Journal Homepage: -www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/24532
DOI URL: http://dx.doi.org/10.21474/IJAR01/24532

RESEARCH ARTICLE

DESIGN AND PERFORMANCE EVALUATION OF HIGH-AVAILABILITY CLOUD DATABASE ARCHITECTURES FOR SAUDI GOVERNMENT DIGITAL INFRASTRUCTURE

Razi Ahmad

Manuscript Info

Manuscript History
Received: 08 September 2026
Final Accepted: 10 October 2026
Published: November 2026

Key words:-
High availability; cloud databases; Saudi digital government; replication; consistency; disaster recovery; service-level objectives; data sovereignty.

Abstract

Government database continuity requires more than duplicate servers: transaction semantics, service dependencies, recovery authorization, and lawful data placement determine whether a supposedly available platform can sustain public services. This integrative review evaluates cloud database architectures for Saudi digital government, drawing on peer-reviewed research published mainly between 2020 and 2025. It distinguishes zonal replication, geographically separated standby, multiwriter designs, and distributed transaction systems; assesses their effects regarding consistency, tail latency, recovery objectives, and security; and proposes an evidence-based design and evaluation framework. The synthesis identifies an unresolved tension between rapid automated recovery and reliable protection against divergent writes, corrupted replicas, and unauthorized data movement. It argues that high availability should be shown through workload-specific failure experiments and traceable governance decisions, not inferred from advertised infrastructure redundancy. The resulting framework prioritizes criticality-tiered architecture selection, quorum-aware failover, isolated recovery copies, and measurable service-level indicators. This is a conceptual synthesis, not an empirical benchmark; recommendations require confirmation against specific agencies, regulations, and workloads.

"© 2026 by the Author(s). Published by IJAR under CC BY 4.0. Unrestricted use allowed with credit to the author."

Introduction:-

Saudi government digitization depends on databases that support identity services, permits, payments, public records, and interagency exchanges. A database outage may interrupt an apparently unrelated portal through shared authentication, reference data, or transaction infrastructure. Availability therefore belongs to an end-to-end service chain rather than to a single database instance. This distinction is important when procurement descriptions promise redundant nodes but omit dependencies such as name resolution, keys, access gateways, and management planes. Cloud computing offers managed failover, distributed storage, and elastic processing, yet each feature adds design choices that matter for correctness and public accountability.The literature shows that geo-replication can provide fault isolation while increasing the cost of synchronous commits. Ferreira and colleagues experimentally compare consistency settings and deployment distances, highlighting workload-specific differences among NoSQL engines [1]. Hamrouni and coauthors place replication within wider economic and correlation-aware objectives [3], whereas

Corresponding Author:- Razi Ahmad

Lima and colleagues show that causal retrieval policies can trade latency against financial cost [4]. These results caution against using the number of replicas as a surrogate for availability. Government systems must consider whether a successful response reflects the authorized latest state, whether a failover preserves committed transactions, and whether the service owner permits degraded operation.

A second motivation concerns the operating context. Government records may include sensitive personal information, administrative decisions, and legally significant timestamps. Therefore, decisions about backups, read replicas, and cross-region recovery require classification and authorization. Assess Saudi-specific compliance obligations against current official requirements during implementation; this paper does not provide legal advice. It contributes by combining database design evidence with a transparent evaluation protocol and governance checkpoints, making architecture choices contestable and auditable rather than vendor-driven.

Aim of the Study:-

The aim is to develop a critical, actionable framework for designing and evaluating high-availability cloud database architectures that maintain transactional trust and continuity for Saudi government services. The framework treats resilience as a measurable combination of functioning service access, committed-data preservation, dependency readiness, and policy-conforming recovery. It does not assume that one database product or deployment model is universally optimal.

Research Objectives:-

Objective 1: distinguish the operational failure domains and transaction guarantees of candidate cloud database patterns. Objective 2: compare replication, consensus, recovery, and tuning approaches against latency, integrity, and resource expenditure. Objective 3: define a reproducible evaluation protocol suitable for different government service criticality levels. Objective 4: incorporate data classification, oversight, and authorized recovery boundaries into architecture selection. Objective 5: identify empirical gaps and testable hypotheses for future Saudi public-sector studies.

Research Questions:-

RQ1: Which combinations of replication topology and consistency guarantees remain suitable when zones, links, or operators fail? RQ2: How should teams measure the difference between infrastructure uptime and application availability that preserves correctness? RQ3: Which operational controls prevent automated recovery from amplifying corruption, credential compromise, or unlawful data transfer? RQ4: What comparative evidence is still needed before a Saudi government agency can justify multi-zone, multi-region, or hybrid database deployment?

