Journal of Green Energy and Transition to Sustainability
ISSN: 2949-8325
Vol 3, Issue 2, 2024

Cost-Based Optimizer Plan Drift Analysis in Oracle 19c and 23c Engines

Rowan Beckford

Abstract

Hybrid OLTP-reporting workloads commonly experience performance instability caused by cost-based optimizer plan drift, where execution plans shift unpredictably despite no changes to SQL text. This study analyzes plan drift behavior in Oracle 19c and 23c engines under mixed transactional and reporting conditions. The results show that drift is primarily triggered by evolving data distributions, dynamic SQL query shapes, and concurrency-induced resource contention. Oracle 23c demonstrates improved runtime feedback mechanisms that reduce short-term volatility and promote faster plan convergence, but drift persists under rapidly fluctuating workloads. Effective mitigation requires selective plan stabilization, controlled query pattern design, and deliberate statistics lifecycle management to maintain predictable performance while preserving optimizer flexibility. The findings provide a structured foundation for diagnosing and minimizing plan drift in modern enterprise deployments.

Keywords: cost-based optimizer, plan drift, hybrid workloads

1. Introduction

Hybrid database environments that support both transactional workloads and analytical reporting often experience unpredictable performance shifts due to cost-based optimizer plan drift. In Oracle 19c and 23c engines, adaptive optimization recalculates access paths based on changing runtime conditions, workload patterns, and data characteristics. When OLTP and reporting actions occur on the same tables, even small variations in session state or data composition can produce markedly different execution plans, leading to inconsistent response times and operational instability [1, 2]. Cloud deployment further amplifies this effect by introducing elastic resource baselines, increasing the optimizer's sensitivity to fluctuations in system load [3]. Variations in cardinality estimation accuracy often underpin these transitions, where minor misestimates distort join and access path decisions [4], sometimes resulting in cascading performance regressions across execution layers [5].

In mixed workloads, transactional queries generally favor indexed lookups optimized for minimal latency, while reporting queries favor broader scans and join-heavy plans optimized for throughput. Oracle APEX applications commonly integrate these two patterns within the same interface, where users enter operational data and also generate real-time reports. This causes the optimizer to alternate between competing cost models during workload fluctuations, especially when background reporting operations and live entry tasks run concurrently [6, 7]. Without stability mechanisms, these context-dependent decisions can cause plan volatility during ordinary usage [8].

Data distribution changes introduce further sources of drift. In hybrid systems where transactional insertion rates are high and summary aggregations are periodically recalculated, table statistics may become misaligned with current data states. This affects join order selection, predicate selectivity, and index usage patterns [9]. Learned cardinality estimation models have demonstrated that traditional histograms may fail to accurately characterize complex or multi-modal value distributions [10], creating opportunities for the optimizer to diverge into suboptimal execution paths even when query structure remains unchanged [11].

Dynamic SQL generation contributes to variability in execution context. Modern reporting dashboards increasingly incorporate guided query-building and low-code abstractions, allowing users to construct filters and groupings interactively. Such variability leads to fluctuations in query shapes and predicate patterns, influencing optimizer routing and potentially triggering unanticipated plan choices [12]. When automated data transformation layers normalize or remap source data streams into analytic structures, execution plans can shift faster than statistics refresh cycles can capture [13, 14].

Recent evaluations of Oracle platforms highlight improvements in adaptive plan stability, particularly through runtime feedback mechanisms that refine future plan selection [15]. Meanwhile, low-code and citizen-development tooling has increased the diversity of query patterns reaching the optimizer, making stability controls increasingly important in production environments [16, 17]. Research on autonomous and policy-aware database management suggests that plan stability benefits from combining learned performance profiles with structured governance constraints [18].

Beyond database internals, hybrid workload stability is also influenced by upstream data engineering behavior. ETL orchestration frameworks, batch–stream unification models, and metadata-driven transformation engines shape data arrival patterns that indirectly affect optimizer behavior [19, 20], [21]. In regulated enterprise environments, compliance workflows and reconciliation pipelines further constrain execution predictability, reinforcing the need for stable access paths [22].

