Data Engineering for AI Governance Checklist

Your Complete Guide to Building Governance-Ready Data Infrastructure

---

Introduction

Before you can implement AI governance, you need the right data foundation.

This checklist provides the specific data engineering requirements for each governance principle, RAG prerequisites, validation frameworks, and a Phase 0 implementation roadmap.

How to use this checklist:

- Review each section against your current infrastructure
- Check boxes as you complete items
- Prioritize items marked as "Critical" for immediate implementation
- Use this as a pre-governance assessment tool

Let's ensure your data architecture can support the governance your AI needs.

Part 1: Data Infrastructure Requirements by Governance Principle

Principle 1: Context-Aware Generation (RAG)

Your AI needs context before it generates outputs. Here's what your data infrastructure must support:

Schema Documentation & Indexing

- ☐ **Database schemas fully documented** (all tables, columns, relationships, data types)
- ☐ **Schema metadata available in structured format** (JSON, YAML, or database catalog)
- ☐ **Relationships between tables clearly defined** (foreign keys, joins, dependencies)
- ☐ **Business logic documented** (what each table/column represents in business terms)
- ☐ **Data dictionary created** (definitions of key terms, acceptable values, constraints)

Semantic Search Infrastructure

- ☐ **Vector database or search engine deployed** (Azure AI Search, Pinecone, Weaviate, Elasticsearch)
- ☐ **Schema indexed for semantic search** (tables, columns, and relationships searchable)

- [ ] **Embedding model selected and tested** (OpenAI embeddings, sentence transformers, etc.)
- [ ] **Search retrieval tested** (can you find relevant schema with natural language queries?)
- [ ] **Search latency acceptable** (<500ms for schema retrieval)

Context Injection Pipeline

- [ ] **Context retrieval automated** (given a query, system retrieves relevant schema)
- [ ] **Context formatting standardized** (consistent structure for injecting into prompts)
- [ ] **Context size optimized** (not too much data, not too little - typically 2-5 relevant tables)
- [ ] **Context caching implemented** (don't re-retrieve same schema repeatedly)

Critical Priority Items:

- Schema documentation (can't build context without it)
- Vector search deployed (RAG foundation)
- Context retrieval automated (manual won't scale)

Principle 2: Multi-Layer Validation

Your AI outputs must pass through validation checkpoints. Here's the data infrastructure needed:

Data Quality Rules

- [ ] **Data quality rules documented** (what makes data "valid" in your domain)
- [ ] **Validation rules codified** (in SQL, Python, or validation framework)
- [ ] **Expected data ranges defined** (min/max values, acceptable formats)
- [ ] **Required fields identified** (what must be present vs. optional)
- [ ] **Data type constraints enforced** (integers, strings, dates, enums)

Schema Validation Infrastructure

- [ ] **Schema validation library integrated** (JSON Schema, Pydantic, Great Expectations)
- [ ] **AI outputs validated against schema** before execution
- [ ] **Schema misalignment errors caught** (AI references non-existent tables/columns)
- [ ] **Data type validation enforced** (AI generates correct types)

Business Rule Validation

- [ ] **Business rules accessible to validation layer** (stored in database, config files, or rules engine)
- [ ] **Rule validation automated** (AI outputs checked against business constraints)

- [ ] **Regulatory compliance rules codified** (HIPAA, SOX, PCI requirements)
- [ ] **Custom validators built** (domain-specific validation logic)

Test Data Infrastructure

- [ ] **Test/staging database available** (for dry-run validation)
- [ ] **Test data representative of production** (covers edge cases, not just happy path)
- [ ] **Test execution automated** (AI outputs tested before production)
- [ ] **Test results logged** (track what passed/failed)

Critical Priority Items:

- Schema validation (prevents most AI errors)
- Test database (catch failures before production)
- Business rule codification (governance requires rules, not just guidelines)

Principle 3: Continuous Observability

Every AI decision must be traceable. Here's the data infrastructure for observability:

Data Lineage Tracking

- [ ] **Data lineage tool deployed** (Apache Atlas, Amundsen, DataHub, or cloud-native)
- [ ] **Source-to-destination tracking enabled** (know where every data point came from)
- [ ] **Transformation logic documented** (how data changes from source to AI input)
- [ ] **Lineage queryable** (can trace backwards from AI output to original source)
- [ ] **Lineage visualization available** (see data flow graphically)

Audit Trail Infrastructure

- [ ] **Audit database or logging system deployed** (separate from production database)
- [ ] **All AI interactions logged** (inputs, outputs, timestamps, user IDs)
- [ ] **Prompts and context logged** (what the AI saw when making decision)
- [ ] **Validation results logged** (which checks passed/failed)
- [ ] **Execution results logged** (what happened after AI decision)

Observability Platform

- [ ] **Observability tool integrated** (LangSmith, Weights & Biases, MLflow, or custom)
- [ ] **Traces searchable and filterable** (find specific interactions by date, user, query type)

- [ ] **Performance metrics tracked** (latency, token usage, error rates)
- [ ] **Anomaly detection configured** (alerts for unusual patterns)
- [ ] **Dashboards created** (technical and executive views)

Retention and Compliance

- [ ] **Audit log retention policy defined** (how long to keep logs - typically 1-7 years)
- [ ] **Log storage scaled for retention** (audit logs grow fast - plan for TB-scale)
- [ ] **Compliance requirements met** (HIPAA requires 6 years, SOX varies)
- [ ] **Log access controlled** (who can see audit trails)
- [ ] **Log immutability enforced** (can't modify past decisions)

Critical Priority Items:

- Audit logging (can't govern what you can't trace)
- Observability platform (LangSmith or equivalent)
- Retention policy (compliance requirement)

Principle 4: Stateful Control

AI workflows need to maintain context and state. Here's the required infrastructure:

State Management

- [ ] **State store deployed** (Redis, DynamoDB, or in-memory cache)
- [ ] **Conversation history tracked** (multi-turn interactions remembered)
- [ ] **User context maintained** (preferences, previous queries, session data)
- [ ] **State persistence configured** (survives restarts, not just in-memory)
- [ ] **State expiration defined** (when to clear old state - typically hours to days)

Workflow Orchestration

- [ ] **Orchestration framework deployed** (LangGraph, Temporal, Airflow, Prefect)
- [ ] **Workflow state managed** (know where user is in multi-step process)
- [ ] **Conditional logic supported** (different paths based on AI output)
- [ ] **Error handling defined** (what happens when workflow fails mid-step)
- [ ] **Workflow resumption enabled** (can pick up where you left off after failure)

Data Consistency

- [ ] **Transaction management implemented** (ensure data consistency across steps)
- [ ] **Rollback capability** (undo changes if workflow fails)
- [ ] **Idempotency enforced** (same request twice = same result, no duplicates)
- [ ] **Concurrency handling** (multiple workflows don't interfere)

Critical Priority Items:

- State store (stateful control requires state storage)
- Orchestration framework (workflows need orchestration)
- Transaction management (prevent data inconsistency)

Principle 5: Human-Verifiable Outputs

Executives and stakeholders need to understand AI decisions. Here's the data infrastructure:

Explainability Data

- [ ] **AI reasoning captured** (chain-of-thought, decision factors)
- [ ] **Supporting evidence stored** (data points that influenced decision)
- [ ] **Confidence scores tracked** (how certain was the AI?)
- [ ] **Alternative options logged** (what else did AI consider?)

Presentation Layer Data

- [ ] **Output formatting standardized** (consistent structure for displaying results)
- [ ] **Natural language explanations generated** (translate technical to business language)
- [ ] **Visual data available** (charts, graphs for dashboards)
- [ ] **Comparison data accessible** (how does this decision compare to historical?)

