INFORMATION SYSTEMS

Optimization of Oracle Real Application Clusters for High-Availability and Fault-Tolerant Healthcare Database Infrastructure

Mujeeb Pullumalayil

Published November 2026

Abstract

Healthcare database infrastructure must remain responsive during emergencies, routine clinical work, planned maintenance and equipment failures. Oracle Real Application Clusters (RAC) offers a shared-database architecture in which multiple instances can support services, but high availability cannot be established by clustering alone. This integrative review examines how RAC configuration, database service design, connection recovery, transaction integrity, workload management, storage resilience, monitoring and disaster recovery can be combined for fault-tolerant healthcare information systems. It proposes a layered optimisation framework connecting clinical service criticality to recovery objectives, technical controls, failure testing and operational governance. Particular attention is given to electronic health record transactions, medication administration, laboratory reporting, admission workflows and interfaces whose behaviour during partial failures may differ. The synthesis identifies the most consequential trade-offs: local continuity versus site-wide recovery, throughput versus tail latency, automated replay versus clinical transaction safety, and infrastructure redundancy versus common-cause failure. A phased assurance model is offered for healthcare organisations in Saudi Arabia and comparable settings. The study is conceptual, reports no experimental performance gains and recommends prospective workload-specific validation before operational deployment.

Keywords: Oracle Real Application Clusters; healthcare databases; high availability; fault tolerance; Application Continuity; transaction integrity; workload management; disaster recovery; database resilience

1 Introduction

Digitised healthcare makes databases an operational dependency rather than a passive repository. Patient registration, electronic prescribing, laboratory orders, clinical documentation, radiology scheduling and revenue workflows often depend on coordinated reads and writes across tightly coupled applications. If the database becomes unavailable, some activities can temporarily revert to downtime procedures, while others may face immediate clinical or administrative consequences. Reliability must therefore be evaluated at the level of an end-to-end care workflow rather than by server uptime alone.

Oracle RAC allows several database instances running on separate nodes to access a common database. Its principal promise is local instance resilience and workload distribution. When an instance fails, services may be recovered on surviving nodes, but sessions, incomplete requests, shared storage, middleware or external clinical interfaces may still experience interruption. A hospital cannot infer uninterrupted care from the presence of two database nodes. Indeed, an overloaded surviving node can turn an otherwise recoverable failure into a sustained service degradation.

This review addresses an implementation gap between database engineering and clinical service continuity. It considers the way service priorities, failure domains, transaction semantics, latency, recovery mechanisms, data protection and human procedures interact. Its emphasis is not a benchmark claim for a particular server arrangement but a defensible architecture and evaluation method. The Saudi healthcare transformation context motivates attention to distributed facilities, interoperability and governance, although individual hospitals require their own measured requirements and regulatory assessment.

2 Aim, objectives and research questions

The aim is to develop an operationally testable framework for optimising Oracle RAC in high-availability, fault-tolerant healthcare database environments while preserving transaction correctness and safe clinical workflows.

Five objectives guide the review. First, establish healthcare-specific service tiers and recovery targets. Second, examine RAC architectural dependencies and workload configuration. Third, evaluate failure detection, connection failover and transaction replay controls. Fourth, integrate database-level high availability with data protection, disaster recovery and operational security. Fifth, propose an evidence-based implementation and testing roadmap.

The research asks: Which failure modes are realistically contained by RAC and which require other mechanisms? How should clinical workload characteristics guide service placement, capacity and instance affinity? When can automated client recovery preserve the semantics of healthcare transactions? What measures distinguish apparent database availability from actual clinical workflow continuity? How can a healthcare operator demonstrate resilience through repeatable tests rather than unverified assumptions?

3 Review methodology

This paper adopts an integrative technical review to synthesise vendor architecture guidance, health information system requirements, resilience standards and selected applied computing literature. The source base emphasises Oracle Database 19c RAC and associated availability technologies, together with institutional guidance available through 2025. The synthesis distinguishes documented product capabilities from proposed implementation choices; it does not treat configuration examples as evidence of universal clinical effectiveness.