Review Methodology:-

The study selected a structured integrative review design because the research question spans database systems, distributed computing, cloud operations, security, and public-sector governance. Candidate evidence was sought through publisher repositories and scholarly indexing routes associated with ACM Digital Library, Springer Nature, IEEE Xplore, Scopus, Web of Science, and Google Scholar. Search concepts combined “cloud database” OR “distributed SQL” OR “NoSQL” with “high availability”, “replication”, “geo-replication”, “consistency”, “failover”, “recovery time”, “service-level objective”, “fault tolerance” or “data integrity”. Related terms for “Saudi”, “digital government”, “sovereignty” and “regulated data” supplied contextual screening rather than an artificial restriction to country-specific experiments.

The primary publication window was 2020–2025; earlier concepts were allowed only as background and were not used to pad the reference count. Eligible records were peer-reviewed journal articles or conference papers providing architecture, algorithmic analysis, controlled experimentation, systematic comparison, or relevant security and operations evidence. Exclusions included marketing descriptions, duplicate versions, unsupported vendor performance claims, opinion-only commentary, and work unrelated to databases or a direct dependency of data services. Selection was conceptual and relevance-based; no reproducible full-database export, independent dual-screening log, or complete flow count was maintained. Accordingly, this study does not report invented search-hit, screening, or inclusion totals and should not be described as a completed PRISMA systematic review.

For each selected paper, a qualitative extraction grid recorded system type, fault model, workload, replication and consistency assumptions, evaluation metrics, security boundary and external validity. We assessed methodological confidence by checking whether workloads and baselines were specified, faults were injected rather than merely

discussed, tail distributions were reported, conclusions matched tested conditions, and artifacts or configurations were described sufficiently for replication. The synthesis compares mechanisms and limitations rather than pooling numerical effect sizes from incompatible experiments. Publisher pages and DOI links were used for bibliographic cross-checking where accessible. Because this review does not independently rerun reported benchmarks, causal interpretations are limited to each study's stated environment.

Thematic Literature Review and Critical Analysis:-

Replication and consistency:-

Replication can improve service reachability but does not, on its own, guarantee coherent reads or durable acknowledged writes. Synchronous cross-zone replication can reduce acknowledged-transaction loss when a single zone fails. At the same time, asynchronous remote copies may permit lower normal-operation latency at the cost of a potentially nonzero recovery point. Ferreira et al. [1] demonstrate that the penalty of stronger settings depends heavily on engine, operation type, and geography. Camilleri et al. [5] present a causal-consistency approach; its weaker ordering contract may suit event dissemination but cannot automatically substitute for serializable state transitions in entitlement or payment records. A policy should therefore state the required invariant before selecting a consistency label.

Quorum placement is a practical architectural decision, not merely an implementation parameter. If voters share a power, routing, or identity dependency, nominally independent replicas can fail together. A regional quorum can allow local commits but may not survive an entire region outage without external coordination. Conversely, spreading a write quorum across distant sites may turn network degradation into higher transaction latency or reduced write availability. Hamrouni et al. [3] highlight the diversity of replication objectives; Lima et al. [4] show why richer retrieval choices can reduce cost while meeting particular causal-access needs. For essential public records, application-level idempotency, monotonic reads, and explicit conflict handling should accompany replica placement.

Architecture and operational placement:-.

A managed single-region multi-zone relational system may be preferable for strongly transactional portals because failover surfaces and maintenance responsibilities are relatively bounded. A dual-region primary–standby pattern adds disaster isolation while requiring continuous log-shipping supervision, promotion fencing, and failback planning. Active–active multiwriter designs can reduce geographic access distance but impose conflict-resolution, coordinated schema evolution, or distributed-commit complexity. A database mesh spanning separate providers may diversify procurement exposure, although differing consistency semantics, identity systems and network policies can make incident response harder rather than safer. Zhao et al. [12] show useful layering principles for private cloud platforms; Alahmad and Agarwal [7] underline the dependence of availability on placement decisions beyond the database layer.

Workload performance and safe tuning:-

Throughput measured during normal operations is an incomplete indicator because citizen-facing systems experience bursts, deadline-sensitive requests, and heterogeneous queries. Track 9thand 99th-percentile latencies, lock contention, rejected transactions, and replication backlog alongside median latency. Zhang et al. [2] explicitly address the risk of unsafe online database tuning under changing workloads. This supports conservative experimentation: parameter changes should have guardrails for saturation, rollback, and consistency checks. Capacity should be reserved for recovery, since a surviving replica may need to serve full traffic while simultaneously rebuilding another node. Resource placement and pricing analysis, including the graph approach of Khan and colleagues [27], makes that reserve visible as a design cost rather than an inefficiency.

Failure recovery and layered dependencies:-

Availability targets must name the failure being tolerated: a process crash, storage loss, zone isolation, region-wide disruption, faulty configuration, compromised administrator, or corrupted logical records. Replicated databases handle some infrastructure faults well but can rapidly replicate deleted tables or malicious updates. Bouizem et al. [9] experimentally distinguish request replication, standby, and retry mechanisms for serverless services, showing that each entails different performance and resource tradeoffs. While application functions differ from stateful databases, the evidence is relevant because apparent recovery may fail at the application boundary when retry semantics are unsafe. Database promotion must be coupled with transaction retry safety, connection pool refresh, and idempotent processing.

