The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

Research Article

Modeling Oracle Performance Degradation in Production: Predictive Analytics Based on Wait Events, I/O, Memory/Sga/Pga, Background Process, And Storage Parameters

Submission Date: July 11, 2022, **Accepted Date:** August 03, 2022,
Published Date: September 28, 2022 |

Olga Badiukova
Oracle DBA ETS, LLC "Ci Ec Tek" Kyiv, Ukraine

Journal Website: https://theamericanjournals.com/index.php/tajet

ABSTRACT

The paper proposes an integrated methodological approach to modeling and forecasting Oracle DBMS performance degradation under high-load industrial environments. Against the backdrop of digital transformation and the consolidation of the AIOps paradigm, there is a shift from reactive monitoring toward proactive intelligent support mechanisms focused on early detection of adverse indicators and incident prevention. A central role is assigned to the integration of multidimensional telemetry: wait events, performance metrics of the input/output (I/O) subsystem, temporal profiles of memory consumption in the SGA and PGA areas, as well as behavioral characteristics of key background processes, including LGWR, DBWR, and ARCH, are taken into account. The proposed methodology for the first time combines classical Oracle diagnostics (AWR, ASH, V$ views) with machine learning and AIOps approaches to predict future degradations before they manifest at the user level.

The architecture of a predictive solution is described, based on a combination of machine learning methods (Random Forest, XGBoost) and deep learning models (LSTM, BERT). This composition ensures robust extraction of latent patterns in operational data streams and enables detection of anomalous states 7–12 minutes before the formation of a critical failure. Substantial attention is devoted to the reconstruction of causal relationships between infrastructural parameters and application-level indicators, which makes it possible to interpret degradation not as a set of disparate symptoms, but as a result of the interaction of resources, internal DBMS mechanisms, and the load profile. Empirical verification on experimental data demonstrates a significant improvement in operational diagnostics, including a MTTD 63%. The material is intended for practicing database operations specialists, designers of high-load solutions, and researchers developing intelligent methods for IT operations.

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

KEYWORDS

Oracle Database, predictive analytics, machine learning, wait events, SGA, PGA, input/output performance, AIOps, background processes, production environments, performance degradation, anomaly detection, root cause analysis

INTRODUCTION

The contemporary landscape of IT infrastructure is characterized by high complexity, within which Oracle Database retains the status of a mature enterprise platform for mission-critical workloads; notably, Oracle Database 19c is positioned by Oracle as its long-term support release [1]. Against the backdrop of ongoing digital transformation, worldwide spending on the digital transformation of business practices, products, and organizations was forecast to reach $1.8 trillion in 2022, which further increases the importance of service continuity and resilience in enterprise IT environments [2].

A decrease in the performance of production Oracle instances, as a rule, is not realized in the form of an instantaneous failure, but through cumulative degradation a decline in throughput accompanied by increasing latency. Subsequently, this may transform into SLA non-compliance and direct financial losses. As of 2022, the economic impact of outages was already substantial: according to Uptime Institute, more than 60% of reported failures resulted in losses of at least $100,000, while 25% of respondents indicated that their most recent major outage cost more than $1 million [3]. The causal structure of such incidents is heterogeneous and includes defects at the application and query levels, as well as infrastructure-related constraints; in addition, greater architectural complexity in distributed and cloud-based resilience models can itself become a source of operational risk [3]. In practice, recurrent contributors include suboptimal SQL execution patterns, I/O contention, memory pressure, and lock or latch contention, all of which are reflected in Oracle wait-event and performance-view diagnostics [5, 13, 14].

The situation is further complicated by the fact that modern distributed and cloud environments generate high-density telemetry arrays in which manual analysis becomes methodologically and organizationally unproductive, since latent dependencies and weak predictors of degradation are practically not amenable to timely identification [4, 6]. Under these conditions, the AIOps (Artificial Intelligence for IT Operations) concept provides tools for automating event analysis, correlating indicators, and constructing predictive failure scenarios [7, 8]. The use of predictive analytics ensures a transition from reactive mitigation of consequences to proactive management of the database state, making it possible to diagnose bottlenecks before they begin to affect user transactions and business-critical services [9, 10].