Access and Permissions

- [ ] **Role-based access configured** (different stakeholders see different views)
- [ ] **Executive dashboards created** (high-level metrics)
- [ ] **Technical dashboards created** (detailed diagnostics)
- [ ] **Audit access controlled** (who can see sensitive decision data)

Critical Priority Items:

- AI reasoning capture (transparency requirement)
- Executive dashboards (strategic visibility)

- Role-based access (security and relevance)

Part 2: RAG Implementation Prerequisites

Before implementing Retrieval-Augmented Generation, ensure these data components are ready:

Knowledge Base Preparation

- [ ] **Source documents identified** (what knowledge will AI reference?)
- [ ] **Documents digitized and accessible** (PDFs, docs, databases accessible programmatically)
- [ ] **Document quality verified** (accurate, up-to-date, authoritative)
- [ ] **Document structure understood** (sections, headers, metadata)
- [ ] **Update frequency defined** (how often does knowledge change?)

Document Processing Pipeline

- [ ] **Document ingestion automated** (can add new documents without manual work)
- [ ] **Text extraction tested** (PDFs, images, complex formats handled)
- [ ] **Chunking strategy defined** (how to split long documents - typically 500-1000 tokens)
- [ ] **Chunk overlap configured** (prevent context loss at boundaries - typically 50-100 tokens)
- [ ] **Metadata extraction automated** (titles, dates, authors, categories)

Vector Database Setup

- [ ] **Vector database selected** (Azure AI Search, Pinecone, Weaviate, Chroma, Qdrant)
- [ ] **Embedding model chosen** (OpenAI text-embedding-3, sentence transformers)
- [ ] **Embedding dimensions configured** (typically 768, 1024, or 1536)
- [ ] **Index created and populated** (documents embedded and stored)
- [ ] **Search performance tested** (sub-second retrieval at scale)

Retrieval Configuration

- [ ] **Similarity metric selected** (cosine similarity, dot product, euclidean)
- [ ] **Number of results (k) tuned** (typically 3-10 relevant chunks)
- [ ] **Relevance threshold defined** (minimum similarity score - typically 0.7-0.8)
- [ ] **Re-ranking implemented** (optional: improve retrieval quality with cross-encoder)
- [ ] **Retrieval fallback defined** (what if no relevant chunks found?)

Context Window Management

- [ ] **Token counting implemented** (track context size to avoid exceeding limits)
- [ ] **Context prioritization defined** (which chunks matter most if context is limited?)

- [ ] **Context compression considered** (summarize less relevant chunks if needed)
- [ ] **Context formatting standardized** (how to present chunks to AI model)

Monitoring and Maintenance

- [ ] **Retrieval quality measured** (are retrieved chunks actually relevant?)
- [ ] **Embedding drift monitored** (does retrieval degrade over time?)
- [ ] **Index refresh scheduled** (how often to re-embed updated documents)
- [ ] **Performance metrics tracked** (latency, accuracy, token usage)

Critical Prerequisites:

- Knowledge base identified and prepared (RAG needs something to retrieve)
- Vector database deployed (can't do semantic search without it)
- Chunking strategy defined (bad chunking = bad retrieval)
- Retrieval tested and tuned (relevance is everything)

Part 3: Data Quality Validation Framework

A comprehensive framework for validating data quality before AI consumption:

Data Completeness Checks

- [ ] **Required fields validation** (no null values in critical columns)
- [ ] **Record completeness validation** (minimum fields populated per record)
- [ ] **Referential integrity checks** (foreign keys point to existing records)
- [ ] **Historical completeness** (data available for required time periods)

Data Accuracy Checks

- [ ] **Format validation** (emails valid, phone numbers formatted correctly)
- [ ] **Range validation** (values within acceptable bounds)
- [ ] **Pattern validation** (regex patterns for structured fields)
- [ ] **Cross-field validation** (related fields logically consistent)
- [ ] **Duplicate detection** (identify and flag duplicate records)