Storage integrity and protected recovery:-
Goswami et al. [8] review integrity protections for outsourced storage; Razaque et al. [16] examine privacy-preserving third-party auditing. Their contribution is complementary: backup existence is not proof that recovered records are authentic, complete, or usable. A robust arrangement combines transaction logs, point-in-time restoration, immutable isolated copies, periodic restore rehearsals, and cryptographic integrity verification. Encryption keys and the identity plane must remain accessible during declared incidents without giving recovery operators unrestricted routine access. Fugkeaw et al. [6] propose a strategy for fine-grained encrypted analytical access; however, before including advanced cryptographic designs in latency-sensitive transactional paths, benchmark them for operational overhead and key-recovery constraints.

Security, access and observability:-
Database high availability fails its public mission if an attacker can continuously reach a service with excessive privileges. Cloud attack surveys [13], [14] and risk-sensitive access models [15] reinforce layered authentication, privilege separation and contextual authorization. Yan et al. [11], Zhang et al. [17] and Huang and Yi [28] explore more elaborate cryptographic or ledger-assisted controls, although implementation complexity and key dependencies can introduce new outage modes. Sefati et al. [18] and Zhang et al. [10] address cloud anomaly detection from differing application and process perspectives. These approaches should complement, rather than replace, deterministic monitoring of replication lag, failed elections, deadlocks, certificate expiry, and unauthorized configuration changes.

Public-sector governance:-
Government systems typically exchange data across agency boundaries, making contractual and ownership interfaces part of resilience engineering. Classification should accompany each primary database, replica, cache, backup, snapshot, log export, and analytics feed. A recovery design cannot assume that copying restricted data to any technically reachable region is authorized. Kumar et al. [19] address secure cloud storage in smart-city settings, while Akhtar et al. [26] assess cloud trust through compliance and feedback signals. These contributions motivate governance measurement, but neither replaces current Saudi law, sector directives, procurement requirements, or agency-specific risk acceptance. Location approval must be evaluated against actual data flows, including automated failover destinations.

Transaction boundaries and interagency exchange:-
A common architectural mistake is to locate resilience entirely within the persistence layer while overlooking workflow semantics. A government application might update a permit database, publish a notification, charge a fee, and amend an external registry. Restoring one database does not guarantee that these actions form a coherent business transaction. Architects should distinguish events that may be retried harmlessly from irreversible actions that require deduplication keys, reconciliation ledgers, or compensating workflows. An outbox pattern can align local database commits with later event publication, but the consumer must still tolerate repeated delivery and delayed ordering. These safeguards matter most after failover, when clients may not know whether their most recent submission was accepted. A user-friendly portal should expose an authoritative transaction identifier and a means to query final status rather than inviting duplicate submissions.

Read scaling and authoritative state:-
Read replicas can improve throughput for public search and reporting services, but not every read may tolerate lag. For an application displaying a current license status, using a stale replica could confuse the user even if the underlying primary remains correct. A design can route confirmed state transitions and legal decisions to authoritative reads while allocating static reference information or historical aggregate queries to asynchronously updated nodes. The trade-off should be documented per query class, because blanket assurances that every replica is equivalent conceal actual consistency conditions. Cache invalidation deserves equal attention: failover without refreshing session state or entitlement caches may perpetuate stale decisions after database recovery. Read freshness can be assessed using known writes and end-to-end probes rather than relying solely on time-based replica statistics.

Table 1. Architecture choices by government transaction and fault tolerance requirements

Deployment pattern	Best-fit use case	Key correctness / recovery risk	Required validation
Managed multi-zone quorum	High-integrity registry, licensing and payment transactions	Shared control plane or degraded write quorum	Zone outage, quorum loss, committed-write audit

Regional primary with warm standby	Critical services needing region disaster tolerance	Asynchronous loss window; unsafe promotion	Region isolation, fencing and failback
Multiwriter / geo-distributed	Low-latency distributed access with explicit reconciliation	Conflicts, consensus delays and schema divergence	Partition test, invariant checks, p99 commits
Backup and point-in-time recovery	Lower-criticality services and malicious deletion protection	Restoration delay and missing credentials	Clean-room restore with data validation

Proposed Architecture and Evaluation Framework:-

The proposed design in Figure 1 separates citizen-facing service continuity from database transaction authority. A protected access layer verifies callers and routes eligible sessions to application services. Application services connect through a controlled database proxy to a primary write quorum across permitted fault domains. An independently managed read pool is used only where staleness tolerance is explicitly declared. A separately protected log and backup path preserves restoration options after correlated logical damage. Monitoring gathers both service-level outcomes and database internals; policy gates constrain promotion to approved locations and require immutable recording of who initiated recovery and why.