The objective of the study is to develop and empirically validate an integrated methodological approach to modeling and forecasting Oracle performance degradation in high-load industrial operation based on the integration of multidimensional telemetry (wait events, I/O, SGA/PGA, background processes, storage parameters) and ML/DL methods for early anomaly detection and prevention of incidents/SLA violations. The goal of this work is to develop a formal, reproducible model that includes: collection of Oracle telemetry, normalization and aggregation of indicators, degradation modeling, forecasting of future problems, and interpretation of degradation causes.

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

We hypothesize that joint analysis of wait events, input/output metrics, SGA/PGA memory profiles, and the behavior of background processes (LGWR/DBWR/ARCH), complemented by semantic interpretation of alert.log/trace, enables robust identification of degradation as a result of the interaction of resources and internal DBMS mechanisms and detection of pre-failure states 7–12 minutes before critical escalation, reducing MTTD.

The scientific novelty lies in the fact that, for the first time, the paper proposes an interpretable predictive AIOps architecture for Oracle that combines multichannel telemetry from V views/AWR/ASH/logs; ensemble models (Random Forest, XGBoost) and deep learning models (LSTM) for different types of predictive tasks; an NLP module (BERT) for transforming unstructured log messages into features; and a layer of explainability and causal attribution (SHAP + RCA orchestration), which shifts diagnostics from a set of symptoms to a causal model of degradation. Additionally, a formal vector-based baseline modeling approach and threshold-based degradation detection across five key domains (SQL/Optimizer, Memory SGA/PGA, Background Processes, I/O Subsystem, Storage & Network) are introduced.

Materials and Methods

The materials of the study comprised Oracle operational and diagnostic data extracted from the internal repositories of the DBMS and the operating system: dynamic performance views, memory statistics, historical AWR snapshots for trend analysis, high-frequency ASH data for capturing short-term spikes, as well as unstructured alert.log journals and trace files containing ORA messages and warnings (including signals for ORA-04031 and ORA-00600). At the same time, the results of other studies and industry statistical sources on AIOps/predictive analytics and on the economics of incidents (cost per minute of downtime, annual damage, prevalence of degradation causes such as suboptimal SQL and resource shortages) were analyzed, which established the context and target benchmarks (for example, reducing detection time and preventing SLA violations).

At the core of Oracle performance-degradation formalization lies the canonical response-time model, in which total latency is represented as the sum of "pure" CPU execution time for SQL and the accumulated waits across event classes:

$$\text{Response Time} = \text{CPUTime} + \Sigma\, \text{WaitTime}_i,$$

where CPUTime captures CPU consumption attributable to execution, and WaitTime_i denotes the aggregated waiting time for each event type (disk operations, locks, latches, memory pressure, network delays, and so forth). Within this framing, degradation manifests either as an increase in $\Sigma\, \text{WaitTime}_i$ as a whole, or as a redistribution of the wait structure-cases where the overall magnitude remains comparable, yet different events become dominant.

From a typological standpoint, degradation modes are most usefully separated by leading observability signals. A CPU-bound deterioration corresponds to high processor utilization with a relatively low background of waits. An I/O-bound profile is characterized by growth in read-related waits, primarily db file sequential read and db file scattered read. A memory-constrained regime tends to surface through waits such as free buffer waits and intensified contention at the latch level, notably latch: cache buffers chains. Competitive degradation is defined by enq: TX waits and the symptom complex of row lock contention. Storage-oriented disruptions are often captured via log file sync and write complete waits,

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

while the network component appears as increased latency in SQLNet messages from/to client events.

Degradation modeling is rationally initiated by constructing a "normal-state" reference. For each snapshot, a baseline dynamics vector may be formed as:

$$\text{Vector} = [CPU_t, IO_t, MEM_t, WAIT_t, LATCH_t, BGPROC_t],$$

where CPU_t denotes CPU utilization (%), IO_t the average I/O latency (ms), MEM_t the buffer cache hit rate (%), $WAIT_t$ total wait time in seconds, $LATCH_t$ latch contention intensity, and $BGPROC_t$ the share of background-process waits. This reference defines the expectation and dispersion of observed metrics, enabling a shift from descriptive monitoring to statistically testable anomaly criteria.

