Palantir Data Engineering Certification

Volume 4 — Data Quality, Governance, Security & Lineage

How to Use This Volume

This volume is written to **exactly match the format, tone, and depth of Volume 3**.

It is: - Conceptual, not tool-manual–like - Explanatory rather than procedural - Focused on *why Foundry works this way*, not just *what to do*

You should read this volume as a continuation of the semantic and operational model introduced by the Ontology.

Chapter 19 — Why Governance Exists in Foundry

19.1 Governance Is an Architectural Concern

In many data platforms, governance is treated as an external constraint imposed by policy, compliance teams, or documentation. In Foundry, governance is treated as an **architectural property of the system itself**.

Foundry was designed for environments where: - Data is shared across many teams - Decisions have operational consequences - Mistakes must be explainable after the fact

In such environments, relying on human discipline is insufficient. Governance must be **enforced by default**, not requested as a favor.

19.2 Governance vs Control

Governance is often confused with control.

Control restricts behavior. Governance enables **safe autonomy**.

Foundry’s goal is not to prevent users from accessing data, but to ensure that whatever access they do have is: - Appropriate - Auditable - Reversible

This distinction explains why Foundry emphasizes lineage, immutability, and layered permissions instead of centralized approval workflows.

Chapter 20 — Data Quality as a Property of Curated Data

20.1 What “Data Quality” Means in Foundry

In Foundry, data quality is not defined by cleanliness alone. High-quality data is data that: - Can be trusted for decision-making - Has clearly defined assumptions - Can be reproduced and audited

Quality is therefore **contextual**, not absolute.

20.2 Raw Data Is Allowed to Be Wrong

Raw datasets exist to preserve evidence. As such, they are allowed to contain: - Missing values - Invalid records - Duplicates - Inconsistent schemas

Correcting raw data would destroy evidence and undermine auditability.

Raw data answers the question:

> “What did the source system say?”

Not:

> “What should the data have said?”

20.3 Curated Data Must Be Defensible

Curated datasets represent organizational belief. They answer:

> “Given the evidence, what do we believe to be correct?”

As a result, curated datasets must: - Enforce schema and types - Resolve duplicates - Handle invalid records explicitly - Encode business rules transparently

Silently passing bad data downstream is a governance failure.

Chapter 21 — Validation and Failure Semantics

21.1 Validation Is a Design Decision

Validation is not simply about catching errors. It is about deciding **what should happen when assumptions are violated**.

Every validation rule should answer:

> “What is the risk if this data is wrong?”

21.2 Fail-Fast Semantics

Fail-fast validation is appropriate when: - Data feeds automated decisions - Incorrect values would propagate harm - Regulatory or financial accuracy is required

In these cases, pipeline failure is safer than silent corruption.

21.3 Fail-Soft Semantics

Fail-soft validation may be acceptable when: - Data is exploratory - Availability is prioritized over correctness - Errors can be tolerated temporarily

However, fail-soft behavior must still be observable. Hidden errors are unacceptable.

Chapter 22 — Access Control as a Layered System

22.1 Why Access Control Is Layered

No single access control mechanism is sufficient for complex organizations.

Foundry therefore enforces permissions at multiple levels: - Dataset - Column - Row - Ontology object - Ontology action

Each layer reduces risk and limits blast radius.

22.2 Dataset-Level Permissions

Dataset-level permissions control who can: - Read data - Write data - Promote changes

Raw datasets typically have restricted readership. Curated datasets are more broadly accessible but more strictly governed.

22.3 Column- and Row-Level Security

Column-level security protects sensitive attributes such as PII. Row-level security restricts which records a user can see.

These controls allow safe data sharing without duplication.

Chapter 23 — Ontology as a Governance Boundary

23.1 Why Ontology Is Safer Than Tables

Tables expose schemas, join logic, and internal representations.

Ontology exposes: - Business entities - Meaningful properties - Allowed actions

This abstraction dramatically reduces misuse and misinterpretation.

23.2 Ontology Permissions

Ontology permissions allow organizations to: - Control who can view entities - Restrict sensitive properties - Limit who can execute actions

This enables self-service without sacrificing governance.

Chapter 24 — Lineage and Organizational Memory

24.1 What Lineage Captures

Lineage captures: - Upstream dependencies - Transformation logic - Dataset versions

It represents the **organizational memory** of how data came to exist.

24.2 Why Lineage Is Central to Trust

When data changes, lineage allows teams to answer: - What changed? - Why did it change? - Who is affected?

Without lineage, trust erodes rapidly.

Chapter 25 — Governance Anti-Patterns

25.1 Common Mistakes

Common governance failures include: - Exposing raw data directly to business users - Encoding access logic in code - Relying on documentation instead of enforcement

These patterns scale poorly and are frequently tested in the exam.

End of Volume 4

This volume completes the governance model that underpins Foundry. The next volume focuses on operating and debugging systems built on these principles.