

The background features a dark, 3D grid of lines. In the center, there is a stack of several translucent, glowing yellow cubes. A bright vertical beam of light passes through the center of these cubes, creating a lens flare effect. The overall aesthetic is futuristic and digital.

Master Data Management and Golden Records

Mario Noioso · January 2026

DATA , AI, AND THE REAL WORLD

Master Data Management and Golden Records

Author: Mario Noioso

Published: January 2026

© 2026 Mario Noioso. All rights reserved.

marionoioso.com

The idea that Master Data Management is about creating a “single source of truth” is misleading. In real systems, truth does not exist upfront, It is not discovered, It is not inherited from a system.

Truth is constructed, negotiated, and revised over time. MDM is not about choosing the best system. It is about defining **how truth is determined when systems disagree** when data arrives incomplete, and when reality changes.

This document describes a pragmatic, production-grade Master Data Management pattern based on **Attribute-Centric Golden Records** designed for complex, heterogeneous environments where:

- multiple systems describe the same real-world entities,
- no system is complete on its own,
- data quality varies by source,
- governance matters more than convenience.

The goal is not theoretical purity. The goal is **operational truth that can be explained, audited, and defended**

Executive Summary

The pattern described in this document is based on a simple but often misunderstood idea: truth is decided at attribute level, not at record or system level.

Instead of selecting a “master system” or merging records blindly, the system:

- **ingests data without assumptions,**
- **resolves identity explicitly,**
- **applies survivorship rules per attribute,**
- **composes a Golden Record as a governed view,**
- **preserves history and uncertainty by design.**

This approach avoids silent data corruption, supports regulatory requirements, and scales to real enterprise complexity.

At a glance, the pattern is defined by:

- **Attribute-Centric Golden Records**
- **Split-and-Link ingestion**
- **Explicit survivorship rules**
- **Identity resolution before merge**
- **Temporal versioning (SCD Type 2)**
- **Governance-first, not system-first design**

Table of Contents

- [How truth is constructed, governed, and revised over time](#)
- [Executive Summary](#)
- [Part I – The core principle: truth is attribute-centric](#)
 - [Bootstrapping identity: when data arrives from the Web](#)
 - [Identity resolution is a decision process, not a lookup](#)
 - [From candidate to Golden: composing truth](#)
 - [Change, conflict, and time](#)
 - [Why this pattern works](#)
 - [Why MongoDB fits naturally \(without marketing\)](#)
 - [Final thought](#)
- [Part II – Architecture & Implementation Patterns](#)
 - [Chapter I – Ingestion & Split-and-Link Architecture](#)
 - [Chapter II – Identity Resolution & Matching](#)
 - [Chapter III – Survivorship & Conflict Resolution](#)
 - [Chapter IV – Data Quality & Validation](#)
 - [Chapter V – Temporal Versioning & History \(SCD Type 2\)](#)
- [Part III – Final Considerations](#)

This first part deliberately avoids implementation details. Its purpose is to establish the mental model of truth governance before introducing any architecture.

Part I – The core principle: truth is attribute-centric

The most common MDM mistake is thinking in terms of records.

Questions like:

- Which system is authoritative?
- Which record wins?
- Which source is the master?

are already the wrong questions.

In the real world:

- one system may be authoritative for legal identity,
- another for contractual responsibility,
- another for contact information,
- another for user-entered preferences.

No single system owns the full truth. Truth is therefore **not record-centric**. Truth is **attribute-centric**. A Golden Record is not selected. It is **composed**

Each attribute in the Golden Record exists because:

- its identity is known,
- its source is governed,
- its validity is explainable.

The scenario: three sources, one real person

To ground the pattern in reality, consider a simple but representative scenario.

Three systems describe the same real person.

National Registry (Anagrafe)

Legal authority for identity.

High data quality.

Provides fiscal code, name, residence.

SAP

Authoritative for contracts and operational roles.

Provides contract data and the DEC (Director of Contract Execution).

Contains partial personal data embedded in business documents.

Web Booking Application

User-facing.

Data entered directly by the person.

Provides email, phone, preferences, mood.

Inherently error-prone.

All three systems describe the same person.

None of them describes the full truth.

