Data Architecture Modeling Guide

Source File: generated-documents\dmbok\data-architecture-modeling-guide.md
Generated: 30/07/2025 at 06:57:22
Generated by: Requirements Gathering Agent - PDF Converter

Data Architecture & Modeling Guide

Generated by adpa-enterprise-framework-automation v3.2.0
Category: dmbok
Generated: 2025-07-16T10:39:46.118Z
Description: Provides target and current state data architecture with modelling standards and roadmap.

Data Architecture & Modeling Guide

For: adpa-enterprise-framework-automation
Aligned to DMBOK 2.0, BABOK v3, and PMBOK 7th Edition Principles

---

1. Introduction

Purpose:
This guide defines the data architecture and modeling standards for the *adpa-enterprise-framework-automation* project—a modular, standards-compliant Node.js/TypeScript automation framework for enterprise requirements, project, and data management.

Scope:
Covers data modeling, architecture principles, governance, and implementation for all framework modules and integrations (including Adobe Creative Suite, SharePoint, Azure, API Center, and AI services).

Strategic Alignment:
Supports organizational data strategies: data quality, security, scalability, and regulatory compliance. Ensures alignment with DMBOK 2.0 best practices.

---

2. Architectural Principles

Core Data Architecture Principles:

- **Data as an Asset**: Treat data as a core enterprise asset with appropriate stewardship.
- **Security by Design**: Embed security and privacy controls at all data layers.
- **Standards Compliance**: Align with DMBOK, PMBOK, and BABOK requirements.
- **Modularity & Reuse**: Promote modular, reusable data models and APIs.
- **Scalability & Performance**: Design for high throughput, low latency, and horizontal scaling.
- **Lineage & Traceability**: Ensure metadata and audit trails for all data flows.

Technology Stack:

- **Backend:** Node.js (TypeScript), Express.js, TypeSpec

- **Data Integration:** REST APIs, OpenAPI, Azure API Center, Microsoft Graph API, Adobe Creative Suite APIs
- **Storage:** Relational (e.g., Azure SQL/PGSQL), Document DB (e.g., MongoDB), Object Stores (Azure Blob, SharePoint)
- **Authentication:** OAuth2, Azure AD, JWT
- **Tooling:** Swagger/OpenAPI, Redocly, Jest, TypeSpec, ERD tools

---

3. Current vs Target Architecture

3.1 Current State

- Markdown-driven pipeline → Puppeteer → PDF output.
- Core document management, template handling, and standards compliance APIs operational.
- Single-user, CLI/REST API, basic authentication and metadata.

3.2 Target State

- Content Analysis → Template Selection → Adobe Creative APIs (InDesign, Illustrator, Photoshop) → Premium Outputs.
- Multi-user, role-based access, real-time collaboration (WebSockets), approval workflows.
- Azure API Center managed APIs, advanced metadata, and automated compliance checks.
- Integration with SharePoint for document publishing and versioning.

3.3 Gap Analysis

- **Multi-user support** and real-time collaboration to be implemented.
- Full Adobe Creative API integration pending.
- Enhanced data lineage, master/reference data management, and unified governance needed.

---

4. Data Modeling Standards

Naming Conventions:

- **Entities/Tables:** PascalCase (e.g., `DocumentJob`, `RequirementsTemplate`)
- **Attributes/Columns:** camelCase (e.g., `createdAt`, `templateId`)
- **Primary Keys:** `id` or `<entity>NameId`
- **Foreign Keys:** `<referencedEntity>Id`
- **Junction Tables:** `<EntityA>_<EntityB>`

Normalization/Denormalization:

- **OLTP:** Third Normal Form (3NF) for transactional entities.
- **OLAP/Reporting:** Denormalized for performance, with clear documentation of derivations.
- **Document Stores:** Embedded documents for tightly coupled data; references for loose coupling.

Modeling Notation:

- **ERD:** Crow's Foot notation for relational schema.
- **UML/Class Diagrams:** For object/TypeScript models.
- **JSON Schema:** For API payload definitions.

