AI-assisted migration to Oracle AI Database 26ai

Elena Marković Adrian Wójcik
Maya Srinivasan Julian Hartmann

Abstract

Database migration combines software compatibility, data correctness, performance engineering, and operational recovery. Generative AI can assist with interpreting inventories, drafting remediations, and assembling tests, but a plausible recommendation is not evidence that a database is ready to move. This paper proposes an evidence-driven framework for AI-assisted migration to Oracle AI Database 26ai. It separates probabilistic assistance from deterministic execution, links recommendations to versioned sources, and requires independent validation before an operational transition.

The framework covers discovery, route selection, bounded transformation, semantic testing, performance qualification, and cutover. Oracle-to-Oracle upgrades form the primary scope; heterogeneous migrations are treated as a separate transformation problem. A reproducible evaluation protocol compares conventional tooling, unconstrained assistance, and assistance with evidence gates. The contribution is a methodology and reproducible evaluation protocol for studying migration quality and engineering effort.

Keywords: database migration · Oracle 26ai · generative AI · semantic equivalence · verification · operational readiness

1 Introduction

An upgrade can finish successfully while an application still fails: a driver may be incompatible, a plan may regress, or a batch job may depend on altered behavior. The object of migration engineering is a working service under explicit operating constraints, rather than a completed tool invocation. Oracle's upgrade documentation provides the execution foundation; AI assistance should operate around that foundation. [1]

Our central research question is: *can evidence-linked AI assistance reduce engineering effort while preserving independently verified migration quality?* This question requires measurement. We specify a method and a study design rather than report an unobserved outcome.

1.1 Contributions

The paper contributes a proposed evidence contract, a migration decision model, a layered validation strategy, and a comparative evaluation protocol. The contributions integrate operational reasoning with the cited literature into a consistent migration methodology.

1.2 Research foundations

Three adjacent research directions motivate the proposal. Retrieval-augmented generation provides a way to condition a response on external evidence [9]. Constrained decoding addresses admissible SQL syntax [7]. Test-suite evaluation examines behavior across database instances [8]. These contributions address different parts of the problem; none establishes that an enterprise migration is operationally safe.

Our proposed synthesis places assistance inside an engineering process with explicit observations, accountable decisions, and independent acceptance gates. A recommendation remains provisional until the relevant evidence supports it. This distinction matters because a system can produce a correct explanation of a documented feature while applying that explanation to the wrong release, workload, or dependency. The framework therefore treats applicability and verification as separate requirements.

2 Define the migration before automating it

"Migration to 26ai" may describe three different changes: upgrading Oracle database software, relocating an Oracle workload, or translating an application from another engine. Combining them in one undifferentiated plan hides both compatibility constraints and the required evidence. Every project should first declare its source, target, movement method, application boundary, and recovery objective.

2.1 Route classification

Route	Primary change	Required evidence
Oracle upgrade	Release and architecture	Supported path, prechecks, application and workload tests
Oracle relocation	Platform or service	Transfer, synchronization, endpoint and recovery tests
Heterogeneous migration	Types, SQL, routines	Mappings, semantic tests, application adaptation

Table 1: Route families and their distinct proof obligations.

The 26ai Upgrade Guide lists Oracle 19c and 21c as direct-upgrade source releases. Earlier releases require a supported intermediate upgrade path for an in-place upgrade; logical export/import is a different route with its own prerequisites. Eligibility alone does not certify an operating system, patch combination, or third-party application. [1]

2.2 Use supported execution mechanisms

Oracle identifies AutoUpgrade and Replay Upgrade as supported upgrade methods for 26ai and desupports earlier manual upgrade mechanisms, including DBUA. An AI assistant should not reconstruct an old manual upgrade recipe from training data. It should identify the documented method and produce reviewable inputs for the selected tool. [2]

Relocation is a separate decision. Zero Downtime Migration documents physical and logical methods, with eligibility determined by source, target, platform, and service. The product name is not a contractual guarantee of uninterrupted writes. Application draining, replication convergence, and connection switching must be measured in rehearsal. [4]

2.3 Boundary of this proposal

The main scenario is a supported Oracle-to-Oracle route with an application regression corpus. Heterogeneous conversion exposes additional semantic risks; AutoUpgrade does not translate other database engines. AI features in the destination are optional modernization work. Enabling vector search or changing an application's data model is not a prerequisite for moving an existing workload.