The conceptual search themes comprised Oracle RAC, Clusterware, Cache Fusion, services, Fast Application Notification, connection pooling, Application Continuity, Transaction Guard, Active Data Guard, Recovery Manager, healthcare databases, electronic health records, disaster recovery, cyber resilience and business continuity. Materials were screened for direct relevance to architecture, fault behaviour, validation or operational governance. Generic availability claims without a clear failure model were not used to support design recommendations. Because a reproducible database-search log and screening dataset were not assembled, the review is not presented as a systematic review or meta-analysis.

Evidence is organised thematically, with an explicit distinction among local node failure, software defects, network partitions, shared-storage failure, data corruption and site loss. The outcome is a design framework and a set of testable hypotheses, not measured service-level performance. Future

studies should compare it against local baselines using prospective clinical workloads, failure injection and independently reviewed outcomes.

4 Clinical criticality and service-level design

Availability objectives must reflect the practical consequences of failed clinical functions. A medication administration record used at the bedside, for example, has different tolerance for delay and duplicate processing than an overnight reporting extract. Hospitals should inventory workflows, identify required databases and interfaces, and classify impact across patient safety, operational disruption, privacy, compliance and recovery effort. A technical service tier is justified only when it maps to such a clinical dependency.

Recovery time objective describes the tolerable duration of disruption; recovery point objective describes acceptable data loss under a particular disaster scenario. Neither represents a guarantee of application recovery, and both need an agreed measurement boundary. The clock may start with the actual loss of a clinician's ability to complete a transaction rather than the moment a database alert fires. Restoration is not complete when an instance is merely open; the application must authenticate, obtain a usable connection, reconcile ambiguous transactions and complete its clinical operation.

Tiering also determines protection priorities. High-consequence services warrant strict validation of order-entry integrity, rapid error detection and human escalation. Less time-sensitive analytics can be isolated through separate services and scheduled workloads to protect essential capacity. Operational owners should agree downtime procedures, back-entry rules and reconciliation requirements before a failure occurs. These workflows must be exercised alongside technical failover testing, since a technically successful recovery can still leave clinical records incomplete.

5 RAC architecture and failure domains

A RAC database uses multiple instances to access the same underlying database files. Oracle Clusterware manages cluster membership and resources, while Oracle Automatic Storage Management may manage database storage. Cache Fusion coordinates access to database blocks across instance buffer caches using the cluster interconnect. These mechanisms permit coordinated multi-instance access, but they also introduce dependencies that must be engineered and monitored.

Nodes should be deployed with consideration for power, network interfaces, operating system maintenance and resource isolation. Redundant interconnect paths reduce some connectivity risks but do not eliminate common switch, configuration or software failures. Shared storage requires its own resilience plan: a cluster with two healthy database nodes may still become unavailable if the shared storage layer cannot serve data. Quorum, fencing and membership mechanisms are fundamental to preventing conflicting access during communication failures.

6 Workload characterisation and capacity planning

Healthcare databases are heterogeneous. Bedside documentation often comprises short, interactive transactions; medication and laboratory workflows demand strict write integrity; imaging-related metadata can cause bursts of concurrent requests; reports and extracts may consume sustained CPU and I/O bandwidth. Treating these as a uniform load conceals the conditions under which RAC succeeds or fails.

Capacity planning should begin with a measured demand model: concurrency, transaction mix, peak arrival rates, CPU consumption, memory pressure, storage latency, redo generation, lock contention and inter-instance block movement. Percentile response times matter because a tolerable average may conceal unacceptable delays for clinicians during busy periods. The effect of a failed instance must be tested with remaining capacity, not estimated solely by dividing nominal CPU totals.

Workload isolation through database services can assign critical applications to suitable preferred and available instances. Resource Manager policies may prevent a background workload from exhausting resources needed for clinical use. Service design should avoid fragile dependency on a single preferred node where availability requires relocation. However, excessive instance affinity can imbalance utilisation and degrade throughput when a node is removed. Operators should establish performance baselines with realistic data volumes, query plans and concurrent interfaces, then retest after schema changes, upgrades or clinical application releases. Representative workload risks and controls are summarised in Table 1.

Table 1. Representative healthcare workload risks and controls.

