ISSN: 2949-8325
Vol 2, Issue 2, 2023

Adaptive Cursor Sharing Dynamics Under Parameter Skew in Oracle SQL Engines

Dr. Daniel J. Mercer & Dr. Mark A. Hollinger

Abstract

Parameter skew in Oracle SQL workloads can cause significant performance variability when a single shared execution plan is applied to bind values that produce very different data selectivity conditions. This article investigates the behavior of Adaptive Cursor Sharing (ACS) in Oracle Database 19c using a trace-based execution analysis framework, examining how cursor states evolve from bind-sensitive to bind-aware as runtime feedback accumulates. The results show that ACS effectively stabilizes performance when skew is pronounced, generating specialized child cursors tailored to distinct value distributions. However, in cases of moderate skew, ACS adaptation occurs only after multiple suboptimal executions, indicating a detectable lag between performance deviation and optimizer response. The study concludes that ACS provides reliable long-term stability but may require complementary tuning strategies to accelerate plan differentiation in latency-sensitive or rapidly shifting workloads.

Keywords: Adaptive Cursor Sharing, Parameter Skew, Oracle SQL Execution

1. Introduction

Cursor sharing is a core component of SQL execution efficiency in Oracle Database systems, allowing identical SQL statements to reuse previously parsed execution plans rather than undergoing repeated hard parsing [1,2]. In typical transactional and analytical applications, most SQL statements are parameterized, and the Oracle engine substitutes runtime bind values during execution. However, not all bind values behave equivalently in terms of data selectivity, and their effect on cardinality and access path decisions can differ significantly. Prior empirical observations have shown that such discrepancies can introduce execution instability when a single plan is reused across heterogeneous runtime conditions [3,4]. Application-tier query frameworks, particularly low-code and metadata-driven development environments, further increase the prevalence of parameterized execution patterns in modern enterprise deployments [5,6].

The root of this challenge lies in parameter skew, where different values supplied to the same bind variable correspond to highly uneven data distributions [7]. Although the Oracle Cost-Based Optimizer relies on statistical constructs such as histograms to estimate selectivity, bind-variable execution can limit the optimizer's ability to exploit these statistics effectively, especially under bind peeking and bind-sensitive execution scenarios [8,9]. In distributed and cloud-oriented database environments, elastic scaling and session multiplexing amplify these effects as workloads are executed across heterogeneous pools [10]. Additional variability is introduced when machine-learning-driven analytics and predictive workflows are integrated through APEX-facing interfaces, subtly influencing cursor reuse and execution path stability [11].

To mitigate skew-driven instability, Oracle introduced Adaptive Cursor Sharing (ACS), which monitors execution feedback and generates alternative child cursors when divergent selectivity patterns are detected [12]. ACS attempts to balance plan reuse with performance isolation by reacting

to observed execution statistics rather than static optimizer assumptions. However, because this mechanism is feedback-driven, it may respond only after suboptimal execution has already occurred [13]. Differences in deployment strategy, including concurrency models and execution isolation between on-premise and public cloud environments, further affect when and how ACS is triggered [14,15].

Beyond data distribution effects, cursor performance is shaped by broader schema and workload design factors such as index selectivity, predicate stability, join topology, and query formulation practices. Practical tuning methodologies therefore emphasize evaluating execution behavior under realistic workload conditions rather than relying exclusively on optimizer theory [16]. Low-code and metadata-driven application development practices introduce additional abstraction layers that can obscure query-generation patterns, complicating diagnosis of cursor reuse behavior [17,18]. This necessitates empirical performance analysis approaches that explicitly connect observed execution outcomes with underlying structural and statistical influences.

Security and governance considerations add further complexity to cursor-sharing behavior. Role-based access models and privilege segmentation can unintentionally partition value ranges accessed by different user groups, resulting in divergent selectivity profiles for identical SQL statements [19]. Deployment of Oracle APEX applications across multi-tenant or multi-region cloud environments compounds this diversity, making cursor performance highly environment-dependent [20]. Consequently, recent performance studies emphasize trace-based and observational evaluation of shared pool behavior and optimizer feedback cycles as essential for understanding plan stability under real-world conditions [21].