Detection of deterioration for a specific wait event i can be expressed by a threshold rule:

$$time\_waited_i > \mu\_baseline + k \cdot \sigma\_baseline,$$

where k governs sensitivity (commonly near 3 to filter rare deviations). Interpretation of detected exceedances must remain causally anchored in the operational domain. db file sequential read typically indicates dominance of index-driven single-block reads and a potential IOPS deficit or elevated latency within the storage subsystem; log file sync describes increased commit latency and, consequently, redo-write path problems; latch: cache buffers chains is frequently associated with "hot" blocks and contention; free buffer waits is consistent with an insufficient DB_CACHE_SIZE under the prevailing workload profile.

For the I/O domain, a convenient comparable metric is average read latency:

$$\text{Avg read ms} = readtim / phyblkrd,$$

which permits consistent estimates across different read intensities. Operational practice often relies on indicative targets: single-block reads under ~5 ms, multiblock reads under ~20 ms, redo writes under ~2 ms, and log sync under ~10 ms; degradation is commonly fixed when latency rises by ~30–50% relative to the baseline. Memory conditions are assessed via SGA/PGA pressure signals: a Buffer Cache Hit Ratio dropping below ~90–95% is treated as buffer cache stress, while a PGA cache hit ratio falling below ~95% combined with growth in multipass executions is a symptom cluster consistent with PGA insufficiency.

Background processes warrant dedicated analysis because local delays are often translated into systemic impact through their activity. LGWR-related waits together with elevated log file sync align with redo-disk slowdown; DBWR waits frequently reflect checkpoint and write-path saturation; ARCn lag points to a bottleneck in the archiving direction. Storage-configuration parameters also participate in the causal chain: undersized redo logs increase log-switch frequency, which can amplify log file sync via more frequent and costly operations on the commit path.

An integral formalization can be achieved by constructing a final degradation vector with domain weights, obtained via regression or ML methods, so that deterioration is expressed as a weighted displacement from the baseline in feature space. At the AIOps level, suitable approaches include anomaly detection (Isolation Forest, autoencoders), forecasting (LSTM, Prophet, ARIMA), explainable causality (SHAP, Bayesian networks), and clustering of degradation regimes (K-Means, DBSCAN). For RCA, investigative logic tends to converge on a stable pipeline: ranking of top wait events → domain mapping → correlation with SQL_ID → execution-plan verification → validation of

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

background processes and storage parameters. Such sequencing reduces the risk of "noise-driven diagnosis" and keeps the analysis inside the cause–effect boundaries set by the response-time model.

Results and Discussion

Wait events in Oracle (Wait Event Interface) constitute a key mechanism for attributing execution time and, thus, a basic tool for explaining performance dynamics [13]. The accumulated wait statistics make it possible to decompose the activity profile into a computational component, when a process is predominantly executed on the CPU, and a wait component, reflecting delays in accessing resources and internal synchronization structures [12, 13]. Such a separation forms the basis for diagnostic classification of the instance state and for the subsequent construction of features suitable for predictive modeling.

Within the study, it was established that in industrial environments the most informative predictors of degradation are waits of the User I/O and Concurrency classes [22]. These groups reflect, respectively, the cost of read/write operations on the side of user queries and the intensity of contention for shared resources, locks, and coordination mechanisms. An indicative marker is a persistent increase in wait time for the event db file sequential read, corresponding to single-block reads. With a comparable data volume and an unchanged access profile, such dynamics are interpreted as an indicator of deterioration in the characteristics of physical storage or an increase in contention at the level of the data access subsystem, including transport on the data bus and associated input/output paths [15].

Table 1 presents the classification and determinants of key Oracle DBMS wait events.

Table 1. Classification and determinants of key Oracle DBMS wait events (compiled by the author based on [13-17]).