Data Consistency Checks

- [ ] **Schema consistency** (data matches defined schema)
- [ ] **Type consistency** (correct data types throughout)

- [ ] **Naming consistency** (standardized naming conventions)
- [ ] **Unit consistency** (measurements in consistent units)
- [ ] **Temporal consistency** (dates make logical sense, not future dates in history)

Data Freshness Checks

- [ ] **Update timestamp validation** (data recently updated as expected)
- [ ] **Staleness detection** (identify data not updated within SLA)
- [ ] **Real-time validation** (streaming data arrives within latency requirements)
- [ ] **Scheduled refresh validation** (batch updates complete successfully)

Data Quality Scoring

- [ ] **Quality metrics defined** (what defines "high quality" - typically % passing validations)
- [ ] **Scoring methodology implemented** (aggregate scores per table/dataset)
- [ ] **Quality thresholds set** (minimum acceptable quality - typically >95%)
- [ ] **Quality reporting automated** (dashboards showing current quality state)
- [ ] **Quality trends tracked** (is quality improving or degrading?)

Automated Remediation

- [ ] **Common errors auto-corrected** (known issues fixed automatically where safe)
- [ ] **Quarantine process defined** (how to handle data that fails validation)
- [ ] **Manual review workflow** (process for human review of edge cases)
- [ ] **Remediation tracking** (log all corrections made)

Integration with AI Pipeline

- [ ] **Pre-AI validation gates** (data validated before feeding to AI)
- [ ] **Validation failure handling** (what happens if data quality too low)
- [ ] **Quality metadata passed to AI** (AI knows quality score of its inputs)
- [ ] **Post-AI validation** (outputs also validated for quality)

Critical Framework Components:

- Completeness and accuracy checks (foundation of quality)
- Quality scoring (need objective measurement)
- Pre-AI validation gates (prevent "garbage in, garbage out")

Part 4: Data Lineage and Audit Trail Requirements

Complete audit trails from data source through AI decision:

Source System Lineage

- [ ] **All data sources documented** (where does data originate?)
- [ ] **Source system metadata captured** (version, update frequency, owner)
- [ ] **Extract timestamp logged** (when was data pulled from source)
- [ ] **Extract method documented** (API, database query, file upload, CDC)

Transformation Lineage

- [ ] **All transformations documented** (what changes to data between source and AI)
- [ ] **Transformation code versioned** (which version of ETL/script was used)
- [ ] **Transformation logic explained** (business meaning of each transformation)
- [ ] **Data quality changes tracked** (how did quality change through pipeline)
- [ ] **Schema evolution tracked** (when did schema change, why, by whom)

AI Interaction Lineage

- [ ] **Input data logged** (exact data fed to AI - may need to sample if large)
- [ ] **Prompt logged** (what prompt was used, including context)
- [ ] **Model version logged** (which model, which version)
- [ ] **Parameters logged** (temperature, max_tokens, top_p, etc.)
- [ ] **Retrieved context logged** (for RAG: which chunks were retrieved)

Decision Lineage

- [ ] **AI output logged** (exact response from model)
- [ ] **Reasoning captured** (chain-of-thought if used)
- [ ] **Confidence scores** (how confident was AI in this decision)
- [ ] **Validation results** (which validation checks passed/failed)
- [ ] **Human override logged** (if human modified AI decision)

Execution Lineage

- [ ] **Execution results logged** (what happened after AI decision - query run, action taken)
- [ ] **Execution timestamp** (when was decision executed)
- [ ] **Execution duration** (how long did it take)
- [ ] **Execution errors** (any errors during execution)
- [ ] **Impact measured** (business outcome of this decision)

User and Access Lineage

- [ ] **User ID logged** (who initiated this AI interaction)
- [ ] **User role logged** (what permissions did they have)
- [ ] **Access timestamp** (when did interaction occur)
- [ ] **IP address/location** (where did request originate - if relevant for compliance)
- [ ] **Session ID** (tie multiple interactions to same session)