This architecture is a selectable pattern, not an instruction to deploy every component everywhere. A low-criticality information site may use managed backups and a documented restoration period. A high-criticality licensing or payment service may require same-region zonal quorum, warm secondary-region capability, and carefully bounded failover. Critical infrastructure integrations may demand local autonomy even when the central data service is unreachable. Table 1 links these patterns to operational choices. The central design rule is to prohibit an availability mechanism from silently relaxing a stronger correctness, privacy, or jurisdictional constraint.

Performance evaluation should use a preregistered workload set representing account lookup, high-contention record updates, payment-like atomic transitions, batch reconciliation, analytic read bursts, and peak concurrent login flows. Before failure experiments, establish baseline distributions during normal periods and planned maintenance. Then inject one disturbance at a time: database process termination, zone isolation, leader loss, network partition, storage saturation, certificate failure, bad deployment, and deliberate replica corruption. Include compound scenarios because apparently independent layers may share a control-plane dependency. Record the recovery decision, configuration revision, and evidence that surviving nodes retained a valid transaction history.

Figure 2 defines an evaluation loop that distinguishes detection time, decision time, promotion time, and verification time. Recovery time objective (RTO) is the elapsed period until business-valid service resumes, not the moment a cluster reports a new leader. Recovery point objective (RPO) measures lost committed business data using an external transaction ledger, not merely replica lag displayed before an incident. Error budgets must include invalid responses, inconsistent acknowledgments, and forbidden access; a healthy HTTP status with a wrong entitlement record should not count as success. Report uncertainty, repetitions and failure-injection boundaries rather than presenting one attractive best-case recovery duration.

To make experiments comparable, each workload should specify the generated record distribution, percentage of updates, concurrency ramp, transaction size, connection limits, and steady-state duration. An external independent journal of submitted operations should capture request IDs and expected outcomes. During faults, measurement must continue from client probes located outside the affected cloud environment. Otherwise, an internal dashboard may report successful database recovery while users still cannot complete transactions. Trials should repeat across different periods, preserve software versions, and document the configuration used for each replica placement. An agreed statistical analysis can compare distributions and confidence intervals rather than reporting misleading differences from a single execution. Baselines should include planned upgrades and maintenance, not only clean startup conditions.

A further design decision concerns promotion authority. Automated leader election can react rapidly to failure, but it should be impossible for two partitions to accept conflicting authoritative writes. Promotion requires durable fencing of an obsolete leader or a consensus protocol with an appropriate voting configuration. Operators should distinguish evidence of a lost quorum from temporary monitoring blind spots. Read-only fallback may be preferable to unsafe writes during some partitions, especially for official registries. This is a substantive service policy, not a technical defect: temporary restriction of amendments can protect legal correctness until authority is restored. Citizens should

receive accurate status messages explaining delayed processing, while audit logs preserve the incident's full chronology.

Figure 1. Policy-constrained high-availability database reference architecture

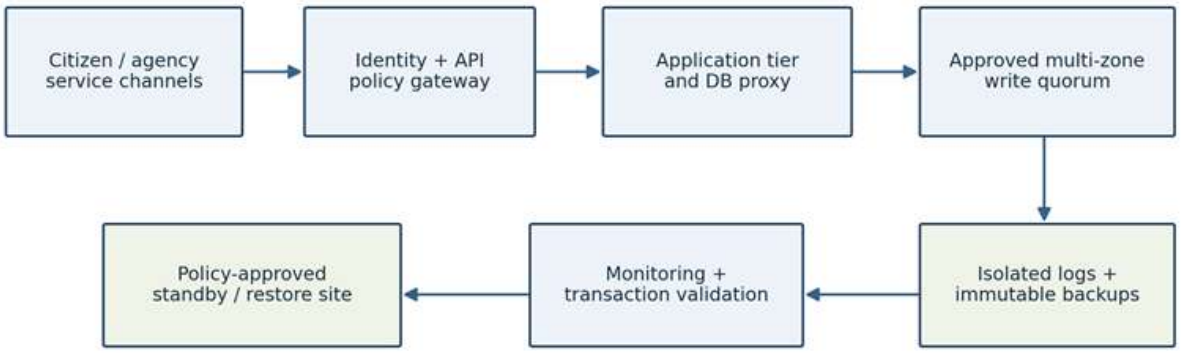


Figure 1. Original conceptual design; arrows represent authorized service, control, and recovery dependencies.