Waiting class	A typical event	Description of the problem	Predictive value
User I/O	db file sequential read	Index read from disk	Disk subsystem slowdown / Poor indexes
User I/O	db file scattered read	Full table scan	SQL Plan Drift / Insufficient memory
Commit	log file sync	Waiting for LGWR confirmation	Redo log write issues
Concurrency	buffer busy waits	Contention for a block in memory	Excessively high intensity of data updates
Configuration	log buffer space	Insufficient space in the Redo Log Buffer	LGWR does not keep up with flushing data

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

A substantial methodological condition for correct modeling is the exclusion of idle wait events, including SQL*Net message from client, since their inclusion leads to a distortion of the integral picture of database time and blurs the signal associated with real resource constraints [26]. In the predictive setting, the emphasis is shifted to the %DBTime metric, reflecting the proportion of time spent on useful execution or on waits caused by actual resource and synchronization delays [18, 22].

A critical contribution to Oracle performance formation is made by memory management, and degradation scenarios arise both in the system area SGA and in the program area PGA [28, 29]. Empirical results show that the use of V$PGA_TARGET_ADVICE data enables the trained model to predict moments when sort operations move from memory to disk and are executed in multi-pass execution mode, which is accompanied by increased latency and reduced throughput. For OLTP workloads, it remains fundamental to retain on the order of 80% of memory in the SGA in order to effectively cache data blocks, whereas for DSS workloads (Data Warehouse) the share of PGA should reach 50% and above, ensuring effective execution of hash joins and sorts [19, 29]. In combination with the PGA_USED_MEM and PGA_AGGREGATE_LIMIT metrics, predictive analysis becomes a tool for preventing incidents associated with forced termination of DBMS sessions when memory limits are exhausted. Within automated monitoring, early detection of anomalous memory consumption growth in an individual session is possible, followed by a recommendation to adjust the PGA_AGGREGATE_TARGET parameter before the onset of a critical state leading to cascading deterioration of responsiveness and stability [29, 30].

Asynchronous writing of data and logs is ensured by Oracle background processes (LGWR, DBWR, ARCH), which under normal operation allows user sessions not to be blocked on slow input/output operations [31, 32]. Under extreme load profiles, the background processes themselves become sources of delays and form new bottlenecks. The most sensitive component in this loop is LGWR (Log Writer): due to the use of the Write-Ahead Logging (WAL) protocol, changes are not considered committed until redo records have been flushed to disk by this process [27, 33]. An XGBoost-based model focused on analyzing log write latency reveals a direct correlation between LGWR performance and transaction response time, captured via the log file sync wait [11, 24]. Experimental observations demonstrate that LGWR degradation is most often driven by two interrelated factors: an increase in write latency, in which an increase in storage delays even by 1–2 ms initiates disproportionate accumulation of transaction commit queues, as well as contention with the ARCH process during log switches, when intensive reading of redo files for archiving can slow the writing of new data by LGWR under shared use of the same physical disks [20-22].

From the perspective of business effects, the introduction of predictive analytics within the AIOps paradigm in Oracle Database operations demonstrates pronounced practical value: the combination of Oracle autonomous functions and external ML models makes it possible to shift operational activity from reactive mitigation of consequences to proactive control of degradation trajectories and reduction of downtime probability while maintaining the required levels of service commitments.

Table 2 compares analysis of operational and financial determinants of effectiveness when transitioning from reactive DBMS administration to a proactive AIOps-

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

based model.

Table 2. Comparison of operational and financial determinants of effectiveness when transitioning from reactive DBMS administration to a proactive AIOps-based model (compiled by the author based on [34]).

Metric	Traditional approach	Predictive approach (AIOps)	Improvement
Reduction in unplanned downtime	Baseline	-91%	Significant
DBA team efficiency	100% (baseline)	+66%	Productivity growth
Time spent on Lights On support	High	-65%	Cost reduction
Annual benefit ($)	-	$4.9 million (on average)	Direct savings

The obtained results indicate high economic feasibility of implementing intelligent monitoring: investments are recouped within the first five months of industrial operation due to the prevention of major emergency scenarios and more rational consumption of cloud resources [34]. Of substantial importance is the coupling of the predictive loop with storage subsystem parameters and characteristics of cloud services, since by 2022 a notable share of Oracle instances operates in cloud environments (OCI, Azure, AWS) or on specialized hardware-software complexes, including Exadata [35, 36]. Under such conditions, the forecasting model must account for measurable storage properties-IOPS, throughput, and latency at the level of virtualized volumes, since these parameters determine the actual cost of input/output operations and shape the observed wait profiles [15, 23, 25].

