Concurrency Pattern Analysis in Multi-User APEX Data Entry Scenarios

Ethan Marwood, Clara Donnelly

Abstract

Multi-user data entry workflows in Oracle APEX–based manufacturing dashboards introduce concurrency challenges when operators simultaneously update shared equipment logs while automated sensor streams also feed machine-state data into the system. These overlapping write operations can lead to silent overwrites, inconsistent machine histories, and misaligned production records if concurrency is not explicitly managed at both the application and data layers. This study analyzes concurrency interaction patterns arising from operator input timing, UI refresh behavior, and background telemetry synchronization. Results show that uniform locking strategies are often inefficient in production settings, while selective techniques such as row-level version stamping and shift-based data partitioning provide reliable conflict mitigation with minimal workflow disruption. The findings emphasize that effective concurrency control in APEX manufacturing systems requires integrated workflow, interface, and data model design rather than database-level enforcement alone.

Keywords: Oracle APEX, Concurrency Control, Manufacturing Data Entry

1. Introduction

Manufacturing environments increasingly rely on operator-driven data entry combined with automated machine telemetry to maintain accurate production records and equipment history logs. In Oracle APEX–based monitoring dashboards, multiple operators frequently modify shared tables related to equipment status, downtime events, and production shift annotations, creating risks of write collisions and silent overwrites when concurrency is not explicitly managed [1,2]. Cloud-backed Oracle deployments further complicate concurrency behavior due to commit latency, asynchronous propagation, and session multiplexing, requiring strategies that extend beyond basic transactional isolation [3,4].

In industrial settings where IoT telemetry streams continuously, manual operator input is often used to supplement, override, or contextualize sensor-generated data. This hybrid human–machine update pattern substantially increases the likelihood of write conflicts, particularly when updates target the same production unit or asset identifier [5]. Studies on burst-driven and causally complex data streams demonstrate that irregular arrival patterns lead to overlapping transaction windows, stressing concurrency control mechanisms [6,7]. Importantly, tuning database performance alone does not eliminate these risks, as many conflicts originate from interaction timing at the application workflow layer rather than from storage-level contention [8].

Oracle APEX user workflows frequently involve multi-step forms, conditional navigation, and asynchronous page regions, meaning that user interactions do not map to database writes in a strictly linear order. When workflow execution is decoupled through background job queues or dynamic actions, concurrent updates may reach the commit layer in non-deterministic sequences [9,10]. In distributed manufacturing systems that rely on multi-region replication for high availability, propagation delay further affects update visibility, increasing the probability that two operators unknowingly modify

overlapping data values [11]. Temporal misalignment in IoT monitoring systems introduces additional uncertainty in write ordering, compounding concurrency risk [12].

Concurrency conflicts in such environments do not always surface as explicit lock errors. Instead, in APEX dashboards, conflicts may appear as stale form values, overwritten log entries, or visually consistent but historically inaccurate equipment timelines [13]. Interface-level interaction layers can unintentionally mask these anomalies; for example, auto-suggestion and assisted form behavior may smooth user experience while allowing underlying write conflicts to propagate undetected [14]. Effective concurrency design therefore requires coordination between backend transactional controls and frontend mechanisms that expose conflict conditions to users.

Access control models in APEX restrict which operators may modify specific production units, but authorization alone cannot prevent concurrency issues when multiple users with similar privileges record events simultaneously [15]. Automated data transformation and workflow orchestration pipelines may further obscure update provenance, complicating audit and traceability efforts [16,17]. In manufacturing environments governed by ISO standards and quality assurance frameworks, the inability to reconstruct accurate, operator-specific modification histories represents a significant compliance risk [18].

Transaction processing theory has long emphasized the need to balance consistency guarantees with throughput efficiency under concurrent access [19]. Empirical research on multi-user DBMS architectures shows that concurrency strategies must be tailored to workload composition and update collision frequency [20]. Recent reviews of concurrency control in cloud-native and hybrid transactional systems indicate that concurrency risk is especially pronounced in operator-driven workflows where manual edits intersect with autonomous system updates [21]. Accordingly, analyzing concurrency behavior in multi-user APEX data entry environments is essential for preserving production log integrity and preventing silent data corruption in industrial monitoring systems.

2. Methodology

The methodology for analyzing concurrency patterns in multi-user APEX manufacturing data entry environments was designed to capture real-world operator behavior, system signaling delays, and commit interaction effects across shared database records. The approach emphasized controlled workload simulation, structured observation of locking and update events, and trace reconstruction of overlapping write operations. To achieve this, the study environment was configured to resemble a typical production-floor logging dashboard, where operators record machine status, downtime reasons, production counts, and shift notes while automated telemetry streams continuously feed machine sensor data into staging tables.

The first step involved constructing a representative APEX application that supports concurrent data entry into shared equipment log records. The application included form-based update screens, interactive report views, real-time KPIs, and background refresh processes. The layout reflected a common industrial workflow where multiple operators interact with the same set of machine-level data at overlapping time intervals. Fields were structured to expose both timestamped and descriptive attributes to allow conflict tracing at a granular level.

A dataset of simulated machine states and production conditions was generated to emulate real manufacturing variability. These simulated entries incorporated periodic status shifts, fault conditions, and recovery transitions to stimulate operator interventions. Sensor streams were introduced through a timed process that inserted or updated readings at configurable intervals. Manual operator entries were then layered on top of this automated feed to analyze how human and system-generated events interact when writing to the same tables.

Concurrent user sessions were simulated using staged user accounts representing multiple operators working in parallel across different workstations. Each simulated operator executed data-entry tasks, including inserting new logs, modifying existing records, and acknowledging machine state changes. The simulation model allowed control over the speed, overlap, and intensity of editing operations, enabling observation of conflict density under varied workload conditions.