Clinical workload	Principal performance risk	Optimisation and verification
Bedside documentation	Tail latency and connection loss	Protect service capacity; test end-to-end completion
Medication and orders	Ambiguous commits or duplicates	Use transaction identifiers and outcome reconciliation
Laboratory interfaces	Queue backlogs after recovery	Monitor acknowledgements and downstream completion
Reporting and extracts	Resource competition with clinical work	Isolate service demand and apply resource controls

7 Cache Fusion, interconnect and database contention

Cache Fusion is central to RAC scalability and can become a source of avoidable latency when workloads repeatedly contend for the same blocks across instances. Excessive global cache transfers may reflect hot indexes, frequently modified control rows, sequential identifiers, poor partitioning or application patterns that concentrate updates. A faster network cannot fully compensate for an application design that generates unnecessary cross-instance coordination.

Engineering teams should examine global cache wait events alongside ordinary database wait analysis, SQL execution plans, row-level contention and logical I/O. Their first task is to identify the clinical or batch operation responsible for contention, not immediately change hidden database parameters. Well-chosen indexes, batched work, partitioning, transaction-duration discipline and service affinity may improve behaviour, but changes must be proven against query correctness and healthcare workflow requirements.

8 Services, connection pools and failure notification

Database services provide an application-facing unit of placement and recovery. Rather than hard-coding instance addresses, supported client configurations should connect through appropriate service names and cluster-aware mechanisms. This separates the clinical application's identity from

the node currently executing its work. It also allows administrators to move workload during maintenance and capacity changes while preserving operational ownership.

Fast Application Notification communicates service and instance events to compatible clients and pools. Fast Connection Failover allows supported pools to remove unusable connections promptly and redistribute subsequent work. These capabilities reduce dependence on lengthy socket timeouts but require appropriate drivers, pool settings and event connectivity. Incompatible clients may still experience slow failure detection, repeated login attempts or user-visible errors.

Connection establishment should be examined under both ordinary load and failure conditions. Excessive reconnection storms can overload a surviving instance, so pool limits, timeout budgets, backoff and admission controls require testing. Application teams should classify calls by idempotency and error handling. Retrying a read is generally different from blindly repeating a medication order or financial charge. A service may be advertised as available while a middle-tier cache, authentication dependency or queue is not healthy. Therefore, synthetic application journeys must complement database service checks.

9 Transaction Guard, Application Continuity and safe replay

The most consequential concern during connection loss is the uncertainty surrounding in-flight work. A client may lose contact after a transaction committed but before it received confirmation. Reissuing that request without checking its outcome can create a duplicate order, duplicate administration record or incorrect patient status transition. Transaction Guard is designed to help determine the outcome of a transaction when a recoverable error occurs, reducing ambiguity for supported clients and workflows.

Application Continuity and Transparent Application Continuity can recover certain interrupted requests by establishing a new session and replaying recoverable work when conditions permit. These technologies are not unconditional guarantees. Replay eligibility, session state, calls with external side effects, mutable values, request boundaries and client support affect protection. A request that initiated an external notification or third-party operation may require explicit application-level reconciliation regardless of database replay mechanisms.

Healthcare organisations should produce a transaction-semantic inventory. For each critical workflow, the inventory should identify the database transaction, external effects, unique business identifier, expected duplicate-prevention mechanism and acceptable user-facing message. Tests should terminate instances at carefully chosen points: before commit, during commit, after commit and during response delivery. The validation criterion is not simply that a screen returns; it is that the record is correct exactly once and an audit trail explains the outcome. Clinical safety and transaction correctness take precedence over invisible replay.

10 Proposed layered optimisation framework

The proposed framework comprises six mutually reinforcing layers. Clinical governance defines which services must continue and what failure consequences are tolerable. Database architecture maps those requirements to instances, services, storage and networks. Workload optimisation establishes capacity headroom and controls cross-instance contention. Session and transaction continuity manages client recovery, verified outcomes and replay boundaries. Data protection covers

backups, corruption response and geographic recovery. Operational assurance links telemetry, failure drills, review and corrective action.

This arrangement deliberately places end-user outcome above cluster status. A monitoring dashboard should trace each important journey from a clinical action through the application interface, database service, transaction outcome and downstream acknowledgement. Dependencies should be labelled by responsible owner and recovery procedure. A failure in the shared storage layer invokes a different plan from a single instance crash; a data-corruption event requires trusted recovery points rather than routine service relocation. The framework is shown in Figure 1.