Modeling I/O degradation is based on joint analysis of queues on the operating system side and their comparison with Oracle wait events, which makes it possible to link manifestations at the DBMS level with root causes in the infrastructure loop. An illustrative example is the use of the OCI Ops Insights cloud service, which provides integration of telemetry from Enterprise Manager and the use of built-in machine learning algorithms to forecast depletion of disk capacity and CPU resources several months before the onset of a critical event.

A drop in OLTP throughput under rising commit latency typically appears as declining TPS alongside a growing share of waits associated with transaction commit. In AWR terms, this manifests as an increase in log file sync

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

and a stronger redo-path contribution to overall response time, directly undermining transactional SLAs even when CPU utilization remains within acceptable limits.

Observed diagnosis begins with verifying system-level accumulation of waits for the relevant event. A minimally sufficient slice is obtained from V$SYSTEM_EVENT, normalizing microseconds to seconds:

```
SELECT event,
time_waited_micro/1000000
FROM V$SYSTEM_EVENT
WHERE event LIKE 'log file sync';
```

An increase in time_waited for this event at the system level should not be interpreted as a "network" or "client-side SQL" delay. Rather, it is a marker that sessions are waiting for completion of redo writes performed by LGWR-in other words, a degradation of the commit tract is being observed.

Causal reconstruction for this degradation profile commonly converges on elevated LGWR I/O latency. In the configuration under discussion, a key amplifier is a small redo log size on the order of 50 MB, which forces frequent log switches and increases the rate of operations competing for I/O resources. At the infrastructure layer, an additional contributor is a shared SAN subsystem: as background load or neighboring write streams intensify, queuing effects emerge. Since the redo path is synchronous in the functional sense of commit completion, it becomes the limiting factor for transactional throughput. The resulting chain is coherent and repeatable: higher LGWR latency → frequent log switches driven by small redo logs → SAN contention and queuing → accelerated growth of log file sync waits → TPS decline.

Mitigation in this framing reduces to breaking the bottleneck and lowering the frequency of structural redo transitions. In practice, this is achieved by increasing redo log size to roughly 4 GB, relocating redo to a dedicated high-performance medium (SSD-class) to isolate it from competing write streams, and enabling asynchronous redo I/O where supported by the platform and configuration-an intervention that reduces the synchronous latency component on the commit path.

From a predictive standpoint, the situation maps cleanly onto a telemetry architecture in which observability sources (Oracle telemetry) feed an aggregation layer (AWR/ASH/V views), are transformed into engineered features, and are then consumed by an ML model. Subsequent steps include anomaly detection and causal inference (root-cause inference), followed by generation of action recommendations. This pipeline supports a shift from post-factum diagnosis to controlled early warning, since log file sync serves as a stable leading indicator of redo-path degradation in transactional systems.

Conclusion

The proposed formalized, reproducible methodology integrates the classical Oracle diagnostic paradigm-grounded in wait-event analysis and performance time slicing-with AIOps approaches, enabling a shift from reactive remediation of consequences to proactive detection of degradations before SLA-level incidents occur. The methodological framework establishes unified rules for root-cause analysis (RCA), within which observable artifacts from AWR/ASH and V performance views are aligned with log-derived evidence and telemetry streams. Primary-cause

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

conclusions are then substantiated through explicit mathematical constructs and, where appropriate, machine-learning models. This combination supports not only standardization of the investigative procedure, but also its repeatability when transferred across distinct operational environments.

The applicability domain covers the principal industrial Oracle architectures and deployment modes, including RAC, Exadata, Multitenant, and Oracle Cloud Infrastructure (OCI). Such coverage makes it feasible to apply a single diagnostic contour within heterogeneous enterprise landscapes. Scaling to enterprise production is supported by event-representation unification, automated signal correlation, and the ability to replicate models across multiple instances and clusters without sacrificing interpretability of outcomes.