Figure 2. Reproducible failure-injection and service-validity evaluation loop

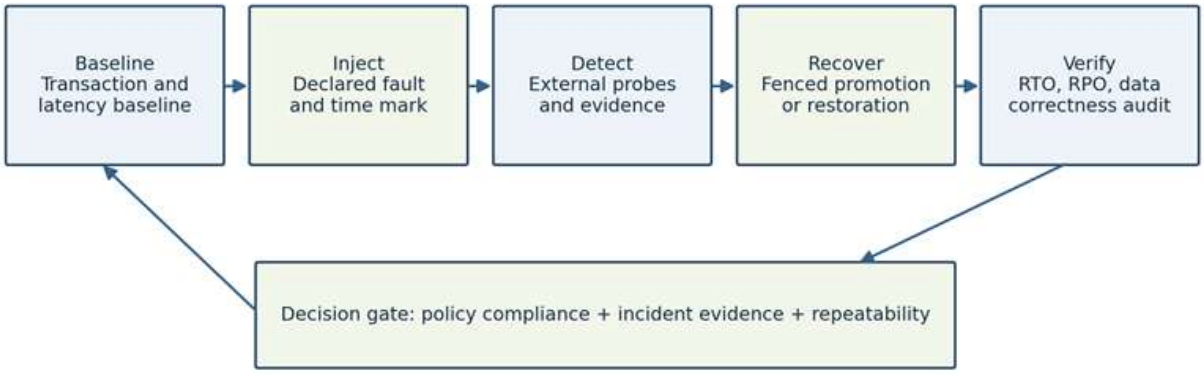


Figure 2. Original measurement flow; verify externally observed correct service availability after every fault.

Table 2. Evidence and benchmark indicators for high-availability service assurance

Evaluation dimension	Indicator / verification method	Interpretation boundary
Citizen-facing continuity	Successful business transactions / attempted transactions, p95/p99 latency	Exclude incorrect or unauthorized success responses
Committed-data safety	Externally reconciled acknowledged writes, observed RPO	Separate committed loss from stale display
Recovery effectiveness	Business-valid RTO, detection/promotion/validation intervals	Do not stop clock at new leader election

Security and location	Access logs, approved replica destinations, key retrieval tests	Promotion must remain within authorized boundaries
Operational sustainability	Failover CPU headroom, restore exercise success, total recovery cost	Test peak-load and compound incident conditions

Findings Synthesis:-

Five evidence-based findings emerge. First, fault isolation works only when replicas, control services, and recovery credentials don't share the same critical dependency. Second, stronger consistency guarantees are best allocated to transactions requiring authoritative state; applying them uniformly can impose unnecessary delay. Third, replication and backup address different threats: the former prioritizes rapid continuity, whereas the latter can recover from propagated logical faults. Fourth, cost-optimal placement under normal demand may be underprovisioned for a sustained incident, making recovery headroom an explicit requirement. Fifth, government assurance depends on reproducible evidence connecting technical test outcomes with approved data movement and service accountability.

These findings do not establish numerical superiority of a named vendor or database engine. Ferreira et al. [1] test selected NoSQL products and configurations, not every managed service available to Saudi agencies. Bouizem et al. [9] examine application-level mechanisms that may not transfer directly to database commit paths. Conversely, experimental detail in these studies is more informative for architecture choice than broad claims that multi-cloud is inherently resilient. The appropriate inference is conditional: the configuration is defensible when its observed behavior meets a particular workload's constraints under declared failure assumptions.

A sixth interpretive finding concerns measurement. Availability, durability, and consistency are related but not interchangeable constructs, so a composite dashboard should show each independently. The service might remain accessible during a network partition but refuse writes to protect consistency; another deployment might accept writes quickly and reconcile divergent states later. Neither behavior is universally preferable without a business contract. A seventh finding is that evidence maturity matters: studies that report controlled failure injection and transaction verification should weigh more heavily than generic architecture proposals without reproducible workloads. This weighting does not disparage conceptual research; it clarifies which claims are design hypotheses and which have empirical support in a defined setting.

Discussion:-

High availability is often advertised as a percentage while implemented as a chain of engineering decisions. A percentage cannot convey whether a database returned stale information, lost a final acknowledged update, or placed a recovery copy outside an approved processing boundary. Therefore, the meaningful evaluation unit should be the government transaction, with service availability assessed alongside data correctness and recovery authorization. Distributed transactions complicate this unit because a request may commit locally but fail before an external service receives confirmation. Idempotent replay, reconciled message delivery, and explicit transaction status become essential compensating mechanisms.

The most consequential trade-off is not simply active–active versus active–passive. It is allocating failure tolerance across a finite latency, staffing, and compliance budget. Active–active designs may benefit geographically distributed reads but require careful handling of concurrent updates; regional standby is easier to reason about yet introduces promotion and consistency-window hazards. Runtime placement studies [7], [21], [23] illustrate how scheduling affects availability or resource use, though they are not direct evidence that a given database topology will meet a statutory service obligation. Cross-discipline transfer must be qualified, not treated as a database benchmark.

Security intensifies the trade-off. Continuous availability can create a permanent attack surface; overly centralized keys, identity services, and policy engines can create correlated outages. Studies on cloud threat modeling [13], [14], [25], fine-grained access [15], [20] and encrypted cloud analytics [6] indicate that protective measures themselves have operating dependencies. Consequently, protection should be staged with emergency access controls, tested credential rotation, and separation between day-to-day operators and recovery authorization. A plausible strategy is to automate low-risk health routing while imposing policy gates on changes that alter geography, data authority, or encryption-key custody.