Empirical studies across enterprise systems confirm that performance stability cannot be evaluated solely at the query level but must account for workload evolution, user interaction diversity, and operational governance [23, 24]. Comparative analyses across cloud-managed and self-hosted platforms further indicate that plan drift is a systemic property of adaptive optimization under mixed workloads rather than an isolated tuning deficiency [25, 26]. Understanding these drivers is therefore essential for sustaining predictable performance in hybrid OLTP–reporting Oracle environments.

2. Methodology

The methodology for analyzing cost-based optimizer plan drift across Oracle 19c and 23c engines in hybrid OLTP-reporting environments was structured around controlled workload reproduction, execution plan capture, and comparative stability evaluation. The primary goal was to observe how execution plans evolve under fluctuating data distributions, mixed access patterns, and dynamic session contexts common to applications that combine transactional input forms and interactive analytic reports. To ensure that drift was attributable to optimizer behavior rather than environmental noise, a consistent hardware baseline, identical schema layout, and synchronized statistics collection procedures were maintained throughout the evaluation.

The first phase consisted of constructing a representative hybrid workload that mimicked realistic enterprise usage. Transactional operations involved high-frequency inserts and updates on core operational tables, while reporting operations involved periodically executed aggregation, filtering, and join-heavy queries derived from actual dashboard designs. Workload scheduling was varied to simulate different business cycle intensities, including peak concurrent user periods and analytic refresh intervals. This ensured that workload shifts could be examined as drivers of plan changes rather than isolated query performance fluctuations.

The second phase involved capturing baseline execution plans under stable conditions. Both 19c and 23c environments were configured with identical query cache, cursor sharing, and adaptive optimization settings. Execution plans were extracted using standard diagnostics views and persisted for comparison. The plans captured at this stage served as ground truth for identifying future drift. Care was taken to ensure that bind variable usage, optimizer environment settings, and session parameters were identical, preventing artificial divergence.

The third phase introduced controlled data distribution changes. Transaction tables were allowed to grow in stages, and aggregation tables were refreshed at different intervals to reflect operational data evolution. Histogram granularity, statistics refresh frequency, and incremental statistics gathering thresholds were varied systematically. By modifying these conditions gradually rather than abruptly, it was possible to isolate the exact triggers that caused plan recalculation and determine whether the optimizer favored hash joins, nested loops, or full table scans as distributions shifted.

The fourth phase examined dynamic SQL generation behavior. Reporting interfaces were used to construct parameterized queries with varying predicates and sorting conditions. These generated query shapes differed structurally even when targeting the same data, providing insight into how the optimizer resolved join ordering and index selection when encountering semantically similar but syntactically distinct SQL patterns. Special attention was given to filter selectivity patterns that commonly cause misestimation in both 19c and 23c environments.

The fifth phase introduced session-level workload variability. Concurrent users were simulated through load-generation processes to evaluate how the engines adjusted plans when concurrency increased or resource competition emerged. Wait event monitoring and buffer pool pressure were tracked to determine whether plan drift correlated with resource contention, particularly in cases where reporting queries competed with transactional workloads for cache residency and memory allocation.

The sixth phase analyzed adaptive plan stability mechanisms available in 23c. Runtime feedback features were enabled to observe whether repeated query executions converged toward stable plans or oscillated between multiple execution strategies. The comparison with 19c provided insight into whether feedback-driven convergence reduced drift or simply altered the conditions under which drift manifested. Observing stabilization patterns allowed evaluation of the practical effectiveness of 23c's optimizer enhancements.

The seventh phase implemented incremental plan stabilization strategies, including SQL plan baselines, outlines, and fixed adaptive configurations. These measures were applied selectively to determine whether stabilization improved performance consistency without degrading flexibility. The objective was to identify where plan locking introduced unnecessary rigidity and where selective stabilization prevented harmful drift without interfering with adaptive tuning.

The eighth phase evaluated performance continuity. Execution times, buffer reads, CPU consumption, and I/O patterns were monitored across repeated runs of representative queries before and after drift events. Sudden performance spikes were recorded to identify threshold conditions under which execution strategies changed. Performance data was analyzed not only at steady-state but across shifting workload intensities to detect systemic fragility.