The route record must state excluded systems, unavailable dependencies, and unverified certifications. Unknowns remain open findings; the assistant must not silently substitute a guessed answer.

2.4 Route decision example

Consider a hypothetical estate with one transactional database, two reporting consumers, and an overnight export. Its route cannot be selected from database size alone. The decision record must identify which consumers require a consistent view, which clients can reconnect, and how a delayed export affects the next business cycle. A route that fits the transfer window may still fail the service contract if a dependent application cannot tolerate the transition.

For each candidate, write down the assumption that would make it unacceptable. Examples include an unavailable platform certification, an unresolved application dependency, or a recovery procedure that cannot meet the agreed deadline. Eliminate a route only with an explicit reason. Compare the remaining candidates by measured rehearsal evidence, engineering effort, operational constraints, and licensed capabilities. Record the decision owner and the date on which the assumptions must be rechecked.

This example defines a planning exercise, not an observed customer migration. The purpose is to make the decision reproducible when a source patch, target service, or workload requirement changes.

3 Build an evidence-bearing inventory

Discovery creates the boundary within which assistance is useful. The proposed inventory is a dependency graph connecting database objects, application entry points, integration jobs, clients, and operational procedures. Nodes carry a collection timestamp and source identifier. Edges represent observed references or declared dependencies; inferred edges remain hypotheses until confirmed.

3.1 Minimum discovery package

Collect the exact database version and release update, architecture, character sets, time-zone configuration, installed options, objects, privileges, external dependencies, scheduler jobs, and recovery topology. At the application layer, record drivers, runtimes, connection pools, authentication, session settings, and deployment ownership. Capture workload shape across normal, peak, reporting, and batch periods.

Use read-only collection with least privilege. Prefer schema definitions and redacted metadata over business rows. An identifier can still be sensitive: object names may reveal a customer or commercial process. Record what was collected, where it is stored, and whether it can enter the chosen model's processing boundary.

3.2 Recommendation contract

Every proposed action should be a structured record. This makes uncertainty visible and permits rejection without interpreting a long narrative. The contract below is a design contribution, not an Oracle interface.

```
finding_id: F-017
object: APP.PAYMENT_POSTING
observation: dependency requires review
source: inventory/objects.json#payment_posting
source_digest: <SHA-256 of collected artifact>
target: <exact release update and platform>
proposal: <bounded change or investigation>
evidence: [<versioned document reference>]
validation: [<independent test identifiers>]
owner: <accountable engineer>
state: proposed
```

3.3 Source authority and freshness

Retrieve release-specific vendor documentation before proposing a compatibility change. Attach the location, retrieval date, applicable release, and relevant finding. Retrieval-augmented generation conditions generation on external material; it does not establish the truth of a conclusion or guarantee that a passage applies. [9]

For driver decisions, inspect the actual JDK, JDBC artifact, and database combination rather than assuming the newest client is a drop-in replacement. Oracle publishes interoperability guidance, but the complete application stack may impose further restrictions. [6]

Acceptance rule. A recommendation without a traceable observation, applicable source, owner, and validation plan is incomplete. It may enter investigation; it may not become an approved migration action.

3.4 Inventory completeness and expiry

Completeness is relative to a declared boundary. Counting database objects cannot reveal whether an external application is missing from the inventory. Establish a list of expected application owners and request their confirmation of the dependency graph. Track unresolved ownership as a finding. An absent response must remain visible rather than be interpreted as an absence of dependencies.