Observed results indicate that combining AWR/ASH/V snapshots with machine-learning algorithms and log analysis reduces MTTR by 40–70 % and MTTD by up to 63 % and enables forecasting of performance degradation hours to days before service deterioration becomes critical. The practical value for Oracle DBA operations is expressed through the emergence of predictive diagnostics, automated RCA, strengthened SLA assurance, more defensible capacity planning, and movement toward autonomous tuning modes in which decisions are anchored in formal criteria and empirically observed workload dynamics.

References

1. Oracle. (2019, February 4). Oracle Database 19c introduction and overview. Retrieved from: https://www.oracle.com/a/tech/docs/database19c-wp.pdf (date accessed: March 18, 2022).
2. International Data Corporation. (2022, May 12). Worldwide digital transformation investments forecast to reach $1.8 trillion in 2022, according to new IDC Spending Guide. Business Wire. Retrieved from: https://www.businesswire.com/news/home/20220512005848/en/Worldwide-Digital-Transformation-Investments-Forecast-to-Reach-%241.8-Trillion-in-2022-According-to-New-IDC-Spending-Guide (date accessed: June 3, 2022).
3. Uptime Institute. (2022, April 29). Annual outage analysis 2022. Uptime Intelligence. Retrieved from: https://intelligence.uptimeinstitute.com/index.php/resource/annual-outage-analysis-2022 (date accessed: June 15, 2022).
4. Uptime Institute. (2022, April 29). Annual outage analysis 2022. Retrieved from: https://intelligence.uptimeinstitute.com/index.php/resource/annual-outage-analysis-2022 (date accessed: June 15, 2022).
5. Oracle. (2021). Oracle Database performance method. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/tdppt/oracle-database-performance-method.html (date accessed: March 21, 2022).
6. Research and Markets. (2021). Global AIOps market 2021–2026. Retrieved from: https://www.researchandmarkets.com/reports/5360397/global-aiops-market-2021-2026 (date accessed: July 1, 2022).
7. Notaro, P., Cardellini, V., Casalicchio, E., & Lo Presti, F. (2021). A survey of AIOps methods for failure management. ACM Transactions on Intelligent Systems and Technology, 12(6), Article 71. https://doi.org/10.1145/3483424
8. Pang, G., Shen, C., Cao, L., & van den Hengel, A. (2021). Deep learning for anomaly detection: A review. ACM Computing Surveys, 54(2), Article 38. https://doi.org/10.1145/3439950
9. Soldani, J., Brogi, A., & Wang, J. (2021). Anomaly detection and failure root cause analysis in

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

(micro)service-based cloud applications: A survey. arXiv. https://doi.org/10.48550/arXiv.2105.12378

10. Marcus, R., Negi, P., Mao, H., Tatbul, N., Alizadeh, M., & Kraska, T. (2021). Bao: Making learned query optimization practical. In Proceedings of the 2021 International Conference on Management of Data (pp. 1275–1288). https://doi.org/10.1145/3448016.3452838
11. Van Aken, D., Yang, D., Brillard, S., Fiorino, A., Zhang, B., Pavlo, A., & Gordon, G. J. (2021). An inquiry into machine learning-based automatic configuration tuning services on real-world database management systems. Proceedings of the VLDB Endowment, 14(7), 1241–1253. https://doi.org/10.14778/3450980.3450992
12. Nordal, H., & El-Thalji, I. (2021). Assessing the technical specifications of predictive maintenance: A case study of centrifugal compressor. Applied Sciences, 11(4), Article 1527. https://doi.org/10.3390/app11041527
13. Oracle. (2021). Descriptions of wait events. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/refrn/descriptions-of-wait-events.html (date accessed: March 29, 2022).
14. Oracle. (2021). Automatic performance diagnostics. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/tgdba/automatic-performance-diagnostics.html (date accessed: April 4, 2022).
15. Oracle. (2021). Gathering optimizer statistics. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/tgsql/gathering-optimizer-statistics.html (date accessed: April 11, 2022).
16. Pavlo, A., Butrovich, M., Joshi, A., Ma, L., Menon, P., Misra, P., Ryan, G., Stonebraker, M., & Zdonik, S. (2021). Make your database system dream of electric sheep: Towards self-driving operation. Proceedings of the VLDB Endowment, 14(12), 3211–3221. https://doi.org/10.14778/3476311.3476411
17. Oracle. (2021). Automatic database performance monitoring. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/tdppt/automatic-database-performance-monitoring.html (date accessed: April 18, 2022).
18. Datadog. (n.d.). Oracle database. Retrieved from: https://docs.datadoghq.com/integrations/oracle/ (date accessed: April 26, 2022).
19. Guo, H., Yuan, S., & Wu, X. (2021). LogBERT: Log anomaly detection via BERT. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1–8). https://doi.org/10.1109/IJCNN52387.2021.9534113
20. Oracle. (2022, April 29). 2-steps OCI Logging and OCI Log Analytics enablement for OCI services. Retrieved from: https://docs.oracle.com/iaas/releasenotes/changes/35ef3492-4164-4e38-b9c4-c80bb15a589e/index.htm (date accessed: May 2, 2022).
21. Li, G., Zhou, X., Sun, J., & Aken, D. V. (2021). Machine learning for databases. Proceedings of the VLDB Endowment, 14(12), 3190–3193. https://doi.org/10.14778/3476311.3476405
22. Oracle. (2021). Database performance tuning guide. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/tgdba/database-performance-tuning-guide.pdf (date accessed: May 9, 2022).
23. Li, G., Zhou, X., Sun, J., Yu, X., Han, Y., Jin, L., Li, W., Wang, T., & Li, S. (2021). openGauss: An autonomous database system. Proceedings of the VLDB Endowment, 14(12), 3028–3041. https://doi.org/10.14778/3476311.3476380
24. Oracle. (2021). Real application clusters administration and deployment guide. Retrieved