The Golden Record lifecycle (high level)

The Golden Record does not appear magically. It is the result of a controlled lifecycle. This lifecycle is conceptual. Each step will be revisited and implemented explicitly in the technical chapters that follow

1. Data is ingested without assumptions
2. Business entities are identified and separated
3. Identity is resolved explicitly
4. Survivorship rules are applied
5. The Golden Record is composed
6. History is preserved
7. Conflicts are governed, not hidden

Each step is intentional.

Skipping steps is how MDM systems fail.

Bootstrapping identity: when data arrives from the Web

The most fragile data always comes from the Web.

Web data is:

- user-entered,
- partial,
- sometimes wrong,
- often missing strong identifiers.

A critical rule applies:

Web data never updates a Golden Record directly.

When a person submits a form, the system does not say:

“This is Giuseppe Verdi, update him.”

It says:

“This is a candidate identity.”

At this stage:

- no Golden Record is touched,
- no identity is assumed,
- no overwrite is possible.

This separation protects identity from accidental corruption.

Identity resolution is a decision process, not a lookup

The system now answers a single question:

Does this candidate correspond to an existing Golden Person?

This is not:

- “find by email”
- “find by name”
- “first match wins”

Identity resolution is a **scored decision process**

Each attribute contributes a signal:

- strong identifiers (when present),
- medium signals (email, phone),
- weak signals (name, address).

Signals are combined into a confidence score that expresses **automate the decision** the probability of truth.

how safe it is to

If confidence is:

- high → controlled association
- ambiguous → manual stewardship
- low → provisional Golden Record

Uncertainty is preserved, not hidden.

From candidate to Golden: composing truth

Once identity is established, survivorship rules apply.

Survivorship rules are defined **before any merge happens**

Examples:

- fiscal code → National Registry
- legal name → National Registry
- work email → SAP
- personal email → Web
- mood → Web

These rules are explicit.

They are not inferred.

They are governance, encoded as architecture.

The Golden Record is then composed as a **governed view**

- each attribute has a value,
- each value has a source,
- losing values are preserved as shadow data.

Nothing is overwritten without trace.

Change, conflict, and time

Reality changes.

People move.

Emails change.

Roles evolve.

A production-grade MDM system must treat change as normal.

This pattern applies:

- conflict resolution **after** identity resolution,
- survivorship strategies per attribute,
- manual intervention when automation is unsafe.

Every meaningful change produces a new version.

The system remembers:

- what was true,
- when it was true,
- why it changed.

This enables:

- auditability,
- point-in-time reconstruction,
- legal defensibility.

Truth changes.

Memory does not.

Why this pattern works

This approach works because it:

- models reality instead of forcing it into tables,
- treats conflicts as first-class citizens,
- separates ingestion, identity, and merge,
- makes governance explicit,
- preserves history by design.

Most failed MDM implementations collapse these steps into:

“If something matches, update.”

That is not MDM.

That is silent data corruption.

Why MongoDB fits naturally (without marketing)

This pattern requires:

- flexible document models,
- embedded lineage and metadata,
- evolving schemas,
- partial updates without rigid joins,
- efficient temporal versioning.

MongoDB does not impose this model.

It simply does not resist it.

The database stays out of the way while governance does the work.

Final thought

A Golden Record is not found.

It is constructed.

When you stop asking:

“Which system is right?”

and start asking:

“Who is right for this attribute, at this time?”

you are no longer managing data.

You are managing truth.

The following chapters describe how the principles introduced above are implemented in a real production-grade architecture.

Part II – Architecture & Implementation Patterns

Chapter I – Ingestion & Split-and-Link Architecture

Why ingestion is the most underestimated part of MDM

Most MDM failures do not originate in identity resolution.

They originate earlier.

They originate at ingestion time, when incoming data is already:

- interpreted too early,
- merged too soon,
- normalized without context,
- or written directly into a master structure.

Once this happens, everything downstream becomes fragile.

Identity resolution becomes probabilistic noise.

Conflict resolution becomes guesswork.

History becomes unreliable.

This chapter describes a production-grade ingestion pattern designed around one core principle:

Data must enter the system without making any assumption about truth.

The non-negotiable rule of MDM ingestion

Before going any further, we need to establish a rule that cannot be bent later:

No incoming data is ever written directly into a Golden Record.

Not from SAP.

Not from Anagrafe.

Not from the Web.

Every piece of data enters the system as **evidence** not as truth.

This single rule is what makes everything else possible.