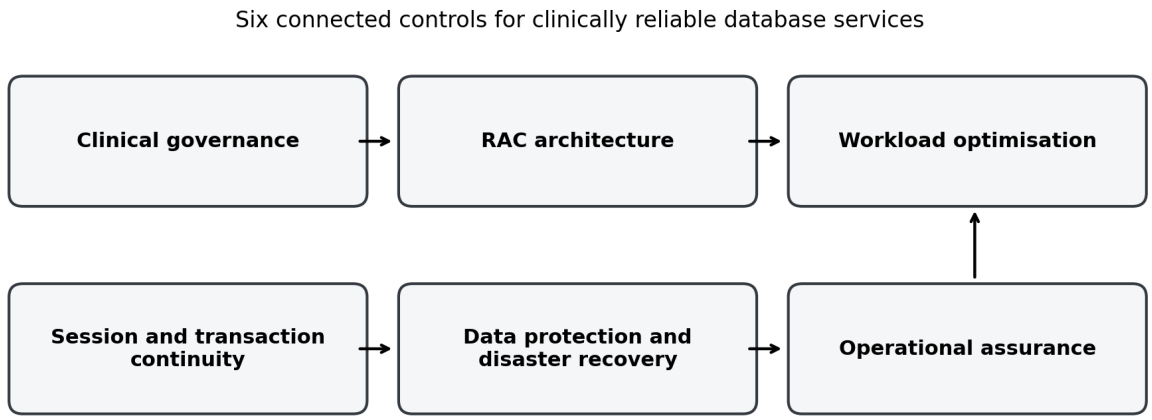


Figure 1. Layered RAC optimisation and clinical resilience framework.

11 Monitoring, alerting and observability

RAC health metrics should be interpreted as evidence about clinical service, not as a substitute for it. Database monitoring typically includes instance and service status, session counts, CPU utilisation, I/O latency, wait classes, global cache waits, interconnect health, redo activity, storage capacity and error events. Application monitoring adds request latency, transaction outcomes, exception rates, queue depth and completion of representative workflows.

Alerts need tested ownership and urgency. A short transient event may justify automated recovery and later review; growing redo latency or exhausted connections during a medication round may need immediate escalation. Thresholds should reflect workload seasonality and capacity after node loss. Alert floods can obscure a single root cause, so correlation and suppression rules should reduce noise while preserving safety-relevant signals. Telemetry itself needs resilience and an independent path for urgent notification.

Performance baselines must remain comparable across software changes and peak periods. A synthetic transaction can exercise user login, chart retrieval and order submission using non-patient test data; it should not create clinical records in production. Monitoring should expose whether the workflow completed, partially completed or failed ambiguously. Review teams should record the elapsed time between an actual service interruption, detection, diagnosis, intervention and demonstrated restoration. These intervals often reveal more actionable improvement opportunities than the number of cluster alerts alone.

A service-level assurance record should preserve three independent timestamps: first unsuccessful clinical action, restoration of dependable application access, and completion of transaction reconciliation. Separating these events prevents a misleading claim of recovery when sessions reconnect quickly but user work remains unresolved. The record should also identify the type of failure, the workload active when it occurred, and any dependency that delayed recovery. Such details allow management to distinguish an isolated component defect from a recurring architectural limitation. Evidence quality matters as much as the numerical target: an unobserved interval should be marked unknown, not treated as zero disruption. Finally, a post-incident review should compare the actual user experience with the simulated failure test and require corrective action when the two diverge.

12 Data protection, disaster recovery and cyber resilience

RAC addresses selected local failures; it is not a replacement for geographically separate recovery or a complete data-protection strategy. A multi-instance database can be affected by shared storage loss, corruption, malicious change, operator error or a site-wide event. Oracle Data Guard can support a separate standby database, while Recovery Manager provides backup and restore capabilities. The choice of technologies and protection mode must reflect measured recovery objectives, network constraints, costs and application requirements.

A disaster recovery runbook should define authority to declare a disaster, replication state checks, failover decision criteria, application endpoint changes, reconciliation and controlled failback. Cross-site recovery must include laboratory interfaces, messaging systems, identity platforms and external integrations, not merely database opening. Exercises should evaluate whether clinicians can find and trust the current patient record after a switch. The objective is not architectural elegance but dependable restoration of correct clinical information.