2. Methodology

This study follows a trace-based observational approach to analyze how Adaptive Cursor Sharing (ACS) responds to parameter skew during repeated execution of parameterized SQL statements. The methodology is designed to reveal cursor branching, selectivity feedback, and plan variation behavior as they occur inside the Oracle SQL execution engine, rather than relying solely on static optimizer predictions. All experiments were conducted on Oracle Database 19c, configured with cost-based optimization, dynamic sampling enabled, and bind variable usage consistent with typical OLTP-style query patterns [1].

A base test relation containing approximately 10 million rows was constructed with intentionally skewed value frequency on the filter column. One value range represented high-frequency (common) occurrences, while another represented low-frequency (rare) occurrences, simulating real-world skew conditions often observed in transactional systems [4]. A single parameterized SQL statement of the form:

SELECT <columns> FROM test_table WHERE filter_col = :b1;

was used as the primary workload driver. The statement was executed repeatedly with alternating bind values, first favoring the common value and then shifting to rare values, in order to test when and how ACS altered cursor behavior [5].

To monitor cursor evolution, the v$sql and v$sql_shared_cursor views were queried after controlled execution cycles. The child_number, executions, and parse_call counters from *v$sql* were tracked to detect cursor reuse versus new cursor generation. The appearance of additional child cursors indicated that Oracle had transitioned from bind-sensitive to bind-aware execution behavior. The BIND_SENSITIVE and BIND_AWARE flags in *v$sql_shared_cursor* were examined to verify that

cursor adaptation was triggered due to selectivity-related feedback, rather than metadata or environment mismatches [8].

Execution plan differences were evaluated using DBMS_XPLAN.DISPLAY_CURSOR, allowing side-by-side comparison of access path selections across different parameter values. These comparisons focused on index range scan, index skip scan, bitmap index scan, and full table scan choices. Previous literature highlights that access path variation is one of the clearest indicators of skew-related plan divergence [5], and observing this variation provided direct evidence of ACS adaptation in runtime conditions.

To ensure that cursor adaptation was driven by data distribution rather than session environment, v$ses_optimizer_env was monitored to verify stability of optimizer-related parameters throughout the experiments. In addition, database statistics, including histograms on the skewed filter column, were locked during execution to maintain consistent cardinality estimation inputs. This eliminated optimizer variability unrelated to parameter skew, aligning with performance investigation practices recommended in real deployment contexts [6].

Workload replay was performed in iterative batches of 100–1,000 executions per bind value segment. Execution performance metrics, including elapsed time, buffer gets, and CPU time, were gathered from *v$sql* runtime statistics. These observations were correlated with cursor state transitions to determine whether cursor adaptation occurred early enough to prevent performance degradation or only after multiple inefficient executions had already occurred a phenomenon acknowledged in practitioner experience but less commonly documented in formal evaluation [11].

Application-layer query generation patterns were also considered, since APEX-based and low-code environments may introduce dynamic parameter variability depending on user interaction flows [3]. Observing cursor adaptation behavior under these conditions helped assess whether ACS activation timing aligns with the performance realities of multi-session user workloads rather than single-threaded synthetic benchmarks [12].

By combining cursor state monitoring, plan inspection, performance metric tracking, and workload pattern analysis, this methodology captures how Oracle's ACS mechanism responds to skew-driven performance instability in a live execution environment. The approach provides a grounded basis for evaluating ACS effectiveness and timing, supporting deeper insight into the practical scenarios where adaptive cursor sharing succeeds, lags, or fails to activate in ways that materially affect workload performance [16].

3. Results and Discussion

The trace-based execution results demonstrated a clear distinction in how the Oracle SQL engine handled parameter skew when adaptive cursor sharing was enabled. During the initial execution cycles, the optimizer consistently reused a single cursor plan regardless of the bind variable input. When the common high-frequency parameter value was used, the chosen plan performed efficiently, favoring index-based access paths with low I/O overhead. However, when the low-frequency parameter value was supplied, the same plan produced significantly higher logical I/O and longer execution times due to an unsuitable index range filter applied to a very sparse data region. This mismatch led to inefficient buffer access and noticeable delay before the system initiated adaptive cursor sharing behavior.