A monitoring mechanism was added to record event timestamps, lock acquisition attempts, commit ordering, and the presence of lost updates. This mechanism captured not only explicit locking behavior but also silent write overwrites and last-commit-wins conflicts, which do not surface through error messages yet result in data inconsistency. The logging layer also recorded each operator's viewport state at the moment of update submission, allowing the reconstruction of cases where users acted on stale visible data.

The study also evaluated how UI refresh timing influenced concurrency visibility. Automatic refresh intervals were varied to test whether frequent screen updates helped prevent operators from working on outdated values or instead created additional contention through repeated refresh-triggered data reads. Manual refresh triggers were observed to determine when operators attempted to resynchronize their display with the live state and whether these attempts resolved or reinforced conflicts.

Conflict patterns were then classified based on their structural signatures. Patterns included simultaneous edit collisions on the same row, temporal misalignment where one operator edited values before another's update became visible, and cascading overwrites where a chain of dependent fields lost synchronization. Categorizing these patterns allowed systematic analysis of conditions that amplify concurrency risk.

Finally, the results were evaluated to determine which application design strategies reduce conflict probability. These included row-level version stamping, explicit edit warnings, form-level record locking, conditional commit logic, and data partitioning based on shift or machine assignments. Each mitigation was tested under equivalent workload conditions to assess its effect on data integrity, operator coordination, and workflow continuity.

3. Results and Discussion

The analysis revealed that concurrency issues in multi-user APEX manufacturing dashboards tend to cluster around equipment records with high operator interaction frequency and rapid state changes. When multiple operators attempted to update the same machine log within short intervals, the default last-write-wins commit model resulted in silent overwrites. These overwrites were especially common in fields capturing downtime reasons, corrective actions, and operator notes areas where human interpretation varies and entries are rarely identical. The absence of visible conflict indicators allowed updates to appear valid in the UI even when previous entries had been overwritten.

UI refresh intervals played a critical role in shaping user perception of system state. Operators working with slower or manual refresh patterns often acted on stale screen data, submitting updates that were correct relative to what they saw, but incorrect relative to the actual current machine state. Increasing refresh frequency reduced this issue but introduced new contention by triggering more frequent data reads and background state load operations. This demonstrated that refresh rate tuning requires balancing informational accuracy with backend stability.

The study also found distinct concurrency behavior patterns between manual entries and automated telemetry feeds. Machine sensor data streams rarely caused direct write conflicts because they typically updated separate columns or staging tables. However, manual operator corrections inserted after a sensor-driven update frequently overwrote recent automated entries, altering machine history

unintentionally. This indicated that concurrency management must consider data origin (human vs. system-generated), not just record scope.

The effectiveness of mitigation strategies varied. Form-level record locking significantly reduced silent overwrites but introduced workflow delays and manual conflict resolution steps, which operators sometimes bypassed by refreshing or reopening forms. Row-level version stamping, however, provided an unobtrusive mechanism to detect stale edits while allowing operators to resolve conflicts before commit. Partitioning records by shift or production segment reduced contention in systems where machine assignments were consistent across shifts.

Table 1 summarizes the relative performance and behavioral impact of the key concurrency mitigation strategies evaluated. As shown in Table 1, row-level version stamping and shift-based record partitioning provided the best balance between operational flow and data integrity preservation, whereas form-level locking, while effective, introduced friction that reduced workflow efficiency.

Table 1. Comparison of Concurrency Mitigation Strategies

Mitigation Strategy	Data Integrity Improvement	Workflow Disruption	Suitability in High-Activity Scenarios	Notes
Form-Level Record Locking	High	High	Medium	Prevents overwrites but slows operator workflows
Row-Level Version Stamping	High	Low	High	Automatically detects stale updates and prompts resolution
UI Auto-Refresh Timing Optimization	Medium	Low to Medium	Medium	Reduces stale edits but increases read load at higher frequencies
Shift-Based Record Partitioning	Medium to High	Low	High	Divides write domains to reduce contention
Machine-Specific Operator Assignment	Medium	Low	Medium	Effective when staffing assignments are stable

These findings confirm that concurrency challenges in APEX multi-user manufacturing systems are not purely transactional but emerge from the interaction of interface timing, human behavior, and workload structure. Effective solutions require selective control rather than blanket locking or system-wide synchronization.

4. Conclusion

This study demonstrates that concurrency behavior in multi-user APEX manufacturing data entry systems arises from the interaction of human input timing, automated telemetry updates, and application UI refresh dynamics. Traditional transactional isolation alone is insufficient for preventing silent overwrites and operator-driven data conflicts because many of the strongest concurrency risks originate at the user interaction and workflow coordination layers, not solely at the commit layer. Ensuring

reliable machine history and production traceability therefore requires application-level strategies that make concurrency states visible and manageable.

The evaluation showed that selective mitigation approaches provide better outcomes than rigid locking mechanisms. Row-level version stamping and shift-based record partitioning preserved data integrity while maintaining workflow efficiency, making them suitable for high-activity industrial environments. In contrast, strict form-level locking, while effective against overwrites, introduced workflow friction and encouraged workaround behavior. The results highlight the need to balance coordination support with operator usability, rather than relying on blocking-based concurrency control alone.

Overall, concurrency management in APEX manufacturing dashboards must be treated as a workflow system design problem rather than a purely database-level concern. Aligning interface refresh timing, operator scope boundaries, and conflict resolution cues significantly improves multi-user editing reliability. Future work should explore adaptive refresh synchronization and predictive conflict detection, enabling systems to anticipate contention patterns rather than react to them after data has already been overwritten.

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. 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.
6. 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.
7. 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*.
8. 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.
9. 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.

10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
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.