13 Security, privacy and healthcare data governance

Database resilience and information protection are inseparable. Healthcare records contain sensitive personal and clinical information; an availability measure that exposes those records or weakens auditability is not acceptable. Access privileges should follow least-privilege principles, privileged actions should be logged, and credentials should be governed across both active and recovery environments. Encryption, patch management and vulnerability review must cover nodes, storage, backups, monitoring services and operational scripts.

For deployment in Saudi Arabia, organisations should assess applicable personal-data protection, health-sector obligations and national cybersecurity requirements with qualified local advisers. Cross-border standby placement, backup transfers and vendor support access are governance questions, not merely network configuration details. Resilience evidence should be retained without copying unnecessary patient information into logs or test datasets. Anonymised synthetic workloads and carefully controlled production traces can support assurance while reducing secondary exposure of health data.

14 Evaluation metrics and controlled failure experiments

A defensible evaluation programme combines availability, performance, integrity and clinical usability measures. Database instance uptime alone misses periods when applications cannot

connect or transactions cannot finish. Recommended indicators include end-to-end workflow completion, recovery time measured from the user perspective, unsuccessful or ambiguous transaction rate, duplicate business events, p95 and p99 response times, service relocation delay, peak error rate and post-failover capacity. Site-recovery tests additionally measure data-loss bounds and reconciliation workload.

Failure experiments should follow an approved, non-disruptive progression. Begin in a representative test environment with realistic anonymised data, supported drivers and production-like load patterns. Inject individual instance failures, node loss, blocked interconnects, service relocations, pool exhaustion, storage degradation and constrained survivor capacity. Record expected and observed outcomes, including a clinician-facing timeline. Tests must not create uncontrolled hazards in live care environments.

A critical safeguard is transaction-level reconciliation. Before each experiment, record test order identifiers and expected state transitions. Afterwards, examine database rows, application logs, audit records and downstream messages to confirm correctness. Evaluate both planned and unplanned maintenance since connection draining behaves differently from abrupt loss. Repeat tests after material patching, driver upgrades or application releases. The proposed framework is supported only when measured performance meets agreed objectives under documented loads and credible failure scenarios; this paper does not assert any such result. The failure tests and the evidence required are listed in Table 2, and the end-to-end recovery and verification pathway is shown in Figure 2.

Table 2. Failure tests and evidence required for assurance.

Failure scenario	Required test evidence	Acceptance focus
Single instance outage	User journey timings; FAN events; connection traces	Clinical completion and controlled tail latency
Mid-transaction connection loss	Transaction identifiers; replay traces; downstream audit	No duplicate or missing business event
Survivor capacity stress	p95/p99 latency; queue depth; resource metrics	Critical clinical workload remains usable
Shared storage or site failure	Standby/restore evidence; consistency verification	Approved recovery objectives and integrity

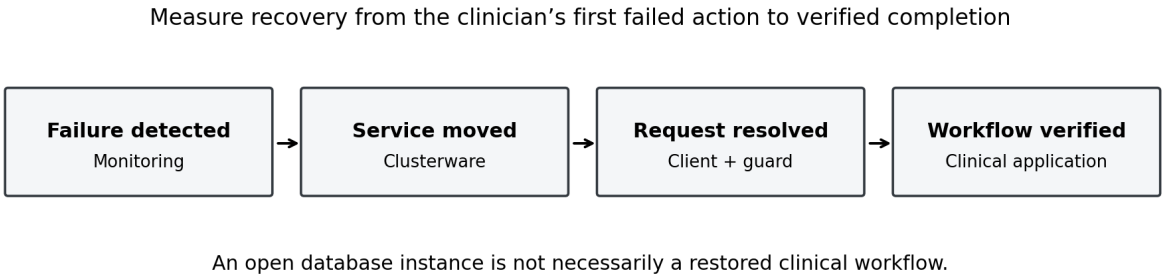


Figure 2. End-to-end service recovery and verification pathway.

15 Implementation roadmap and assurance

A phased programme begins with a dependency and criticality assessment. Healthcare and IT owners identify patient-facing workflows, record the present architecture and agree recovery and integrity targets. The next phase establishes baseline observability and an application transaction inventory. Teams then validate RAC services, connection pool behaviour, cluster interconnect health, storage dependencies and surviving-node capacity before introducing more complex replay configuration.