Attach a freshness policy to each artifact. A schema export becomes stale after relevant DDL; a client inventory becomes stale after a runtime deployment; a workload sample may be unrepresentative after a business-process change. Recollection should be triggered by these events as well as by elapsed time. Preserve the older artifact so that a reviewer can determine which recommendation depended on which estate state.

Before a rehearsal, compare the current inventory digest with the approved planning bundle. A difference does not automatically invalidate the entire plan. It requires a scoped impact review that identifies affected findings, tests, and approvals. The resulting decision belongs in the evidence record.

4 Separate suggestion from authority

The proposed architecture has three planes: evidence collection, assisted reasoning, and controlled execution. Only the execution plane can mutate an environment. A model may draft a plan or patch, but cannot grant its own permissions, mark its own validation as passed, or promote its output to production.

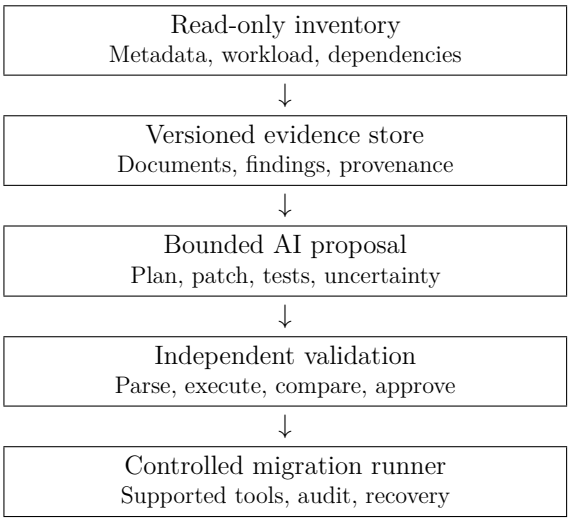


Figure 1: Proposed control flow. Failed gates return to investigation.

4.1 Retrieval and bounded generation

Resolve source and target versions before selecting documentation. Limit each task to a defined unit, such as one finding or procedure. Require the assistant to state assumptions and abstain when a necessary artifact is missing. Constrained SQL generation research motivates syntax checks, but valid syntax alone does not prove that a translated statement preserves behavior. [7]

4.2 Enforcement outside the model

Apply permissions, command allowlists, sandbox boundaries, and artifact signing in conventional software. Treat comments in schemas, retrieved pages, and tool output as untrusted input. A malicious comment must not expand the assistant's privileges or redirect a report to an external destination. Tests and policy checks execute independently of the text proposing a change.

4.3 Reproducibility record

Persist model identifier, inference settings, prompt-template revision, document and input digests, generated diff, tool versions, and validation outcomes. Store secrets outside this record. Repeated runs with changed inputs receive a new evidence bundle. This supports investigation even when model output is not bit-for-bit reproducible.

4.4 Explicit failure behavior

A task request should name permitted inputs, expected output fields, a resource budget, and the conditions for abstention. For example, a model may be asked to explain one precheck finding using an approved document set and to return an investigation plan. A missing document, an ambiguous object reference, or contradictory release information should produce a structured unresolved result rather than an improvised command.

The validation service records the submitted proposal before checking it. It associates each test result with the exact proposal digest and rejects attempts to reuse an approval after the proposal changes. Approval is therefore a transition applied to a particular artifact, not a general statement that the assistant is trustworthy.

If execution fails, the runner should return the observed state and a stable failure identifier. The assistant may summarize that evidence and suggest an investigation, but retry authority remains governed by the runbook. Repeating a partially completed operation is permitted only when its documented recovery or idempotency contract makes that repetition valid.

5 Turn findings into reviewable changes

Planning should produce an ordered dependency graph rather than a flat checklist. Resolve foundational compatibility issues before application qualification, and qualify the application before scheduling cutover. A change to platform, topology, or character set can invalidate earlier measurements; the plan must make these dependencies explicit.