Audit Query Capabilities

- [ ] **Search by user** (find all AI decisions by specific user)
- [ ] **Search by date range** (find decisions in time window)
- [ ] **Search by data source** (which decisions used specific data)
- [ ] **Search by outcome** (find all failures, or all high-confidence decisions)
- [ ] **Reconstruct decision** (given an audit ID, replay entire decision process)

Compliance and Retention

- [ ] **Regulatory requirements mapped** (HIPAA, SOX, GDPR, etc.)
- [ ] **Retention periods defined** (typically 1-7 years depending on regulation)
- [ ] **Immutability enforced** (audit logs cannot be modified or deleted)
- [ ] **Access controls strict** (only authorized personnel can access audit logs)
- [ ] **Encryption at rest and in transit** (protect sensitive audit data)
- [ ] **Regular audit reviews** (periodic review of audit logs for anomalies)

Critical Audit Requirements:

- AI interaction logging (input, output, model, context)
- Decision lineage (can reconstruct any decision)
- Immutability and retention (compliance requirement)
- Search capabilities (audit logs useless if not searchable)

Part 5: Phase 0 Implementation Guide

Your 8-12 week roadmap to build governance-ready data infrastructure:

Week 1-2: Assessment and Planning

Goal: Understand current state and prioritize gaps

- [ ] **Inventory existing data infrastructure** (databases, pipelines, tools)
- [ ] **Document current data flows** (where data comes from, where it goes)

- [ ] **Assess against this checklist** (identify gaps)
- [ ] **Prioritize critical gaps** (focus on items marked "Critical Priority")
- [ ] **Define success metrics** (what "governance-ready" means for you)
- [ ] **Assign team roles** (who owns each component)
- [ ] **Create implementation timeline** (8-12 week plan with milestones)
- [ ] **Secure budget and resources** (tooling, cloud resources, headcount)

Deliverable: Phase 0 project plan with prioritized backlog

Week 3-4: Schema and Knowledge Base Foundation

Goal: Document and index your data for context-aware AI

- [ ] **Complete schema documentation** (tables, columns, relationships)
- [ ] **Create data dictionary** (business definitions)
- [ ] **Deploy vector database** (Azure AI Search, Pinecone, or Weaviate)
- [ ] **Index schema for RAG** (embed and store schema metadata)
- [ ] **Test schema retrieval** (can you find relevant tables with natural language?)
- [ ] **Document business rules** (what rules must AI follow?)
- [ ] **Create rule repository** (centralized store for governance rules)

Deliverable: Searchable schema index, documented business rules

---

Week 5-6: Data Quality and Validation

Goal: Build validation framework to catch AI errors

- [ ] **Define data quality rules** (completeness, accuracy, consistency)
- [ ] **Implement validation library** (Great Expectations, Pydantic, or custom)
- [ ] **Create test database** (for dry-run validation)
- [ ] **Build validation pipeline** (automate quality checks)
- [ ] **Set quality thresholds** (minimum acceptable quality levels)
- [ ] **Create quality dashboard** (visualize data quality metrics)
- [ ] **Test validation with sample AI outputs** (does it catch expected errors?)

Deliverable: Automated data quality validation framework

Week 7-8: Observability and Audit Infrastructure

Goal: Enable traceability of all AI decisions

- [ ] **Deploy observability platform** (LangSmith, MLflow, or custom)
- [ ] **Set up audit database** (separate from production)
- [ ] **Configure logging** (all AI interactions, inputs, outputs)
- [ ] **Implement lineage tracking** (source to AI to execution)
- [ ] **Define retention policy** (how long to keep logs)
- [ ] **Create audit dashboards** (technical and executive views)
- [ ] **Test end-to-end traceability** (can you reconstruct a decision?)