Government adoption adds organizational questions insufficiently visible in architecture diagrams. Procurement contracts must define fault domains and testing rights; audit teams need records of configuration changes and restoration exercises; service owners must accept the limits of degraded operation. Resource analysis [22], [27] raises a further question: is spare capacity economically justified by avoided disruption and preserved public trust? Compute invoices alone cannot answer it. It requires a decision record that explains essentiality, maximum tolerable interruption, recovery investment, and the residual uncertainty after testing.

Viewing resilience as an organizational capability helps explain why technically similar clusters can have different recovery outcomes. One team may recognize a replica lag anomaly, isolate a bad release and restore authoritative traffic quickly; another may have nominal redundancy but unclear escalation paths and untested permissions. Runbooks need decision criteria, not just command lists: when to suppress automatic promotion, when to invoke incident command, how to prove last committed state, and who may reopen traffic after checks pass. Exercising these judgments reveals hidden assumptions that static diagrams cannot capture. Such rehearsals also have an ethical dimension. Public services should avoid disrupting real citizen transactions merely to demonstrate an aggressive experiment; fault injection should be staged within controlled environments before carefully approved production exercises.

Another boundary concerns analytical workloads. Operational databases should not be overloaded with long-running cross-agency reporting queries during peak service windows. Read-only replicas, change-data-capture pipelines and properly governed analytical warehouses can absorb legitimate demand, although their freshness and access guarantees differ from the system of record. Fugkeaw and colleagues [6] illuminate how secure analytical access is possible, while cloud ETL optimization research [29] emphasizes that movement and transformation have distinct performance costs. Data provenance is essential: an analytical snapshot should identify its source commit watermark, classification, and retention authority. Without this context, dashboards may appear available while decisions rest on outdated or incompletely reconciled records.

Research Gaps and Future Research:-

The first gap is the scarcity of public, reproducible benchmarks that reproduce Saudi government service mixes and location constraints. Future studies should compare managed relational, distributed SQL, and NoSQL deployments with identical synthetic citizen transactions, documented read/write mixes, and fault schedules. Metrics should include correct completed transactions per second, p99 latency, zero-loss verification, recovery time, and administrative effort. Researchers should release anonymized configurations where lawful, while excluding operational secrets.

The second gap concerns correlated faults. Published evaluations often isolate a single failed process or network link, but real incidents can combine bad configuration, shared authentication dependencies, and regional connectivity problems. Experiments should test quorum loss together with certificate expiry, key-service impairment, or malicious deletion. A third gap concerns failback: a promoted secondary is not a finished incident until original service authority, logs and governance controls have been safely reconciled. Studies should measure split-brain prevention and reconciliation cost after restoration.

Fourth, the economics of sovereignty-aware resilience require sharper modeling. Future work could estimate the incremental price of approved local replication against the reduction in loss exposure and latency. Khan et al. [27] offer a useful direction for placement modeling, while Dinesh and Devi [29] illustrate the distinct optimization requirements of cloud analytical pipelines. Fifth, human operations merit longitudinal assessment: recurring exercises, staff turnover, and incident rehearsal quality may explain more operational variance than an incremental infrastructure feature. These hypotheses should be tested rather than asserted as established Saudi outcomes.

A sixth gap is the absence of agreed public-sector evaluation datasets that preserve realistic relationships without exposing citizens. Synthetic datasets should model correlated household records, address changes, licensing renewals and fee events, including exceptional cases that stress unique constraints. Another gap is the limited treatment of control-plane disruption during performance tests. A failover study that assumes flawless identity federation, secret retrieval, and DNS resolution gives agencies only a partial picture of recoverability. A coordinated research program could publish open fault scripts, anonymized metrics and reproducible deployment descriptions suitable for independent academic scrutiny. The objective would be to establish bounded evidence, not to declare one cloud architecture permanently superior.

Practical, Managerial and Policy Implications:-

Public-sector technology leaders should create service criticality tiers before selecting cloud database products. Each tier needs a documented correctness contract, business-valid RTO, committed-data RPO, location rules and minimum recovery evidence. Architecture reviews should distinguish single-node redundancy, availability-zone independence, regional disaster tolerance, and logical corruption recovery. A vendor SLA may support procurement but cannot replace tests performed through the actual government application, identity system, and network paths.

Database engineering teams ought to maintain versioned dependency maps; explicit leader-fencing procedures; verified replication-lag alerts; tested connection and retry behavior; point-in-time recovery exercises; and periodic reports of invalid or delayed transactions. Financial owners should compare full lifecycle cost, including reserved recovery capacity, logs, bandwidth, backup egress, security verification, incident staffing, and the price of switching providers. Risk committees should require exception approval when a proposal trades guaranteed correctness or permitted data placement for faster automatic failover.