As executions continued, the database began to recognize the skew-driven performance inconsistency, transitioning the cursor from bind-sensitive to bind-aware status. This transition was characterized by the creation of additional child cursors associated with the same SQL statement but optimized for

different selectivity conditions. The new child cursor selected an access path more suitable for the rare parameter value, often choosing full table scan or filtered index scan strategies that offered better runtime efficiency relative to the input distribution. The timing of this shift was critical; adaptive cursor sharing did not activate immediately but required multiple execution cycles before sufficient feedback accumulated to classify the bind pattern as performance-relevant.

Performance measurements before and after cursor branching highlighted the practical impact of adaptive cursor sharing. Once the system recognized the skew and introduced specialized child cursors, execution times stabilized across parameter variations. The common parameter value continued to use the original index-based cursor variant, while the rare value shifted to the new plan suited to its lower selectivity. This alignment reduced the performance gap that had previously existed between efficient and inefficient execution paths. The improvement demonstrated the intended purpose of adaptive cursor sharing: enabling plan diversity only when it materially benefits performance, rather than encouraging unnecessary cursor proliferation.

However, the analysis also revealed scenarios where adaptive cursor sharing did not activate soon enough to prevent avoidable performance degradation. In cases where the performance difference between common and rare parameter values was moderate rather than extreme, the optimizer required additional execution feedback to classify the cursor as bind-aware. During this delay phase, the system continued to apply the suboptimal plan, resulting in unnecessary overhead for a portion of the workload. This lag suggests that while ACS effectively adapts to clear cases of skew, borderline selectivity shifts may require supplemental intervention, such as targeted statistics improvements or manual optimizer directives, to accelerate detection.

Overall, the results confirm that adaptive cursor sharing improves runtime consistency in environments affected by parameter skew, but its effectiveness depends heavily on the degree of skew, the stability of execution patterns, and the speed at which selectivity feedback accumulates. The mechanism performs best when skew is pronounced and execution frequency is sufficiently high to allow the optimizer to detect and respond. In moderate or variable skew environments, adaptive cursor sharing may take longer to respond, during which time performance volatility remains possible. These findings underscore the importance of observing cursor evolution under realistic workload repetitions and highlight opportunities for fine-tuning ACS responsiveness in environments where skew-induced behavior is operationally significant.

4. Conclusion

This study examined how Oracle's Adaptive Cursor Sharing mechanism responds to parameter skew during repeated execution of parameterized SQL statements. By observing real cursor evolution, rather than relying solely on static optimizer predictions, the analysis revealed how bind-sensitive behavior transitions into bind-aware plan diversification as execution feedback accumulates. The findings confirm that ACS can effectively mitigate performance instability by generating cursor variants suited to distinct selectivity conditions, ensuring more consistent response times across parameter values that differ significantly in data frequency.

However, the results also highlight a fundamental timing limitation in ACS behavior. The optimizer requires several execution cycles before recognizing performance divergence, meaning suboptimal plans may persist during early stages of workload execution. This effect is most noticeable when skew is moderate rather than extreme, where plan inefficiency is present but not sufficiently dramatic to trigger early cursor adaptation. As a result, systems with rapidly shifting workloads or latency-sensitive operations may experience short-term inefficiencies before plan specialization is fully established.

These observations suggest that while adaptive cursor sharing is an effective mechanism for handling skew-driven workload instability, it performs optimally when supported by informed schema design, stable statistics, and repeatable workload patterns. In environments where workload variability is high or selectivity boundaries are subtle, complementary tuning strategies such as histogram refinement, targeted optimizer hints, or adaptive sampling policies may help accelerate cursor adaptation and avoid transitional performance impacts. Overall, understanding how ACS activates and evolves allows database practitioners to better align optimizer behavior with real operational workload characteristics, improving the predictability and efficiency of SQL execution under parameter skew.

References

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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*.
10. 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.
11. 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.
12. 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.

13. 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.
14. 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.
15. 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.
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. (2022). Deploying Oracle APEX applications on public cloud: Performance & scalability considerations. *International Journal of Communication and Computer Technologies*, *10*(1), 32-37.
19. 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.
20. 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.
21. 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.