5.1 Oracle-to-Oracle assistance

Useful assistant tasks include grouping precheck findings by owner, linking them to documentation, drafting configuration changes, and identifying tests affected by a deprecation. Oracle's release-change catalog is a live input bound to the selected target, not a static list copied into a runbook. [3]

Execute approved actions through the selected supported mechanism. Keep generated configuration under version control and compare it with the reviewed configuration immediately before execution. An assistant's explanation of a warning cannot replace resolution or documented acceptance by its owner.

5.2 Heterogeneous transformations

For another source engine, define a type-and-behavior mapping before translating code. Review empty strings and nulls, numeric precision, timestamps and time zones, collation, identifier casing, generated keys, exceptions, and transactional DDL. Similar-looking SQL may express different contracts. A rewrite that compiles can still change the result at a boundary value.

Construct	Potential failure	Independent check
Numeric mapping	Rounding or overflow	Boundary fixtures
Date/time conversion	Offset or clock shift	Transition tests
Generated identifier	Collision after reload	Concurrent inserts
Error handling	Changed retry behavior	Failure injection
Text comparison	Changed equality	Locale fixtures

Table 2: Illustrative transformation review matrix.

5.3 Small patches, explicit abstention

Each patch should address one finding, list affected objects, preserve the original artifact, and include a rollback or forward-repair procedure. Reject changes that broaden permissions without justification or modify unrelated objects. Do not evaluate an assistant solely by how many patches it emits: abstaining from an unsupported rewrite can be the correct outcome.

Separate required migration fixes from optional modernization. Combining an engine transition with application redesign makes causal diagnosis harder and increases the behavior that must be requalified.

5.4 Review the behavioral contract

A transformation review should begin with the behavior that must be preserved. Suppose an application derives a financial total from nullable inputs. A reviewer must determine whether a null represents an unknown amount, an absent row, or a value that the application deliberately treats as zero. The correct rewrite depends on that contract. A visually similar aggregate expression is insufficient evidence.

Construct a small fixture for each interpretation and compare source and target outcomes. Include the error path and the transaction boundary, not only a successful query. Where the specification is ambiguous, record an application-owner decision before choosing a mapping. This turns an implicit dependency into an explicit requirement.

Review should also cover negative requirements: permissions that must not be broadened, objects that must not be modified, and data that must not enter generated logs. A candidate patch is accepted only when both its intended effect and its prohibited side effects have been checked. The patch count remains a descriptive measure, not a quality objective.

6 Verify behavior beyond successful execution

Validation must distinguish object completeness, data equivalence, query behavior, and transaction semantics. These layers catch different errors. A schema that imports cleanly is not proof that balances reconcile, and matching row counts are not proof that the same rows arrived.

6.1 Layered correctness gates

Layer	Evidence	Example rejection
Structure	Objects, constraints, grants, jobs	Missing constraint
Data	Consistent-snapshot reconciliation	Changed row
Semantics	Multi-fixture comparisons	Boundary mismatch
Transactions	Concurrency and failure traces	Duplicate posting
Application	End-to-end workflows	Broken integration

Table 3: Proposed independent correctness gates.

6.2 Reconciliation design

Compare data at an agreed consistency boundary. During ongoing writes, two unrelated snapshots can disagree even when replication is working. Record the snapshot or synchronization marker, partition boundaries, counts, and exceptions. Canonicalize types explicitly before checksumming; a formatted timestamp or locale-dependent numeric string is not a reliable cross-system representation.

Use counts, partition-level digests, and business invariants together. A ledger may require debit/credit equality and preserved account totals as well as row-level matching. Define how nulls, large objects, floating-point tolerances, and unordered results are compared. Every tolerated difference needs a domain rationale and accountable owner.

6.3 Differential and metamorphic tests

Execute equivalent workloads against source and target with duplicates, nulls, empty inputs, extreme values, and adverse ordering. Related text-to-SQL research motivates checking queries across multiple database instances rather than trusting agreement on one dataset. Its benchmark findings are not evidence of Oracle migration quality. [8]

Metamorphic properties complement differential comparison: adding an unrelated account should not change an existing balance; splitting a valid batch should preserve its final total. The property must reflect the application contract. Model-generated assertions need review because they can reproduce the same misunderstanding as a generated patch.