---

5. Logical Data Model

Core Domain Entities:

Entity	Key Attributes	Relationships
User	id, email, name, role, status, createdAt	Member of Team; Owner of DocumentJob, Requirements

Entity	Key Attributes	Relationships
Team	id, name, description, createdAt	Has Members (User); Manages Projects
Project	id, name, description, teamId, status	Contains DocumentJobs, Requirements, Templates
DocumentJob	id, projectId, userId, templateId, ...	Uses Template; Generates Output; Linked to Files
RequirementsTemplate	id, name, version, category, ...	Used by DocumentJobs; Tagged with Standards
Standard	id, name, version, type	Linked to Templates, Compliance Checks
File/Asset	id, jobId, type, location, metadata	Output of DocumentJob; Published to SharePoint/Adobe
AuditLog	id, userId, action, entityType, ...	Tracks changes across all main entities

Constraints:

- All IDs are UUIDs.
- Email is unique for User.
- Foreign keys must have ON DELETE CASCADE where appropriate.

- DocumentJobs cannot be orphaned (must belong to a Project).

---

6. Physical Data Model

Schema per Environment:

- **Development:** Use suffix `_dev` for schemas/tables.
- **Test:** Use `_test`.
- **Production:** No suffix.

Indexing & Partitioning:

- **Indexes:** On primary keys, foreign keys, and search/filter columns (e.g., `createdAt`, `status`).
- **Partitioning:** By `teamId` or `projectId` for scalability (if supported by DB).
- **API Payloads:** Validate and serialize using TypeSpec, Zod, or Joi.

Performance Considerations:

- Use pagination for all list endpoints.
- Prefer asynchronous, batched writes for document generation jobs.
- Use document storage (e.g., Azure Blob) for large output files; only store metadata/URIs in DB.

---

7. Master & Reference Data Considerations

- **Standards Catalog:** Reference datasets for BABOK, PMBOK, DMBOK, etc. (versioned, read-only).
- **Templates Library:** Master data, versioned, with approval and publishing workflows.
- **Roles & Permissions:** Enumerated, centrally managed, extensible for new roles.
- **Status Enums:** Centralized management for all workflow statuses.
- **Tag Management:** Controlled vocabulary for template/project tagging.

---

8. Governance & Stewardship

Roles & Responsibilities:

- **Data Steward:** Ensures data quality, reference/master data management.
- **Data Owner:** Accountable for business meaning and compliance of data entities.
- **Data Architect:** Maintains models, oversees changes, ensures alignment with DMBOK.

Change-Control Process:

- All changes to data models require:
  - Impact analysis
  - Peer review (data architect, stewards)
  - Versioning and migration scripts
  - Documentation update (ERDs, JSON Schemas)
- Use PRs and change logs in Git for traceability.

---

9. Tooling & Repository Management

- **Modeling Tools:**
  - ERD: dbdiagram.io, Lucidchart, or equivalent
  - API: Swagger/OpenAPI, TypeSpec, Redocly
  - Data Validation: Joi, Zod, AJV
- **Repository:**
  - All models, schemas, and migrations stored in Git (src/models, src/schemas, docs/erd)
  - Use semantic versioning for models and APIs.
- **Version Control Strategy:**
  - Feature-branch workflow for model changes.
  - Tags/releases for major model versions.
  - Scripts for automated migration and rollback.

---

10. Implementation Roadmap

Phase	Milestone	Target Date
1	Baseline logical/physical model; core entities	Complete
2	Multi-user, role/permissions, team/project collaboration	In Progress
3	Integrate Adobe Creative Suite APIs and metadata	July 2025
4	Enhanced lineage, audit, and governance features	Q3 2025
5	Advanced reporting, OLAP cube/data warehouse integration	Q4 2025

---

11. Approval

Name	Role	Signature	Date
[Data Architect]	Data Architecture		
[Data Steward]	Data Governance		
[Product Owner]	Product		

References:

- DAMA-DMBOK 2.0
- BABOK v3, PMBOK 7th Edition
- Project documentation: ARCHITECTURE.md, PHASE-2-IMPLEMENTATION-GUIDE.md, COLLABORATION-TOOLS-ROADMAP.md

---

This document is to be reviewed and updated as the project evolves, with each major architectural or modeling milestone.