The American Journal of Engineering and Technology
(ISSN – 2689-0984)
VOLUME 04 ISSUE 09 Pages: 17-27
SJIF IMPACT FACTOR (2020: 5. 32) (2021: 5. 705) (2022: 6. 456)
OCLC – 1091588944
Publisher: The USA Journals

from: https://docs.oracle.com/en/database/oracle/oracle-database/21/racad/real-application-clusters-administration-and-deployment-guide.pdf (date accessed: May 16, 2022).

25. Zhou, X., Jin, L., Sun, J., Zhao, X., Wang, T., & Li, S. (2021). DBMind: A self-driving platform in openGauss. Proceedings of the VLDB Endowment, 14(12), 2743–2746. https://doi.org/10.14778/3476311.3476334
26. Oracle. (2021). V$EVENT_NAME. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/refrn/V-EVENT_NAME.html (date accessed: May 23, 2022).
27. Oracle. (2021). Configuring options for optimizer statistics gathering. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/tgsql/options-for-optimizer-statistics-gathering.html (date accessed: May 31, 2022).
28. Oracle. (2021). Monitoring real-time database performance. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/tdppt/monitoring-real-time-database-performance.html (date accessed: June 6, 2022).
29. Oracle. (2021). Managing memory. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/admin/managing-memory.html (date accessed: June 13, 2022).
30. Oracle. (2021). PGA_AGGREGATE_TARGET. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/refrn/PGA_AGGREGATE_TARGET.html (date accessed: June 20, 2022).
31. Oracle. (2021). Oracle Database concepts. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/cncpt/database-concepts.pdf (date accessed: June 27, 2022).
32. Oracle. (2021). Oracle Database reference. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/refrn/database-reference.pdf (date accessed: July 4, 2022).
33. Oracle. (2021). Database performance tuning guide. Retrieved from: https://docs.oracle.com/en/database/oracle/oracle-database/21/tgdba/database-performance-tuning-guide.pdf (date accessed: July 8, 2022).
34. Oracle. (2019). Thinking autonomous: What's your business's data worth? Retrieved from: https://www.oracle.com/a/ocom/docs/thinking-autonomous-business-data-worth.pdf (date accessed: July 12, 2022).
35. Oracle. (2022, July). Using Oracle Analytics Day by Day. Retrieved from: https://docs.oracle.com/en/cloud/paas/analytics-cloud/biday/index.html (date accessed: July 18, 2022).
36. Oracle. (2020, October 20). Ops Insights. Retrieved from: https://docs.oracle.com/iaas/releasenotes/services/operations-insights/ (date accessed: July 25, 2022).