Release gate. All critical business invariants must pass. An unresolved discrepancy blocks promotion unless the application owner approves a precisely documented change in behavior.

6.4 Investigate a mismatch

A failed comparison is an observation that needs classification. First confirm that both executions used the intended schema, session settings, fixture, and consistency boundary. Then distinguish representation differences from changed values and changed business behavior. Preserve the failing input and both outputs before attempting a repair; otherwise the evidence needed to reproduce the defect may disappear.

Reduce the fixture to the smallest example that retains the mismatch. This assists diagnosis and makes the resulting regression test easier to review. A reduced example should retain the relevant type, nullability, ordering, and concurrency conditions. A reduction that removes the triggering condition can produce a misleading test that passes on both systems.

For every resolved discrepancy, link the explanation, corrective change, and independent rerun to the original finding. A waived difference requires a stated business rationale and owner. Do not silently remove a failing case from the corpus. Report excluded tests and the reason for exclusion alongside the acceptance results.

7 Qualify the service, not just the SQL

A target can preserve answers while failing its service-level objective. Qualification should include response-time distributions, throughput, CPU, I/O, locking, batch completion, and connection recovery. Evaluate representative concurrency and realistic data distributions; a small synthetic database may hide plan instability and contention.

7.1 Paired workload measurements

Establish a source baseline and run the same workload on an isolated target. Record hardware, storage, parameters, statistics, bind values, cache state, and competing load. Repeat trials and report variability. A single faster execution does not establish an improvement, and a mean can conceal severe tail-latency regression.

Oracle SQL Performance Analyzer supports before/after SQL workload analysis. Its reports can inform investigation of changed SQL behavior; application load tests remain necessary for service-level behavior. Verify entitlement before using separately licensed capabilities. [5], [11]

7.2 An explicit acceptance envelope

$$\text{Accept} = \text{C} \wedge \text{D} \wedge \text{P} \wedge \text{O} \wedge \text{R} \quad (1)$$

Here C denotes compatibility, D data and semantic correctness, P performance within the agreed envelope, O operational readiness, and R demonstrated recoverability. This conjunction is a proposed governance rule. A strong score on one dimension cannot compensate for a failed critical gate on another.

Dimension	Proposed criterion
Correctness	No unexplained critical invariant failures
Latency	Agreed p95/p99 budget at required load
Batch	Completes within the business deadline
Connections	Pool reconnect and retry validated
Recovery	Rehearsal meets agreed RTO and RPO

Table 4: Illustrative acceptance criteria; owners must set actual limits.

7.3 Operational completeness

Verify backups and restoration, monitoring, alert routing, scheduler execution, integrations, and privileges. Check endpoint resolution from each application environment. Recovery time objective (RTO) and recovery point objective (RPO) are project commitments that require measured evidence, not optimistic estimates from an assistant.

AI can summarize regressions and correlate evidence, but engineers should validate causal explanations. A plan change is an observation; identifying it as the cause of a service regression requires controlled investigation.

7.4 Interpret performance evidence

Compare measurements at equivalent service demand. A lower response time obtained by accepting fewer requests is not an improvement in the same workload. Record offered load, completed throughput, rejected work, queue depth, and timeout behavior together. Include the time spent waiting outside the database when that time contributes to the application service objective.

Aggregate statistics can hide critical regressions. A low-volume settlement operation may matter more than many fast lookup queries. Report results by business workflow and criticality as well as by total execution time. When a threshold is exceeded, preserve the workload inputs and environment state needed to reproduce the failure before applying tuning changes.

Capacity headroom should be assessed separately from a single successful load point. Increase demand within an agreed test envelope and observe the first limiting resource or service constraint. These experiments are proposed qualification activities. This paper supplies no measured throughput, latency, or capacity result and does not prescribe a universal percentage threshold.