The pilot should select a bounded clinical workload whose transaction semantics are understood. Configuration changes are made under version control and tested first in a representative environment. Engineers compare normal operation with maintenance and fault conditions, recording service interruption, tail latency, data correctness and operational intervention. A multidisciplinary review determines whether the new configuration provides meaningful improvement without increasing duplicate-processing risk or hidden fragility.

Scaling requires standard runbooks, operational ownership, patching protocols and a recurring exercise calendar. A governance forum should include clinical informatics, database administration, application development, infrastructure, cybersecurity, continuity and quality assurance. Supplier claims must be matched with local compatibility testing. Independent review is especially appropriate for failover involving treatment orders, medication administration or other high-consequence state changes. Any unresolved integrity defect should block deployment, regardless of attractive uptime figures. This disciplined sequence turns clustering from an installed product into an evidenced operational capability.

16 Discussion

The synthesis suggests that the central design problem is not simply the number of RAC nodes but the correspondence between failure mechanisms and protective controls. RAC can keep a service operating after some instance failures, yet its protection stops at shared dependencies and application behaviours that it does not independently repair. Service relocation without sufficient spare capacity may preserve a technical endpoint while leaving clinical users facing unacceptable delays. Similarly, automatic session replay is valuable only when transaction outcome and side effects are controlled.

A second tension concerns automation. Automatic recovery can shorten outages, but opaque retry mechanisms can mask clinically significant uncertainty. Clinicians may need an explicit message that an order's outcome is being checked rather than a silently resubmitted action. Transaction Guard and compatible continuity features are components of a safer system, not substitutes for business identifiers, reconciliation or downstream coordination. The most defensible measure of success remains correct clinical work completed within an acceptable time.

Third, availability requires a broader security perspective. Corruption, ransomware, inadvertent deletion and failed replication are not solved by adding a database node. Recoverable backups, well-tested standby arrangements and independent operating procedures complete the resilience argument. Institutions also need resilience for identity, network and interoperability services. A database can be intact while clinicians lack a functioning pathway to it.

Finally, the proposed framework treats test evidence as part of governance. An architecture diagram documents intention, while a repeatable failure experiment reveals actual behaviour under specified conditions. This distinction matters for procurement and audit, where high availability is sometimes

reduced to a percentage in a service agreement. Clinical owners should approve meaningful endpoints and failure scenarios. The database team should report the boundaries of protection and disclose residual failure modes instead of representing RAC as a universal safeguard.

17 Research gaps and future research

Comparative healthcare-specific studies of RAC configurations remain valuable because many public performance claims do not disclose workload mix, transaction semantics or measurement boundaries. Prospective research could compare service policies, affinity strategies, connection pool configurations and replay protection under controlled node failures. Experiments should report patient-workflow completion, tail latency and data correctness rather than only transactions per second.

Human factors also warrant direct evaluation. Research should examine whether clinicians interpret recovery messages correctly, when manual downtime procedures become safer than automated retries, and how recovery uncertainty affects trust in electronic records. Cross-institutional evaluation in Saudi healthcare environments could provide guidance on governance and service criticality, but must protect patient privacy and account for variation in hospital scale, technology and clinical workflow.

18 Practical, managerial and policy implications

Healthcare executives should commission availability planning as a patient-service continuity programme rather than a database upgrade project. Clinical leadership defines the essential journeys and the consequences of lost or duplicated transactions. Technology teams translate those requirements into node capacity, service placement, supported client features, monitoring, standby arrangements and test evidence. Budget decisions should include sustained maintenance, independent review and rehearsed recovery capability, not only hardware procurement.

Database and application managers should maintain a shared service catalogue identifying recovery targets, owners, dependencies and known replay restrictions. Incident reviews should investigate end-user disruption and transaction correctness even when cluster automation appears to have worked. Procurement terms can require compatibility disclosures, patch support, test participation, recovery documentation and access to diagnostic evidence.

19 Limitations

This paper presents a conceptual integrative review and does not report newly collected experimental data, a reproducible systematic search or a statistical meta-analysis. Oracle RAC behaviour depends on version, licensing, driver support, application design and deployment topology. Some architecture documents describe capabilities that require deliberate configuration and validation. The framework does not establish numeric recovery guarantees or represent legal compliance advice. The proposed test programme should be adapted by qualified clinical and engineering professionals to local safety constraints.

