The Eastasouth Journal of Information System and Computer Science
Vol. 2, No. 02, December 2024, pp. 411-416
ISSN: 3025-566X, DOI: 10.58812/esiscs.v2i02

Runtime-State and Queue-Processing Validation After Cloud Database Restoration

Surya Kiran Yelagam
Independent Researcher, McKinney, Texas, USA

Article Info

Article history:

Received December, 2024
Revised December, 2024
Accepted December, 2024

Keywords:

Application Recovery;
Database Restoration;
Idempotency;
Message Queues;
Recovery Acceptance

ABSTRACT

A restored database can accept connections while workers are still processing messages against the wrong application state. This paper proposes a recovery acceptance procedure that follows one authorized test transaction from application intake through queued work to its durable outcome. It identifies what each checkpoint can establish and when the evidence is insufficient to resume service. The procedure is a design proposal. Its error rate and effect on recovery time remain unmeasured.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Name: Surya Kiran Yelagam
Institution: Independent Researcher, McKinney, Texas, USA
Email: sy.papertrail@gmail.com

1. INTRODUCTION

A successful database probe can coexist with a stalled business transaction. A worker may retain a connection to the previous instance, or a queue may hold requests whose corresponding records disappeared when the database was rolled back. In both cases an operator can observe a responsive service before the workflow is safe to reopen.

This paper specifies the evidence required to reopen a selected workflow. A recovery team identifies the restore boundary, verifies application and worker state, and follows a controlled request to a durable business outcome. The contribution is a release decision that another operator can reconstruct from the exercise record. The scope is the tested workflow and the restore point used in that exercise.

2. RELATED WORK

Abadi describes why consistency requirements and workload characteristics matter when moving data management to the cloud [1]. Barker and colleagues evaluate the operational cost of live database migration, including query latency and interference with other workloads [2]. Those studies address database behavior. A recovery team also needs to establish what happened to operations that span the database and an independently retained queue.

Saltzer, Reed and Clark use duplicate suppression and delivery acknowledgment to illustrate the end-to-end argument [3]. An application knows what constitutes a duplicate business effect. The same distinction appears in the documented at-least-once behavior of Amazon SQS standard queues [4]. A repeated delivery can be legitimate even when a second committed business effect would be an error. The

acceptance procedure applies this distinction after a restore.

Recovery of the database itself is well studied. ARIES restores a consistent state by replaying a write-ahead log and undoing incomplete transactions [5], and cloud-native engines such as Amazon Aurora move much of that redo work into the storage layer [6]. These mechanisms establish that committed data is durable and consistent at the recovery point. They do not report whether a worker or a queue outside the database agrees with that point.

Work that spans components raises a separate question. Elnozahy and colleagues survey rollback-recovery protocols for message-passing systems and describe how a process restored to an earlier state can disagree with messages it has already sent or received [7]. Sagas divide a long-lived activity into steps with compensating actions, so partial work can be undone without a single enclosing transaction [8]. Helland argues that messages can be retried or reordered and that applications should be designed so that repeated processing is harmless [9]. Log-based messaging systems such as Kafka retain messages independently of the consumer's database and let a consumer read again from an earlier position [10], which is one way a queue and a restored database can diverge.

Contingency-planning guidance for federal information systems calls for recovery procedures to be tested and for the results to be documented [11]. The procedure proposed here supplies one concrete form for that record at the level of a single workflow.

3. RECOVERY CONTEXT AND EVIDENCE

Start with the database restore identifier, recovery point, application version, worker configuration and queue retention boundary. Record which components were restored together. A database reset to an earlier state can disagree with a queue or a downstream service that continued processing. Restrict the test to an approved environment and isolate integrations that could trigger an external action. Figure 1 shows the evidence sources that feed each stage of the procedure.