The final phase consolidated observations to form a comprehensive view of how cost-based plan drift emerges, stabilizes, or escalates under hybrid workload conditions. The analysis emphasized interpretability, focusing on how workload transitions, statistics freshness, query variability, and adaptive optimization interact to influence plan selection stability. These insights formed the basis for the recommendations and conclusions presented in subsequent sections.

3. Results and Discussion

The analysis revealed that cost-based optimizer plan drift was significantly more pronounced in hybrid OLTP-reporting environments than in purely transactional or purely analytical systems. In Oracle 19c, execution plan selection showed high sensitivity to changes in table cardinalities and session-level predicate selectivity, particularly when workloads shifted abruptly between data entry bursts and reporting refresh intervals. When transactional inserts altered the distribution of values within frequently filtered columns, the optimizer's cardinality estimates diverged from actual runtime conditions, leading to unexpected transitions from index-driven plans to full scan or hash join plans. These transitions typically emerged without changes to query text, indicating that the drift behavior was tied to internal optimizer model adjustments rather than user-driven query variability.

Oracle 23c demonstrated improved stability in plan continuity under moderate workload changes due to refined runtime feedback and more deterministic re-optimization patterns. In repeated execution cycles, 23c converged toward stable plan selections more quickly than 19c when runtime statistics deviated from stored metadata. However, the stabilization benefit was less pronounced during periods of rapid hybrid workload fluctuation, such as when reporting queries and transactional inserts occurred simultaneously. In these cases, plan convergence occasionally oscillated between join strategies across successive executions, indicating that 23c reduces but does not eliminate drift under concurrent pressure.

Dynamic SQL generation played a measurable role in triggering drift in both 19c and 23c. When reporting interfaces allowed users to apply varying filter conditions, the optimizer interpreted similar queries as distinct execution contexts. This led to the creation of multiple child cursors, each with different selectivity assumptions. In 19c, this frequently produced divergent access plans, while in 23c, the divergence remained present but generally converged faster toward a dominant plan after repeated execution. The behavior suggests that improved cursor feedback in 23c narrows performance instability windows but still requires structural control to prevent drift from accumulating under varied user interaction patterns.

Data growth and aggregation refresh cycles produced another key source of plan shift. When operational tables expanded during continuous use, threshold effects were observed in both versions where small changes in table size triggered large plan reorganizations. These shifts occurred when cardinality thresholds influenced whether nested loops or hash joins were selected. In 23c, threshold transitions occurred fewer times and more predictably, but the magnitude of performance difference between the two execution strategies remained significant. This highlights that even improved plan stability does not fully mitigate the cost impact of misalignment between workload and optimizer assumptions.

Concurrency-sensitive drift emerged most clearly when reporting queries coincided with transactional bursts. In 19c, shared buffer contention and latch pressure caused execution plans to skew toward paths that attempted to reduce I/O contention but inadvertently increased logical read volume. In 23c, memory and buffer management improvements reduced the severity of these shifts, but drift still occurred when system resource availability fluctuated quickly. The optimizer's attempts at adaptive balancing under these conditions occasionally led to oscillation between scan-heavy and index-driven strategies until workload conditions stabilized.

Overall, the results indicate that Oracle 23c offers meaningful improvements in execution plan stability over 19c, particularly in environments where query execution patterns repeat frequently. However, stability gains do not fully address the core drift mechanisms associated with non-stationary data distributions, dynamic SQL generation, and concurrency-induced resource variability. Effective drift mitigation still requires intentional workload design, explicit stabilization controls, and monitoring mechanisms that detect transition points before performance degradation becomes operationally visible.

4. Conclusion

The findings of this study show that execution plan drift in Oracle 19c and 23c primarily arises from the interaction between evolving data characteristics, dynamic SQL usage, and shifting workload concurrency levels in hybrid OLTP-reporting systems. While Oracle 23c introduces improved runtime feedback and plan convergence behaviors that reduce short-term instability, the core drivers of drift remain inherent to environments where transactional updates and analytic queries operate on the same dataset. This means that even enhanced adaptive optimization cannot fully prevent plan changes when underlying distributions or access patterns shift during live operations.