Deliverable: Full audit trail for AI decisions, searchable observability

---

Week 9-10: State Management and Orchestration

Goal: Enable stateful AI workflows with governance controls

- [ ] **Deploy state store** (Redis, DynamoDB for session state)
- [ ] **Implement orchestration** (LangGraph, Temporal for workflows)
- [ ] **Configure state persistence** (survives restarts)
- [ ] **Define workflow logic** (multi-step processes with governance checkpoints)
- [ ] **Implement error handling** (graceful failures, rollback)
- [ ] **Test stateful workflows** (multi-turn interactions work correctly?)

Deliverable: Orchestrated, stateful AI workflows with governance embedded

---

Week 11-12: Integration, Testing, and Documentation

Goal: Integrate all components and prepare for AI governance implementation

- [ ] **Integrate all Phase 0 components** (schema, validation, audit, state, orchestration)
- [ ] **End-to-end testing** (simulate full AI workflow with governance)
- [ ] **Performance testing** (latency, throughput at expected scale)
- [ ] **Security testing** (access controls, encryption working)
- [ ] **Create runbooks** (how to operate and maintain infrastructure)
- [ ] **Document architecture** (diagrams, component descriptions)
- [ ] **Train team** (ensure team can operate and troubleshoot)
- [ ] **Define handoff to Phase 1** (ready for AI governance implementation)

Deliverable: Production-ready data infrastructure, ready for AI governance layers

---

Phase 0 Success Criteria

You're ready for AI governance implementation (Phases 1-4) when:

✅ **Schema is documented and searchable** (context-aware generation possible)
✅ **Validation framework catches errors** (multi-layer validation operational)
✅ **All AI interactions are logged** (continuous observability in place)
✅ **State management works** (stateful control enabled)
✅ **Dashboards show AI activity** (human-verifiable outputs available)
✅ **Data quality is measured** (validation framework operational)
✅ **Audit trails are complete** (lineage tracked source to execution)
✅ **Team is trained** (can operate infrastructure)

Common pitfalls to avoid:

❌ **Skipping documentation** (undocumented schemas = no context for AI)
❌ **No test environment** (validating in production is dangerous)
❌ **Insufficient logging** (you can't govern what you can't trace)
❌ **Manual processes** (won't scale, governance requires automation)
❌ **No retention policy** (audit logs fill disk, or get deleted too soon)

Next Steps

Immediate Actions (This Week)

1. **Complete the assessment** - Go through this checklist and mark your current state
2. **Identify your top 5 gaps** - What's blocking governance most?
3. **Prioritize quick wins** - What can you accomplish in 2-4 weeks?
4. **Assign ownership** - Who will drive Phase 0?

Getting Help

Need guidance implementing Phase 0?

Schedule a free Data Engineering Assessment with Pendoah. We'll:

- Review your infrastructure against this checklist

- Identify critical gaps blocking AI governance
- Design your specific 8-12 week Phase 0 roadmap
- Share architecture patterns from similar implementations

Schedule Assessment →

Resources

Tools mentioned in this checklist:

- **Vector Databases:** Azure AI Search, Pinecone, Weaviate, Chroma, Qdrant
- **Data Quality:** Great Expectations, Pydantic, Apache Griffin
- **Observability:** LangSmith, Weights & Biases, MLflow
- **Orchestration:** LangGraph, Temporal, Airflow, Prefect
- **Lineage:** Apache Atlas, Amundsen, DataHub
- **State Management:** Redis, DynamoDB, Memcached

Further reading:

- **Blog Posts**
- **Whitepapers**

About Pendoah

Pendoah helps mid-market companies (50-1,000 employees) build AI systems with governance baked in from day one. Our Data Engineering & Integration services provide the foundation that makes AI governance possible.

Services:

- **Data Engineering & Integration**
- **AI Strategy & Development**
- **Custom AI Solutions**
- **MLOps & Model Governance**

Contact: hello@pendoah.com
Website: www.pendoah.com