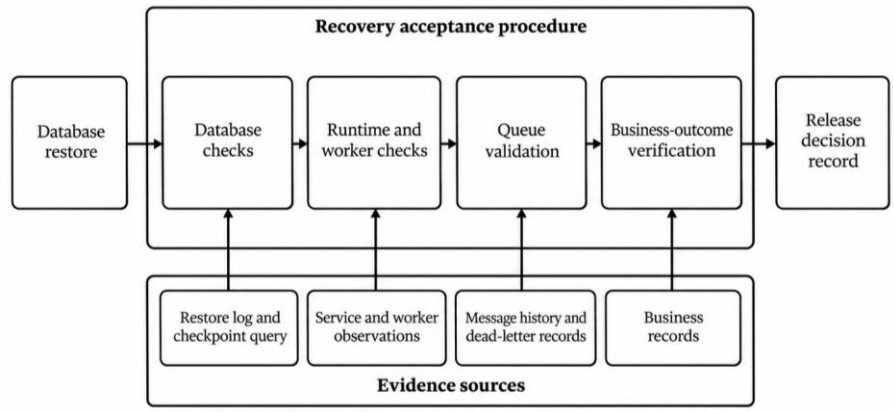


Figure 1. Evidence sources for a recovery decision

3.1 Establish the baseline

Choose a representative workflow and a stable business identifier. Before issuing work, record the expected state transition, the number of permitted effects and the observation window. Capture the initial record count for that identifier. If a consumer can write outside the restored database, include that destination in the evidence plan or explicitly exclude it from the scope of the release.

Verify that the application and workers reach the intended database instance under their ordinary identities. Compare a known checkpoint with the expected restore boundary. Confirm the schema and application version are compatible. A successful administrator connection cannot establish that a

worker using a different endpoint or identity will reach the same data.

3.2 *Record uncertainty at the point of observation*

For each check, record the expected value, observed value, evidence location and observation time. Distinguish a failed check from missing evidence. A missing trace could indicate a logging gap or a worker that never received the message. Both require investigation, but they call for different corrective actions.

4. RUNTIME AND QUEUE VALIDATION

Once the database checks pass, test the service path and its dependencies. A worker can report healthy while its credentials have expired or it is reading a stale endpoint. Compare the endpoint seen by the worker with the verified restore target, then trace a synthetic request through its ordinary handler. Figure 2 shows the order of the checks and the return path after a failure.

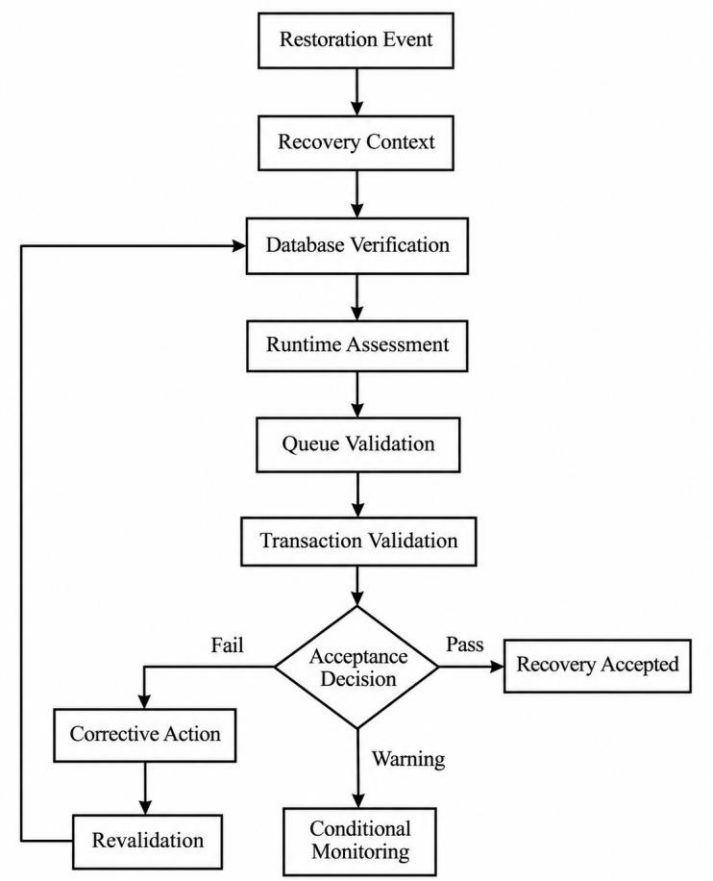


Figure 2. Sequence of checks and revalidation after a failure

Inspect message age, retry history and dead-letter disposition alongside the durable record. A queue can empty when a consumer rejects or discards a message. Conversely, an increasing queue depth may reflect an unrelated workload. Neither measurement alone establishes the result for the selected identifier. Figure 3 relates these runtime and queue observations to the application records.