Sustained performance consistency in such environments therefore requires a strategy that goes beyond reliance on optimizer adaptability. Selective plan stabilization measures, predictable query shape design, controlled transformation pipelines, and deliberate statistics management practices are necessary to minimize the frequency and operational impact of drift events. The evaluation confirms that targeted stabilization rather than widespread plan locking provides the best balance between reliability and flexibility, preserving the optimizer's ability to respond to legitimate workload evolution while preventing regressive execution shifts.

Overall, achieving execution plan stability in hybrid workload environments is a deliberate design activity rather than an automatic feature of database engines. Oracle 23c offers a stronger foundation for stability than 19c, but consistent performance ultimately depends on aligning system architecture, workload patterns, application logic, and optimizer control policies. The insights from this analysis reinforce the importance of understanding the mechanisms that cause plan drift and applying structured mitigation to maintain predictable performance in enterprise systems.

References

1. Keshireddy, S. R., & Kavuluri, H. V. R. (2019). Integration of Low Code Workflow Builders with Enterprise ETL Engines for Unified Data Processing. *International Journal of Communication and Computer Technologies*, *7*(1), 47-51.
2. Keshireddy, S. R., & Kavuluri, H. V. R. (2019). Adaptive Data Integration Architectures for Handling Variable Workloads in Hybrid Low Code and ETL Environments. *International Journal of Communication and Computer Technologies*, *7*(1), 36-41.
3. Keshireddy, S. R., & Kavuluri, H. V. R. (2020). Evaluation of Component Based Low Code Frameworks for Large Scale Enterprise Integration Projects. *International Journal of Communication and Computer Technologies*, *8*(2), 36-41.
4. Doustjalali, S. R., Gujjar, K. R., Sharma, R., & Shafiei-Sabet, N. (2016). Correlation between body mass index (BMI) and waist to hip ratio (WHR) among undergraduate students. *Pakistan Journal of Nutrition*, *15*(7), 618-624.
5. Ahmed, J., Mathialagan, A. G., & Hasan, N. (2020). Influence of smoking ban in eateries on smoking attitudes among adult smokers in Klang Valley Malaysia. *Malaysian Journal of Public Health Medicine*, *20*(1), 1-8.
6. Keshireddy, S. R. (2021). Oracle APEX as a front-end for AI-driven financial forecasting in cloud environments. *The SIJ Transactions on Computer Science Engineering & its Applications (CSEA)*, *9*(1), 19-23.
7. Keshireddy, S. R., & Kavuluri, H. V. R. (2021). Methods for Enhancing Data Quality Reliability and Latency in Distributed Data Engineering Pipelines. *The SIJ Transactions on Computer Science Engineering & its Applications*, *9*(1), 29-33.

