CHECKLIST

Secure MongoDB Index Design Checklist

A practical PDF checklist for designing MongoDB indexes that support fast queries without weakening tenant, role, or document-level security boundaries.

Secure MongoDB Index Design Checklist

Use this checklist to review MongoDB indexes before they are added, changed, or promoted to production. It is designed to help you keep query paths fast while avoiding unnecessary exposure of sensitive data.

---

1) Confirm the access model first

- [ ] Identify the approved application roles, services, and tenants that will use the collection.
- [ ] Document the required access boundaries: tenant ID, account ID, environment ID, role scope, or another scoping key.
- [ ] Verify that the query pattern is part of a legitimate, repeatable workflow.
- [ ] Confirm that the index supports authorized behavior, not ad hoc exploration.
- [ ] If data model boundaries are not defined yet, pause index design until access control rules are clear.

2) Define the exact query shape

- [ ] List the filters, sorts, and joins the application actually uses.

- [ ] Identify the smallest useful query shape.
- [ ] Separate operational queries from reporting, analytics, and debugging queries.
- [ ] Note which fields are always present in the predicate.
- [ ] Confirm which query shapes are safe to optimize and which should remain slower by design.

3) Choose the right leading fields

- [ ] Put the scoping field first in the compound index when it is part of the authorization boundary.
- [ ] Prefer tenant, account, or role scope as the leading field when applicable.
- [ ] Place the most selective and most common approved boundary early in the index.
- [ ] Avoid leading with sensitive attributes unless they are required for a controlled, approved access path.
- [ ] Do not choose a leading field just because it is convenient for ad hoc search.

4) Keep the index as narrow as possible

- [ ] Include only fields needed for approved filters, sorts, or joins.
- [ ] Exclude fields that are merely available in the document but not necessary for the query.
- [ ] Avoid indexing internal identifiers, recovery attributes, fraud flags, escalation metadata, or other sensitive operational fields unless required.
- [ ] Check whether a smaller index can satisfy the same access boundary.
- [ ] Prefer one purpose-built index over one broad index that serves many unrelated use cases.

5) Validate that the safe query path is the fast path

- [ ] Run explain plans for the intended query.
- [ ] Confirm the index is used for the approved access pattern.
- [ ] Verify the plan does not encourage broader searches than intended.
- [ ] Check that the optimizer does not favor an alternate index that weakens boundaries.
- [ ] Compare the plan against the authorization logic used by the application.

6) Review exposure risk

- [ ] Ask whether the indexed field makes sensitive data easier to search, sort, or aggregate.
- [ ] Confirm that application roles are allowed to query every indexed sensitive field.
- [ ] Review whether query logs or explain output reveal more data shape than desired.
- [ ] Make sure indexing does not turn a protected attribute into a convenient query dimension.
- [ ] Verify that debugging and support workflows cannot misuse the new index.

7) Handle partial and sparse indexes carefully

- [ ] Use partial or sparse indexes only when the coverage limits are intentional and documented.
- [ ] Verify that excluded documents are excluded for a reason, not by accident.
- [ ] Confirm operators and support teams understand what the index does not cover.
- [ ] Check whether incomplete search coverage is acceptable for security and operations.
- [ ] Document any conditions that make a document searchable or unsearchable.

8) Separate operational and analytical access paths

- [ ] Do not reuse operational indexes for broad reporting if doing so exposes more of the data shape than intended.

- [ ] Create separate purpose-built indexes for reporting or analytics when needed.
- [ ] Apply separate authorization rules to analytical workflows.
- [ ] Keep support, analyst, and automation paths distinct when they have different trust levels.
- [ ] Confirm each path has the minimum index support needed.

9) Evaluate write and maintenance cost

- [ ] Count how many indexes already exist on the collection.
- [ ] Estimate insert and update overhead from the new index.
- [ ] Check whether high-ingest workloads can absorb the added write cost.
- [ ] Remove unused indexes before adding new ones.
- [ ] Prefer a smaller index set aligned to real traffic.

10) Review deployment impact

- [ ] Confirm when the index will be built or rebuilt.
- [ ] Check for performance impact on live traffic during build time.
- [ ] Review replication, backup, and rollback implications.
- [ ] Ensure the rollout window fits operational constraints.
- [ ] Verify that the new index does not create a broader query path than the one it replaces.

11) Validate before production

- [ ] Test the index against representative production-like data.
- [ ] Confirm expected latency improvements under load.
- [ ] Verify behavior across relevant roles, tenants, and services.

- [ ] Review audit logs for unexpected query patterns.
- [ ] Obtain sign-off from both security and operations if the index touches sensitive data.

12) Use this decision rule

A secure index is usually acceptable when all three are true:

- [ ] The query is part of a legitimate, repeatable access pattern.
- [ ] The leading fields align with a boundary such as tenant, account, or role scope.
- [ ] The plan and access rules have been validated before production.

An index is likely too broad when any of these are true:

- [ ] It begins with a sensitive attribute that is not a required boundary.
- [ ] It mainly supports ad hoc analysis.
- [ ] It makes queries easier that should instead be controlled by authorization logic.

Quick review summary

Before approving a MongoDB index, make sure it:

- [ ] Supports a real application query path
- [ ] Starts with the correct access boundary
- [ ] Avoids unnecessary sensitive fields
- [ ] Has an explain plan that matches intent
- [ ] Does not expand exposure through convenience
- [ ] Fits operational, deployment, and write-cost constraints

---

Final approval questions

- [ ] Would this index still be acceptable if an unauthorized user could see the query pattern but not the data?
- [ ] Does this index make the safe path the easiest path?
- [ ] Have we confirmed the index does not widen access beyond policy?
- [ ] Is the performance gain worth the operational and security trade-offs?

If the answer to any of these is no, redesign the index before release.