Multi-Tenancy Models

Version 1.2.2-SNAPSHOT, 2025-09-13T18:13:34Z

Table of Contents

Quantum supports multiple multi-tenant models for MongoDB deployments:

Chapter 1. One Tenant per Database (in a MongoDB Cluster)

- Each tenant is mapped to a dedicated MongoDB database within a cluster.

- Strong isolation at the database level; operational controls via MongoDB roles.

- Pros: Simplified backup/restore per tenant; reduced risk of data bleed.

- Cons: More databases to manage (indexes, connections), higher operational overhead.

How Quantum helps:

- DataDomain carries tenant identifiers (e.g., tenantId, ownerId, orgRefName) on each model.

- Repositories can resolve connections/DB selection per tenant, enabling routing to the appropriate database.

Chapter 2. Many Tenants in One Database (Shared Database)

- Multiple tenants share a single database and collections.

- Isolation is enforced at the application layer using DataDomain filters.

- Pros: Fewer databases to manage; efficient index utilization and connection pooling.

- Cons: Strict discipline required to enforce filtering and access rules.

How Quantum helps:

- DataDomain is part of every persisted model, enabling programmatic, rule-based filtering.

- RuleContext and DomainContext can be used to inject tenant-aware filters into repositories and resources.

- Cross-tenant sharing can be modeled by specific DataDomain fields and RuleContext logic granting read access across tenants on a per-functional-area basis.

Chapter 3. Freemium and Trial Tenants

- Programmatically create tenants to support self-service onboarding.

- Attach time-bound or capability-bound policies.

- Use scheduled jobs to convert/expire trials.

Quantum patterns:

- Tenant onboarding service creates a DataDomain scope and any default records.

- Policies are encoded in RuleContext checks to allow or restrict actions based on time, plan, or feature flags.