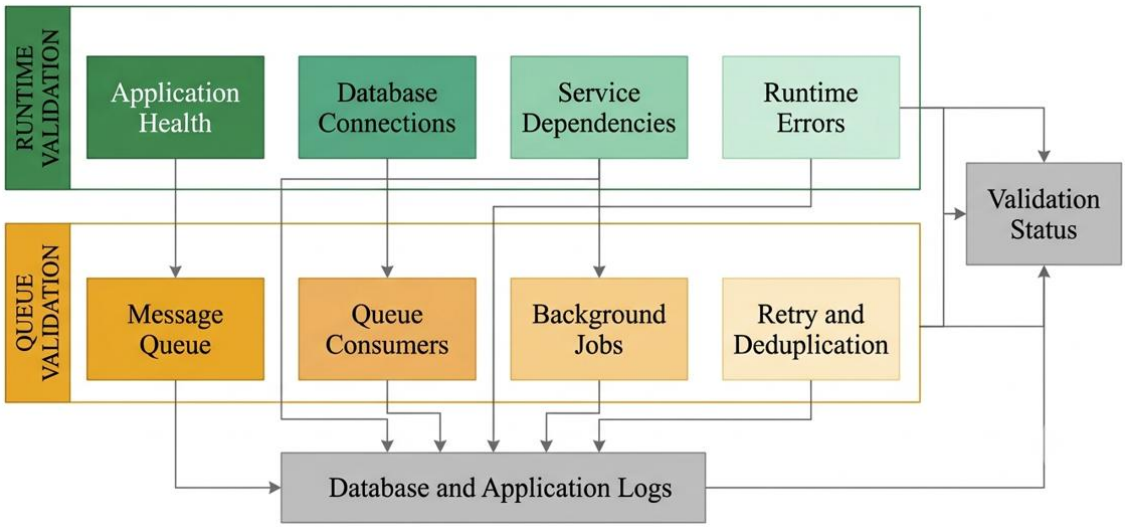


Figure 3. Runtime and queue observations correlated with application records

5. WORKED RECOVERY DECISION

Consider a synthetic request that creates one test reservation. Before the exercise, confirm that the reservation identifier has no existing entry. The expected terminal state is one confirmed reservation with one matching allocation entry. Use the same business identifier throughout the test, including any retries. This example illustrates the decision rule and contains no field measurements.

Suppose the worker commits the reservation, loses its acknowledgment and receives the request again. Two delivery attempts and one confirmed reservation can satisfy the check if exactly one allocation remains and the request reaches the defined terminal state. One reservation with two allocation entries fails. If there is no allocation record and no surviving worker trace, record the result as indeterminate. An empty queue does not resolve that uncertainty.

Evidence to Retain

Table 1. Evidence required for each recovery decision

Check	Observation	Evidence
Database	Restore point, schema and endpoint	Restore log and checkpoint query
Runtime	Worker identity and actual dependency target	Service and worker observations
Queue	Delivery attempts, age and final disposition	Message history and dead-letter records
Business effect	Observed terminal state and number of effects	Reservation and allocation records
Decision	Expected versus observed, exception owner and time	Recovery exercise record

Table 1 lists the evidence retained for each check. Release the tested workflow only when the observed result of every mandatory check matches its expected result and the evidence is recorded. A failed or indeterminate mandatory check keeps that workflow on hold. Monitoring may accompany release for a documented noncritical warning. It cannot substitute for a missing mandatory result.

