WorkspaceOps — MongoDB Schema Mapping (From SQL)

Ground rules we are following (important)

1. **Workspace-scoped everything**
2. **No hidden nesting**
3. **No smart denormalization**
4. **Join tables stay collections**
5. **Future SQL migration must remain possible**

If Mongo tempts you to "embed everything" — **resist it**.

1️⃣ `tenants` → `tenants` collection

SQL source

`tenants`

Mongo collection

- `tenants`

Document shape

- `{`
- `  _id: ObjectId,`
- `  name: "Acme Corp",`
- `  createdAt: ISODate`
- `}`

Notes

- Very stable
- Rarely queried directly

2️⃣ `workspaces` → `workspaces` collection

SQL source

`workspaces`

- `workspaces`

- `{`
- `  _id: ObjectId,`
- `  tenantId: ObjectId,`
- `  name: "Main Workspace",`
- `  createdAt: ISODate`
- `}`

Indexes

- `db.workspaces.createIndex({ tenantId: 1 })`

3️⃣ `users` → `users` collection

SQL source

`users`

- `users`

- `{`

- `  _id: ObjectId,`
- `  email: "user@example.com",`
- `  passwordHash: "...",`
- `  createdAt: ISODate`
- `}`

Indexes

- `db.users.createIndex({ email: 1 }, { unique: true })`

Rule

❗ **NO workspace data here**

4️⃣ `workspace_users` → `workspaceUsers` collection (RBAC spine)

SQL source

`workspace_users`

- `workspaceUsers`

- `{`
- `  _id: ObjectId,`
- `  workspaceId: ObjectId,`
- `  userId: ObjectId,`
- `  role: "OWNER" | "ADMIN" | "MEMBER" | "VIEWER",`
- `  createdAt: ISODate`
- `}`

Indexes (critical)

- `db.workspaceUsers.createIndex(`
- `  { workspaceId: 1, userId: 1 },`
- `  { unique: true }`

- `)`
-
- `db.workspaceUsers.createIndex({ userId: 1 })`

Why NOT embed in workspace

- User can belong to many workspaces
- Role is workspace-specific
- This keeps RBAC clean

5️⃣ `entities` → `entities` collection

SQL source

`entities`

- `entities`

```
{
  _id: ObjectId,
  workspaceId: ObjectId,
  name: "John Doe",
  role: "SELF" | "CUSTOMER" | "EMPLOYEE" | "VENDOR",
  createdAt: ISODate
}
```

Indexes

```
db.entities.createIndex({ workspaceId: 1 })
db.entities.createIndex({ role: 1 })
```

Rule

❌ No nesting
❌ No parent-child
Flat by design

6️⃣ `document_types` → `documentTypes` collection

SQL source

`document_types`

- `documentTypes`

- `{`
- `  _id: ObjectId,`
- `  workspaceId: ObjectId,`
- `  name: "Passport",`
- `  hasMetadata: true,`
- `  hasExpiry: true`
- `}`

Index

- `db.documentTypes.createIndex({ workspaceId: 1 })`

7️⃣ `document_type_fields` → embedded inside `documentTypes`

This is one of the **few intentional embeds**.

SQL source

`document_type_fields`

Mongo decision

✅ **Embed** (schema definition belongs to type)

- `{`
- `  _id: ObjectId,`
- `  workspaceId: ObjectId,`
- `  name: "Passport",`
- `  hasMetadata: true,`
- `  hasExpiry: true,`
- `  fields: [`
- `    {`
- `      key: "passportNumber",`
- `      type: "text",`
- `      required: true,`
- `      isExpiryField: false`
- `    },`
- `    {`
- `      key: "expiryDate",`
- `      type: "date",`
- `      required: true,`
- `      isExpiryField: true`
- `    }`
- `  ]`
- `}`

Why embed?

- Fields never queried alone
- Always fetched with document type
- Pure configuration

8️⃣ `documents` → `documents` collection

SQL source

`documents`

- `documents`

- `{`

- `  _id: ObjectId,`
- `  workspaceId: ObjectId,`
- `  documentTypeId: ObjectId,`
- `  entityId: ObjectId | null,`
- `  fileUrl: "https://s3/...",`
- `  metadata: {`
- `    passportNumber: "A123456",`
- `    expiryDate: ISODate("2030-01-01")`
- `  },`
- `  createdAt: ISODate`
- `}`

Why metadata is embedded here

- Metadata is document-specific
- Read together
- Avoid extra collection joins

Indexes

- `db.documents.createIndex({ workspaceId: 1 })`
- `db.documents.createIndex({ entityId: 1 })`
- `db.documents.createIndex({ documentTypeId: 1 })`

9️⃣ `work_item_types` → `workItemTypes` collection

SQL source

`work_item_types`

- `workItemTypes`

- `{`
- `  _id: ObjectId,`
- `  workspaceId: ObjectId,`
- `  name: "Employee Visa Renewal"`

- `}`

Index

- `db.workItemTypes.createIndex({ workspaceId: 1 })`

🔟 `work_items` → `workItems` collection

SQL source

`work_items`

- `workItems`

- `{`
- `  _id: ObjectId,`
- `  workspaceId: ObjectId,`
- `  workItemTypeId: ObjectId,`
- `  entityId: ObjectId,`
- `  ownerUserId: ObjectId,`
- `  status: "DRAFT" | "ACTIVE" | "COMPLETED",`
- `  createdAt: ISODate`
- `}`

Indexes

- `db.workItems.createIndex({ workspaceId: 1 })`
- `db.workItems.createIndex({ entityId: 1 })`
- `db.workItems.createIndex({ ownerUserId: 1 })`
- `db.workItems.createIndex({ status: 1 })`

1️⃣1️⃣ `work_item_documents` → `workItemDocuments` collection

SQL source

`work_item_documents`

- `workItemDocuments`

- `{`
- `  _id: ObjectId,`
- `  workItemId: ObjectId,`
- `  documentId: ObjectId`
- `}`

Indexes

- `db.workItemDocuments.createIndex(`
- `  { workItemId: 1, documentId: 1 },`
- `  { unique: true }`
- `)`

Why NOT embed documents in workItems

- Documents can be reused
- Avoid duplication
- Avoid bloated work item docs

1️⃣2️⃣ `audit_logs` → `auditLogs` collection

SQL source

`audit_logs`

- `auditLogs`

- `{`
- `  _id: ObjectId,`
- `  workspaceId: ObjectId,`
- `  userId: ObjectId,`
- `  action: "CREATE_DOCUMENT",`
- `  targetType: "DOCUMENT",`
- `  targetId: ObjectId,`
- `  createdAt: ISODate`
- `}`

Indexes

- `db.auditLogs.createIndex({ workspaceId: 1 })`
- `db.auditLogs.createIndex({ userId: 1 })`
- `db.auditLogs.createIndex({ createdAt: -1 })`

Mongo vs SQL — sanity check

Concept	SQL	Mongo
Tenant boundary	FK	tenantId
Workspace scope	FK	workspaceId
Join tables	Tables	Collections
Metadata	KV ta	Embedded object

	bl e	
Schema config	Tables	Embedded arrays

Nothing is lost.

Final verdict (facts only)

- ✅ This Mongo schema is **correct**
- ✅ No shortcuts taken
- ✅ SQL migration remains possible
- ❌ No over-embedding
- ❌ No premature optimization
-