8 Make the transition a controlled experiment

Cutover is a sequence of operational state changes. The proposed runbook records preconditions, commands or tool operations, expected evidence, maximum duration, decision authority, and the response to failure. Operators should identify the current state from artifacts rather than infer it from a model's narrative.

8.1 State and authority

Prepared → Rehearsed → Approved → Draining → Synchronized → Switched → Observed → Accepted

Each transition has an owner. Freeze the approved artifact bundle before the production window. Confirm target readiness and recoverable source state, drain write paths, establish the final consistency boundary, reconcile critical data, switch endpoints, and verify application behavior. Failed gates return to the documented recovery branch.

8.2 Recoverability after new writes

Before target writes begin, switching connections back may be possible if the source remains consistent and the route permits it. After target-only writes, a simple endpoint reversal can discard valid transactions or create divergence. The runbook must define the last safe reversal point and a separate strategy for later failures, such as tested reverse synchronization, restore-and-replay, or forward repair.

Do not equate a restore point, a backup, or an available old environment with a tested application recovery path. A rehearsal must cover credentials, keys, connectivity, reconciliation, external side effects, and the resulting service state. The selected migration method constrains available recovery actions. [1], [4]

8.3 Evidence and human accountability

Retain approvals, configuration digests, logs, reconciliation outputs, performance reports, incident decisions, and final acceptance. Use an append-only audit trail with appropriate access and retention controls. An assistant may assemble the record but cannot provide the accountable owner's approval.

Role	Accountable decision
Database engineering	Supported route, tooling, recovery
Application owner	Semantics and end-to-end acceptance
Security owner	Exposure, credentials, privileges
Change authority	Window, release, escalation

Table 5: Proposed responsibility boundaries.

Observe at least the operational cycles chosen during planning. A successful interactive smoke test does not establish that the next overnight close or month-end workload will succeed.

8.4 Define the incident decision boundary

The cutover runbook should identify the evidence that triggers a pause and the person authorized to choose recovery or forward repair. Examples include unexplained reconciliation failures, sustained critical workflow errors, or a synchronization state that cannot be established before the agreed deadline. Thresholds belong to the project contract and must be set before the production window.

Record the effect of external systems. A message already delivered to a downstream service may not be undone by restoring the database. Rehearse duplicate detection, compensation, or reconciliation for those effects where they are part of the application contract. The recovery plan must describe the state of the whole service, not merely the database files.

After an incident decision, preserve the timeline, observed markers, commands, and approval. Distinguish the moment a decision was made from the moment it became effective. This supports a later review of whether the runbook supplied enough information and whether its assumed recovery deadline was achievable.

9 A reproducible evaluation protocol

The central hypothesis is that evidence-gated assistance reduces engineering effort without reducing verified quality. This is untested. A credible study should compare matched tasks, record unsuccessful attempts, and separate Oracle upgrades from heterogeneous translation. Mixing these populations would obscure which intervention produced an effect.

9.1 Experimental conditions

Compare **A:** conventional tooling and documentation; **B:** the same baseline plus unconstrained AI assistance; and **C:** the same baseline plus the proposed evidence contract and independent gates. Keep model, tool versions, inputs, and access fixed between B and C. Give all groups the same acceptance criteria.

Stratify tasks by object complexity, dependency count, criticality, and route. Use counterbalanced assignment to reduce learning effects, and prevent engineers from repeating the same task under another condition. Include hidden holdout fixtures and independent reviewers blinded to the assistance condition when practical.

9.2 Outcomes and analysis

Metric	Operational definition
Engineering effort	Active discovery, implementation, review, rework
Verified completion	Passes all predeclared acceptance gates
Escaped defect	Holdout detects a missed defect
Evidence quality	Applicable and traceable recommendation support
Total cost	Engineering, inference, infrastructure, tools

Table 6: Metrics to collect; no results have been measured.