Policy implementation should be evidence-led and revisited as official guidance evolves. The receiving organization must map each dataset and potential recovery destination to applicable Saudi rules, contract terms, and sector-specific restrictions using current primary legal texts. An approved architecture record should state who owns personal data, who can decrypt it, who can authorize emergency promotion, and how copies are deleted after temporary recovery. These steps are governance recommendations, not claims that any particular configuration has already received regulatory approval.

A practical accountability matrix should connect every service dependency to an owner and a measurable test. The application team owns business-level probes and idempotency, database administrators own replica health and restore integrity, cybersecurity owns access and key controls, and continuity leadership owns incident declaration and cross-agency communication. Suppliers should provide timely architecture change notifications, incident evidence, and export procedures, but the agency must still verify that contractual promises operate in its own integrated environment. The governance board should reassess tier assignments after material changes in usage, data sensitivity, service integrations, or the national policy landscape. A service that originally handled noncritical information can become essential after other public workflows begin relying on it.

Limitations:-

This article synthesizes heterogeneous studies without a pooled effect estimate, exhaustive systematic search audit, or independent reproduction of experiments. The referenced works use different workloads, platforms, and measurement conventions; results cannot be converted into uniform recovery-time expectations. Several supporting studies address adjacent cloud security, serverless scheduling, or data pipelines rather than core transactional database engines [24], [30], and their transferability is explicitly limited. No Saudi government production logs, procurement records, or agency interviews were available. The proposed architecture therefore remains a hypothesis-driven design aid, not an independently validated deployment blueprint or a statement of regulatory certification. Retest the framework whenever major dependencies change, because assurance is time-sensitive and previously validated recovery procedures may no longer reflect operational conditions or acceptable governance.

Conclusion:-

Saudi government database resilience should be engineered as correctness-preserving service continuity under explicitly declared failures. A useful architecture combines suitable replication semantics, controlled promotion, independent recoverability and traceable data governance; no individual redundancy feature establishes those outcomes alone. The literature supports tiered placement decisions and workload-specific experiments, while revealing important uncertainty about correlated failures, failback, human operation and the economics of approved geographic recovery. The framework presented here makes those uncertainties visible through distinct service metrics, fault scenarios, and governance evidence. Its practical value lies in enabling agencies to justify a design against their own transactions and obligations and to replace assumed availability with demonstrable, repeatable recovery.