6. INTERPRETATION AND LIMITATIONS

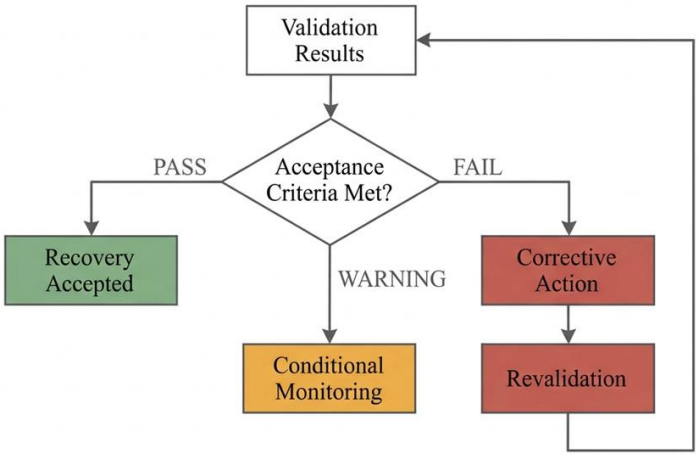


Figure 4. Decision paths for acceptance, monitoring and corrective action

Figure 4 summarizes the decision paths. After a failure, identify the earliest checkpoint at which the observations diverged from expectations. Correct that condition and repeat the affected steps with a fresh test identifier. Retain the original exercise record so that a successful retry cannot obscure the initial failure.

A passing result applies to one workflow, configuration and observation window. Delayed side effects, other message types and dependencies outside the exercise remain untested. Replaying selected in-flight work on either side of the restore boundary would help assess whether the procedure also detects stale, duplicated or lost pre-restore operations.

7. EVALUATION PLAN

A controlled comparison could run a database-only check and the proposed workflow check against the same recovery scenarios. Include healthy restores, stale worker connections, delayed redelivery and duplicate effects. Establish the expected outcome for each scenario before testing. Report false clearances, unresolved cases and time to a decision separately. No comparative measurements are available from this design study.

8. CONCLUSION

Recovery acceptance requires an observed business outcome tied to the restore point and the tested request. Recording the evidence and the decision boundary gives the next operator a basis for reopening that workflow or continuing recovery.

REFERENCES

[1] D. J. Abadi, "Data Management in the Cloud: Limitations and Opportunities," *IEEE Data Eng. Bull.*, vol. 32, no. 1, pp. 3–12, 2009, [Online]. Available: https://www.cs.umd.edu/~abadi/papers/abadi-cloud-ieee09.pdf

[2] S. K. Barker, Y. Chi, H. J. Moon, H. Hacigümüş, and P. J. Shenoy, "'Cut Me Some Slack': Latency-Aware Live Migration for Databases," in *Proceedings of the 15th International Conference on Extending Database Technology*, 2012, pp. 432–443. doi: 10.1145/2247596.2247647.

[3] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984, doi: 10.1145/357401.357402.

[4] Amazon Web Services, "Amazon SQS at-least-once delivery," Amazon Simple Queue Service Developer Guide. [Online]. Available: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/standard-queues-at-least-once-delivery.html

[5] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz, "ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging," *ACM Trans. Database Syst.*,

vol. 17, no. 1, pp. 94–162, 1992, doi: 10.1145/128765.128770.

[6] A. Verbitski *et al.*, "Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1041–1052. doi: 10.1145/3035918.3056101.

[7] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems," *ACM Comput. Surv.*, vol. 34, no. 3, pp. 375–408, 2002, doi: 10.1145/568522.568525.

[8] H. Garcia-Molina and K. Salem, "Sagas," in *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, 1987, pp. 249–259. doi: 10.1145/38713.38742.

[9] P. Helland, "Idempotence is not a medical condition," *Commun. ACM*, vol. 55, no. 5, pp. 56–65, 2012, doi: 10.1145/2160718.2160734.

[10] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB '11)*, Athens, Greece, 2011, pp. 1–7.

[11] M. Swanson, P. Bowen, A. W. Phillips, D. Gallup, and D. Lynes, *Contingency Planning Guide for Federal Information Systems*. NIST Special Publication 800-34 Revision 1, 2010. doi: 10.6028/NIST.SP.800-34r1.