Report paired differences or stratified estimates with confidence intervals. Count failed and abandoned tasks; do not report savings only among successful AI runs. Pre-register a quality non-inferiority margin justified by application risk, and derive sample size from pilot variance. A small study with no severe failures cannot establish production safety.

9.3 Threats to validity

Model and documentation drift, benchmark leakage, engineer experience, incomplete discovery, and unrepresentative workloads limit generalization. Expert review costs time and may dominate simple tasks. Tests approximate a specification and can omit rare concurrency or recovery failures. These limits require explicit reporting, not a model confidence score.

9.4 Publish failures and resource use

A study report should expose the denominator for every metric. List assigned tasks, completed tasks, failed tasks, and exclusions for each condition. Report assistance attempts that exhausted their budget or required substantial manual repair. Counting only accepted patches would conceal effort spent evaluating incorrect or irrelevant proposals.

Preserve the unit of analysis. Multiple attempts on the same task are not independent observations. Where several tasks share an application or engineer, account for that grouping when estimating uncertainty. Define the treatment of interrupted runs, infrastructure failures, and missing reviewer decisions before examining comparative outcomes.

Release a redacted task specification, fixture generator, scoring rubric, and environment manifest when permitted. If an artifact cannot be released, explain the resulting reproducibility limitation. Report model and tool costs separately from labor assumptions so that readers can recalculate economic outcomes under their own conditions. These reporting requirements are part of the proposed study protocol, not results from a completed experiment.

10 Conclusion

AI assistance is most defensible when it produces inspectable proposals inside a supported, independently verified migration process. The next research step is to run the comparative study and publish artifacts and failures. Until then, this paper provides a framework to test, not evidence of a proven productivity gain.

References

Primary documentation and related research. Web sources checked on 25 October 2026.

[1] Oracle. 2026. Oracle AI Database 26ai: Database Upgrade Guide. Upgrade planning, supported paths, AutoUpgrade, and post-upgrade tasks.

[2] Oracle. 2026. Scripts Associated with Upgrading Oracle AI Database. Supported upgrade tooling and desupported manual methods.

[3] Oracle. 2026. Oracle Database Changes, Desupports, and Deprecations. Version-sensitive changes; check against the target release update.

[4] Oracle. 2026. Introduction to Zero Downtime Migration, Release 26.1. Physical and logical migration methods and target categories.

[5] Oracle. 2026. Introduction to SQL Performance Analyzer. Before/after SQL workload comparison and isolated testing.

[6] Oracle. 2026. JDBC Developer's Guide: Getting Started. Driver, database, and JDK interoperability.

[7] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. Related research on constraining generated SQL syntax.

[8] Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic Evaluation for Text-to-SQL with Distilled Test Suites. Related research motivating evaluation across multiple database instances.

[9] Patrick Lewis et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Foundational retrieval research; no Oracle migration results.

[10] Oracle. 2026. Database Upgrades and Migrations. Official entry point for current tools, CPAT, and migration resources.

[11] Oracle. 2026. Tools for Tuning the Database. Database Replay, SQL Performance Analyzer, and licensing considerations.

Source interpretation

References [1-6, 10-11] support product and tooling descriptions. References [7-9] are adjacent research, not validation of the proposed framework.

Reproducibility and document control

This publication describes a proposed methodology. Its examples define hypothetical engineering tasks and must not be interpreted as reports of production deployments. To reproduce a future study, retain the exact paper revision with the task corpus, documentation snapshots, model configuration, and scoring implementation used for that study.

Public documentation links identify the primary source at the time of writing. A later page at the same address may contain revised guidance. Record retrieval dates and content digests when using these sources for an operational decision. Vendor support matrices and application certifications remain project-specific inputs that this paper cannot replace.

Separate public publication artifacts from private migration evidence. A publicly shared experiment should contain no production secrets, personal data, private object names, or unrestricted database extracts. Redaction should preserve the behavior needed for evaluation and should itself be documented so that readers can understand which properties of the original task remain testable.