References:-

1. Ferreira, S., Mendonça, J., Nogueira, B., Tiengo, W., & Andrade, E. (2024). Impacts of data consistency levels in cloud-based NoSQL for data-intensive applications. Journal of Cloud Computing, 13, 158. https://doi.org/10.1186/s13677-024-00716-7
2. Zhang, X., Wu, H., Li, Y., Tan, J., Li, F., & Cui, B. (2022). Towards dynamic and safe configuration tuning for cloud databases. Proceedings of SIGMOD, 631–645. https://doi.org/10.1145/3514221.3526176
3. Hamrouni, T., Mokadem, R., & Khelifa, A. (2023). Review on data replication strategies in single vs. interconnected cloud systems: Focus on data correlation-aware strategies. Concurrency and Computation: Practice and Experience, 35, e7758. https://doi.org/10.1002/cpe.7758
4. Lima, S., Araujo, F., Guerreiro, M. O., Correia, J., Bento, A., & Barbosa, R. (2023). Efficient causal access in geo-replicated storage systems. Journal of Grid Computing, 21, 8. https://doi.org/10.1007/s10723-022-09640-z
5. Camilleri, C., Vella, J. G., &Nezval, V. (2024). Horizontally scalable implementation of a distributed DBMS delivering causal consistency via the actor model. Electronics, 13, 3367. https://doi.org/10.3390/electronics13173367
6. Fugkeaw, S., Suksai, P., & Hak, L. (2024). SSF-CDW: Achieving scalable, secure, and fast OLAP query for encrypted cloud data warehouse. Journal of Cloud Computing, 13, 129. https://doi.org/10.1186/s13677-024-00692-y
7. Alahmad, Y., & Agarwal, A. (2024). Multiple objectives dynamic VM placement for application service availability in cloud networks. Journal of Cloud Computing, 13, 46. https://doi.org/10.1186/s13677-024-00610-2
8. Goswami, P., Faujdar, N., Debnath, S., Khan, A. K., & Singh, G. (2024). Investigation on storage level data integrity strategies in cloud computing: Classification, security obstructions, challenges and vulnerability. Journal of Cloud Computing, 13, 45. https://doi.org/10.1186/s13677-024-00605-z
9. Bouizem, Y., Dib, D., Parlavantzas, N., & Morin, C. (2023). Integrating request replication into FaaS platforms: An experimental evaluation. Journal of Cloud Computing, 12, 94. https://doi.org/10.1186/s13677-023-00457-z
10. Zhang, X., Cui, L., Shen, W., Zeng, J., Du, L., He, H., & Cheng, L. (2023). File processing security detection in multi-cloud environments: A process mining approach. Journal of Cloud Computing, 12, 100. https://doi.org/10.1186/s13677-023-00474-y
11. Yan, L., Ge, L., Wang, Z., Zhang, G., Xu, J., & Hu, Z. (2023). Access control scheme based on blockchain and attribute-based searchable encryption in cloud environment. Journal of Cloud Computing, 12, 61. https://doi.org/10.1186/s13677-023-00444-4
12. Zhao, L., Hu, G., & Xu, Y. (2024). Educational resource private cloud platform based on OpenStack. Computers, 13, 241. https://doi.org/10.3390/computers13090241
13. Dawood, M., Tu, S., Xiao, C., Alasmary, H., Waqas, M., & Rehman, S. U. (2023). Cyberattacks and security of cloud computing: A complete guideline. Symmetry, 15, 1981. https://doi.org/10.3390/sym15111981
14. Abdulsalam, Y. S., &Hedabou, M. (2022). Security and privacy in cloud computing: Technical review. Future Internet, 14, 11. https://doi.org/10.3390/fi14010011
15. Alharbe, N., Aljohani, A., Rakrouki, M. A., & Khayyat, M. (2023). An access control model based on system security risk for dynamic sensitive data storage in the cloud. Applied Sciences, 13, 3187. https://doi.org/10.3390/app13053187
16. Razaque, A., Frej, M. B. H., Alotaibi, B., & Alotaibi, M. (2021). Privacy preservation models for third-party auditor over cloud computing: A survey. Electronics, 10, 2721. https://doi.org/10.3390/electronics10212721
17. Zhang, J., Chen, A., & Zhang, P. (2023). Provably secure data access control protocol for cloud computing. Symmetry, 15, 2111. https://doi.org/10.3390/sym15122111
18. Sefati, S. S., Arasteh, B., Fratu, O., et al. (2025). SSLA: A semi-supervised framework for real-time injection detection and anomaly monitoring in cloud-based web applications with real-world implementation and evaluation. Journal of Cloud Computing, 14, 38. https://doi.org/10.1186/s13677-025-00765-6
19. Kumar, A., Khan, S. B., Pandey, S. K., Shankar, A., Maple, C., Mashat, A., &Malibari, A. A. (2023). Development of a cloud-assisted classification technique for the preservation of secure data storage in smart cities. Journal of Cloud Computing, 12, 92. https://doi.org/10.1186/s13677-023-00469-9
20. Tao: Improving resource utilization while guaranteeing SLO in multi-tenant relational database-as-a-service. (2024). Proceedings of the ACM on Management of Data, 2(4). https://doi.org/10.1145/3677141
21. Nkenyereye, L., Lee, B. G., & Chung, W. Y. (2025). Functionality-aware offloading technique for scheduling containerized edge applications in IoT edge computing. Journal of Cloud Computing, 14, 13. https://doi.org/10.1186/s13677-025-00737-w

22. Ghorbian, M., Ghobaei-Arani, M., & Esmaeili, L. (2025). An energy-conscious scheduling framework for serverless edge computing in IoT. Journal of Cloud Computing, 14, 52. https://doi.org/10.1186/s13677-025-00780-7
23. Ullah, I., Lim, H., Seok, Y., et al. (2023). Optimizing task offloading and resource allocation in edge-cloud networks: A DRL approach. Journal of Cloud Computing, 12. https://doi.org/10.1186/s13677-023-00461-3
24. Implementing a synchronization method between a relational and a non-relational database. (2023). Big Data and Cognitive Computing, 7, 153. https://doi.org/10.3390/bdcc7030153
25. Ni, K., Mondal, S. K., Kabir, H. M. D., Tan, T., & Dai, H.-N. (2024). Toward security quantification of serverless computing. Journal of Cloud Computing, 13, 140. https://doi.org/10.1186/s13677-024-00703-y
26. Akhtar, S. I., Rauf, A., Abbas, H., Amjad, M. F., & Batool, I. (2024). Compliance and feedback based model to measure cloud trustworthiness for hosting digital twins. Journal of Cloud Computing, 13, 132. https://doi.org/10.1186/s13677-024-00690-0
27. Khan, A. Q., Matskin, M., Prodan, R., Bussler, C., Roman, D., & Soylu, A. (2024). Cost modelling and optimisation for cloud: A graph-based approach. Journal of Cloud Computing, 13, 147. https://doi.org/10.1186/s13677-024-00709-6
28. Huang, J., & Yi, J. (2024). The key security management scheme of cloud storage based on blockchain and digital twins. Journal of Cloud Computing, 13, 15. https://doi.org/10.1186/s13677-023-00587-4
29. Dinesh, L., & Devi, K. G. (2024). An efficient hybrid optimization of ETL process in data warehouse of cloud architecture. Journal of Cloud Computing, 13, 12. https://doi.org/10.1186/s13677-023-00571-y
30. Testing very large database management systems: The case of SAP HANA. (2022). Datenbank-Spektrum. https://doi.org/10.1007/s13222-022-00426-x