CHECKLIST

NoSQL Data Modeling Production Readiness Checklist

A practical checklist for validating NoSQL data models before rollout, focused on access patterns, partitioning, indexing, duplication, and operational safety.

NoSQL Data Modeling Production Readiness Checklist

Use this checklist to validate a NoSQL data model before production rollout. The goal is to confirm that the design matches real access patterns, distributes load safely, and can evolve without creating performance or operational surprises.

---

1) Access patterns are defined first

- [ ] The top read paths are documented.
- [ ] The top write paths are documented.
- [ ] Each critical query has a clear business purpose.
- [ ] Each query is ranked by frequency and latency sensitivity.
- [ ] The model is designed around actual access patterns, not entity purity.
- [ ] There is a clear answer to: ?What must be fetched together??
- [ ] There is a clear answer to: ?What changes together??

2) Key design supports distribution

- [ ] The primary key supports the dominant lookup pattern.

- [ ] The partition key spreads traffic evenly under realistic load.
- [ ] Known skew sources have been identified, such as:
- [ ] Hot tenants
- [ ] Time-based write bursts
- [ ] Popular accounts or entities
- [ ] Sequential IDs
- [ ] The design avoids concentrating traffic on a small number of partitions.
- [ ] The model has been checked for hotspot risk during peak traffic.
- [ ] Cardinality of key values is sufficient for expected scale.

3) Document and entity boundaries are intentional

- [ ] Related data that is usually read together is embedded or co-located.
- [ ] Large or frequently changing fields are not embedded without reason.
- [ ] The maximum document or item size is known and respected.
- [ ] Fields that change independently are separated when appropriate.
- [ ] The model avoids unnecessary joins or cross-partition lookups.
- [ ] The design clearly distinguishes between source-of-truth data and read-optimized copies.

4) Duplication is controlled

- [ ] Any duplicated data has a named owner or update source.
- [ ] The reason for duplication is tied to a specific query path.
- [ ] Update propagation rules are defined.
- [ ] Staleness tolerance is explicitly accepted where duplication exists.
- [ ] There is a documented way to reconcile divergent copies.
- [ ] No duplicated field exists ?just in case.?

5) Indexing is deliberate

- [ ] Every secondary index maps to a known production query.
- [ ] Each index is justified by measurable read benefit.
- [ ] The write overhead of each index has been reviewed.
- [ ] The number of indexes is minimized.
- [ ] Indexes are not being used to compensate for a weak key or partition design.
- [ ] Index growth has been estimated at expected scale.
- [ ] Queries are tested to confirm the index is actually used.

6) Write behavior is safe at scale

- [ ] Write amplification has been estimated.
- [ ] The model avoids unnecessary fan-out on updates.
- [ ] Update frequency for each field is understood.
- [ ] High-churn fields are isolated when possible.
- [ ] Bulk write behavior has been considered.
- [ ] Conflicting writes and last-write-wins behavior are understood.
- [ ] Retry behavior will not create duplicate side effects.

7) Read performance is predictable

- [ ] The main reads can be served without broad scans.
- [ ] Tail latency has been evaluated, not just average latency.
- [ ] Query paths have been tested under realistic concurrency.
- [ ] Large result sets are paginated or bounded.
- [ ] Queries that may scan are known and intentionally accepted.

- [ ] Cross-partition reads are minimized or justified.

8) Data evolution is manageable

- [ ] The schema can evolve without a full rewrite.
- [ ] New optional fields can be added safely.
- [ ] Old and new versions can coexist during rollout.
- [ ] Backfill or migration steps are documented.
- [ ] Rollback steps are documented.
- [ ] The team knows how to handle partially migrated data.
- [ ] Validation rules exist for required fields and allowed shapes.

9) Consistency and correctness are understood

- [ ] The required consistency level is defined per access path.
- [ ] Eventual consistency is acceptable where used.
- [ ] Strong consistency is reserved for paths that truly need it.
- [ ] The application can tolerate temporary stale reads where applicable.
- [ ] The source of truth for each business object is clear.
- [ ] Conflict resolution behavior is documented.

10) Security and access boundaries are considered

- [ ] Sensitive fields are identified.
- [ ] Read and write permissions align with service responsibilities.
- [ ] The model supports least-privilege access.
- [ ] Shared collections or datasets do not blur tenant boundaries.
- [ ] Data layout does not make authorization checks ambiguous.

- [ ] Security-sensitive updates are auditable.

11) Operational readiness is verified

- [ ] The model has been tested with realistic data volume.
- [ ] The model has been tested with realistic skew and cardinality.
- [ ] Failure modes have been reviewed, including throttling and retry storms.
- [ ] Monitoring exists for latency, throughput, error rate, and hot partitions.
- [ ] Alerts exist for unusual scan patterns or index pressure.
- [ ] Backup and recovery implications are understood.
- [ ] Capacity planning assumptions are documented.

12) Production rollout is gated

- [ ] A staging or pre-production test mirrors expected behavior closely enough.
- [ ] The model has been load tested at expected peak traffic.
- [ ] The team has reviewed read/write trade-offs explicitly.
- [ ] Incident response ownership is clear.
- [ ] A rollback or fallback plan exists if the model performs poorly.
- [ ] The rollout can be paused if hotspots or latency regressions appear.

Quick decision checks

Use these final questions before approving the model:

- [ ] Can the top queries be served without expensive scans?
- [ ] Will the chosen key distribution hold under real traffic skew?
- [ ] Is duplication intentional and manageable?

- [ ] Are indexes supporting the model instead of masking design issues?
- [ ] Can the schema evolve without a full redesign?
- [ ] Are consistency, security, and operational risks clearly understood?

Red flags that should block rollout

- [ ] A small number of partitions will receive most traffic.
- [ ] The model depends on multiple lookups for every critical request.
- [ ] Indexes were added before the access pattern was proven.
- [ ] Large documents are growing without size or update controls.
- [ ] Duplication exists without ownership or sync rules.
- [ ] The team cannot explain the update and rollback path.
- [ ] The design has not been tested against real-world skew.

Final approval statement

Before release, confirm the following:

> This NoSQL model is built around known access patterns, distributes load safely, supports the required reads and writes, and has documented controls for indexing, updates, consistency, security, and rollback.

If that statement is not fully true, the model is not ready for production.