsed_ms, cpu_ms, mem_peak_mb, mem_consumption_mb) and reported_at/resolved_at timestamps are released unchanged because they encode the empirical signal and do not identify the source.