8. Haque, A. H. A. S. A. N. U. L., Anwar, N. A. I. L. A., Kabir, S. M. H., Yasmin, F. A. R. Z. A. N. A., Tarofder, A. K., & MHM, N. (2020). Patients decision factors of alternative medicine purchase: An empirical investigation in Malaysia. *International Journal of Pharmaceutical Research*, *12*(3), 614-622.
9. Nazmul, M. H. M., Fazlul, M. K. K., Rashid, S. S., Doustjalali, S. R., Yasmin, F., Al-Jashamy, K., ... & Sabet, N. S. (2017). ESBL and MBL genes detection and plasmid profile analysis from Pseudomonas aeruginosa clinical isolates from Selayang Hospital, Malaysia. *PAKISTAN JOURNAL OF MEDICAL & HEALTH SCIENCES*, *11*(3), 815-818.
10. MKK, F., MA, R., Rashid, S. S., & MHM, N. (2019). Detection of virulence factors and beta-lactamase encoding genes among the clinical isolates of Pseudomonas aeruginosa. *arXiv preprint arXiv:1902.02014*.
11. Nazmul, M. H. M., Salmah, I., Jamal, H., & Ansary, A. (2007). Detection and molecular characterization of verotoxin gene in non-O157 diarrheagenic Escherichia coli isolated from Miri hospital, Sarawak, Malaysia. *Biomedical Research*, *18*(1), 39-43.
12. Keshireddy, S. R., & Kavuluri, H. V. R. (2020). Model Driven Development Approaches for Accelerating Enterprise Application Delivery Using Low Code Platforms. *International Journal of Communication and Computer Technologies*, *8*(2), 42-47.
13. Keshireddy, S. R., Kavuluri, H. V. R., Mandapatti, J. K., Jagadabhi, N., & Gorumutchu, M. R. (2022). Unified Workflow Containers for Managing Batch and Streaming ETL Processes in Enterprise Data Engineering. *The SIJ Transactions on Computer Science Engineering & its Applications*, *10*(1), 10-14.
14. Keshireddy, S. R., Kavuluri, H. V. R., Mandapatti, J. K., Jagadabhi, N., & Gorumutchu, M. R. (2022). Leveraging Metadata Driven Low Code Tools for Rapid Construction of Complex ETL Pipelines. *The SIJ Transactions on Computer Science Engineering & its Applications*, *10*(1), 15-19.
15. Keshireddy, S. R. (2022). Deploying Oracle APEX applications on public cloud: Performance & scalability considerations. *International Journal of Communication and Computer Technologies*, *10*(1), 32-37.
16. Keshireddy, S. R., & Kavuluri, H. V. R. (2021). Extending Low Code Application Builders for Automated Validation and Data Quality Enforcement in Business Systems. *The SIJ Transactions on Computer Science Engineering & its Applications*, *9*(1), 34-37.
17. Keshireddy, S. R., & Kavuluri, H. V. R. (2021). Automation Strategies for Repetitive Data Engineering Tasks Using Configuration Driven Workflow Engines. *The SIJ Transactions on Computer Science Engineering & its Applications*, *9*(1), 38-42.
18. KESHIREDDY, S. R. (2023). Blockchain-Based Reconciliation and Financial Compliance Framework for SAP S/4HANA in MultiStakeholder Supply Chains. *Akıllı Sistemler ve Uygulamaları Dergisi*, *6*(1), 1-12.
19. Keshireddy, S. R., Kavuluri, H. V. R., Mandapatti, J. K., Jagadabhi, N., & Gorumutchu, M. R. (2023). Enhancing Enterprise Data Pipelines Through Rule Based Low Code Transformation Engines. *The SIJ Transactions on Computer Science Engineering & its Applications*, *11*(1), 60-64.
20. Keshireddy, S. R., Kavuluri, H. V. R., Mandapatti, J. K., Jagadabhi, N., & Gorumutchu, M. R. (2023). Optimizing Extraction Transformation and Loading Pipelines for Near Real Time Analytical Processing. *The SIJ Transactions on Computer Science Engineering & its Applications*, *11*(1), 56-59.
21. Subramaniyan, V., Fuloria, S., Sekar, M., Shanmugavelu, S., Vijeepallam, K., Kumari, U., ... & Fuloria, N. K. (2023). Introduction to lung disease. In *Targeting Epigenetics in Inflammatory Lung Diseases* (pp. 1-16). Singapore: Springer Nature Singapore.
22. Jamal Hussaini, N. M., Abdullah, M. A., & Ismail, S. (2011). Recombinant Clone ABA392 protects laboratory animals from Pasteurella multocida Serotype B. *African Journal of Microbiology Research*, *5*(18), 2596-2599.

23. Hussaini, J., Nazmul, M. H. M., Masyitah, N., Abdullah, M. A., & Ismail, S. (2013). Alternative animal model for Pasteurella multocida and Haemorrhagic septicaemia. *Biomedical Research*, *24*(2), 263-266.
24. Arzuman, H., Maziz, M. N. H., Elsersi, M. M., Islam, M. N., Kumar, S. S., Jainuri, M. D. B. M., & Khan, S. A. (2017). Preclinical medical students perception about their educational environment based on DREEM at a Private University, Malaysia. *Bangladesh Journal of Medical Science*, *16*(4), 496-504.
25. KESHIREDDY, Srikanth Reddy. "Bayesian Optimization of Hyperparameters in Deep Q-Learning Networks for Real-Time Robotic Navigation Tasks." *Akıllı Sistemler ve Uygulamaları Dergisi* 6.1 (2023): 1-12.
26. Keshireddy, S. R., & Kavuluri, H. V. R. (2022). Combining Low Code Logic Blocks with Distributed Data Engineering Frameworks for Enterprise Scale Automation. *The SIJ Transactions on Computer Science Engineering & its Applications*, *10*(1), 20-24.