20 Conclusion

Optimising Oracle RAC for healthcare infrastructure requires a layered design that connects database availability to real clinical outcomes. RAC can provide local instance resilience and workload flexibility, but shared dependencies, connection handling, transaction ambiguity, data corruption and site-wide events demand additional measures. A strong implementation uses clinically defined service tiers, measured capacity headroom, well-designed services, compatible failure notification, safe transaction recovery, protected backups and tested disaster recovery.

The framework proposed here emphasises clear failure domains and repeated verification of user-visible continuity and record integrity. It makes no universal performance claim: healthcare organisations must demonstrate that specific configurations achieve agreed outcomes under representative failures and workloads. In this role, RAC is an important technical component of a broader patient-centred resilience system, not a standalone guarantee that every clinical transaction will continue uninterrupted.

References

[1] Oracle. *Application Development Best Practices for Oracle Real Application Clusters (RAC): A Developer's Checklist*. Oracle technical white paper; 2024.

[2] Oracle. *Oracle Database 19c Real Application Clusters Administration and Deployment Guide*. Oracle Database Documentation; 2023.

[3] Oracle. Oracle Database 19c Real Application Clusters Installation Guide for Linux and UNIX. Oracle Database Documentation; 2023.

[4] Oracle. *Oracle Database 19c Clusterware Administration and Deployment Guide*. Oracle Database Documentation; 2023.

[5] Oracle. *Oracle Database 19c High Availability Overview*. Oracle Database Documentation; 2023.

[6] Oracle. *Oracle Database 19c Administrator's Guide*. Oracle Database Documentation; 2023.

[7] Oracle. *Oracle Database 19c Performance Tuning Guide*. Oracle Database Documentation; 2023.

[8] Oracle. *Oracle Database 19c Reference*. Oracle Database Documentation; 2023.

[9] Oracle. *Oracle Database 19c Automatic Storage Management Administrator's Guide*. Oracle Database Documentation; 2023.

[10] Oracle. *Oracle Database 19c Backup and Recovery User's Guide*. Oracle Database Documentation; 2023.

[11] Oracle. *Oracle Database 19c Data Guard Concepts and Administration*. Oracle Database Documentation; 2023.

[12] Oracle. *Oracle Database 19c Security Guide*. Oracle Database Documentation; 2023.

[13] Oracle. *Oracle Database 19c Net Services Administrator's Guide*. Oracle Database Documentation; 2023.

[14] Oracle. *Oracle Database 19c Global Data Services Concepts and Administration Guide*. Oracle Database Documentation; 2023.

[15] Oracle. *Oracle Database 19c PL/SQL Packages and Types Reference*. Oracle Database Documentation; 2023.

[16] Oracle. *Oracle Maximum Availability Architecture: Oracle RAC and Clusterware Best Practices*. Oracle MAA guidance; 2024.

[17] Oracle. *Oracle Maximum Availability Architecture: Oracle Database High Availability Best Practices*. Oracle MAA guidance; 2024.

[18] Oracle. *Oracle Maximum Availability Architecture: Oracle Data Guard Best Practices*. Oracle MAA guidance; 2024.

[19] National Institute of Standards and Technology. *The NIST Cybersecurity Framework (CSF) 2.0*. NIST CSWP 29; 2024.

[20] National Institute of Standards and Technology. *Security and Privacy Controls for Information Systems and Organizations*. NIST SP 800-53 Revision 5; 2020.

[21] National Institute of Standards and Technology. *Contingency Planning Guide for Federal Information Systems*. NIST SP 800-34 Revision 1; 2010.

[22] International Organization for Standardization. *ISO/IEC 27001:2022: Information security management systems — Requirements*. Geneva: ISO; 2022.

[23] International Organization for Standardization. *ISO 22301:2019: Security and resilience — Business continuity management systems — Requirements*. Geneva: ISO; 2019.

[24] Saudi Data and Artificial Intelligence Authority. *Personal Data Protection Law Implementing Regulations*. Kingdom of Saudi Arabia; 2023.

[25] World Health Organization. *Global Strategy on Digital Health 2020–2025*. World Health Organization; 2021.