Architectural overview – the Split-and-Link ingestion pattern

Everything described in this paragraph happens before identity resolution, survivorship, or data quality enforcement

At a high level, ingestion is composed of four irreversible steps:

1. Raw ingestion (staging)
2. Semantic split into business entities
3. Entity-level processing in a defined order
4. Linking via stable Golden identifiers

Each step has a single responsibility.

Each step is designed to prevent contamination of master data.

Step 1 – Raw ingestion: preserving the original signal

All incoming messages, regardless of source, are first stored in a raw staging area.

This area is append-only.

No transformation.

No normalization.

No correction.

Only minimal system metadata is added.

Conceptually:

```
{
  "_id": "raw_event_001",
  "source": "Web",
  "ingested_at": "2024-03-01T10:12:00Z",
  "status": "PENDING",
  "payload": { ... }
}
```

The purpose of this stage is not storage.

It is **forensic integrity**

If something goes wrong later, this is the only place where the original signal still exists.

Any architecture that skips this step cannot be audited.

Step 2 – The first real decision: split by business meaning

Most enterprise systems emit documents, not entities.

SAP emits transactions.

The split step answers one question only:

Which real-world entities are described by this payload?

Example: a SAP message may describe:

- a Contract
- a Person acting as DEC
- a relationship between them

This is not parsing.

This is semantic analysis.

The output of this step is **multiple entity candidates** for one record.

A critical architectural constraint: entity order matters

Split-and-Link is not symmetric.

Some entities must exist before others can be created.

In almost every enterprise domain:

- Persons must exist before Contracts
- Organizations must exist before Relationships
- Reference data must exist before facts

For this reason, the split process is **ordered** not parallel.

A typical execution order:

1. Person
2. Organization
3. Contract
4. Relationship

This order is enforced explicitly.

Not inferred.

Step 3 – First-contact scenario: Web data creates the initial Golden Record

To understand why this pattern works, start from the weakest possible signal.

The Web.

A user fills a form with partial, possibly wrong data.

```
{
  "source": "Web",
  "payload": {
    "first_name": "Giuseppe",
    "last_name": "Verdi",
    "email": "giuseppe.verdi@gmail.com",
    "phone": "3331234567"
  }
}
```

At ingestion time:

- no identity is assumed,
- no match is forced,
- no external authority is available.

Identity resolution is attempted.

No existing Golden Record is found with sufficient confidence.

The correct action is not to wait.

The correct action is to create a **provisional Golden Record**

This is how the first Golden Record is born.

```
{
  "_id": "GOLD-PER-001",
  "entity_type": "Person",
  "origin": "Web",
  "confidence": "LOW",
  "view": {
    "first_name": "Giuseppe",
    "last_name": "Verdi",
    "email": "giuseppe.verdi@gmail.com",
    "phone": "3331234567"
  },
  "lineage": {
    "first_name": { "source": "Web" },
    "last_name": { "source": "Web" },
    "email": { "source": "Web" },
    "phone": { "source": "Web" }
  }
}
```

This record is not “true”.

It is **the best available truth at that moment**

And it is explicitly marked as such.

Step 4 – Authoritative data arrives later: enrichment, not overwrite

Time passes.

Data from Anagrafe arrives.

```
{
  "source": "Anagrafe",
  "payload": {
    "cf": "VRDGPP80A01H501U",
    "first_name": "Giuseppe",
    "last_name": "Verdi",
    "residence": "Via Roma 10"
  }
}
```

Identity resolution now has a strong identifier.

The system links this payload to the existing Golden Record.

No record is replaced.

No delete happens.

Instead, survivorship rules are applied at attribute level.

Result:

```
{
  "_id": "GOLD-PER-001",
  "entity_type": "Person",
  "view": {
    "cf": "VRDGPP80A01H501U",
    "first_name": "Giuseppe",
    "last_name": "Verdi",
    "residence": "Via Roma 10",
    "email": "giuseppe.verdi@gmail.com",
    "phone": "3331234567"
  },
  "lineage": {
    "cf": { "source": "Anagrafe" },
    "first_name": { "source": "Anagrafe" },
    "last_name": { "source": "Anagrafe" },
    "residence": { "source": "Anagrafe" },
    "email": { "source": "Web" },
    "phone": { "source": "Web" }
  }
}
```

The Web data did not disappear.

It was **contextualized**

This is why early ingestion must not contaminate master data.

Step 5 – Split-and-Link with composite payloads (SAP case)

Later, SAP sends a contract message embedding personal data.

This payload is split again:

- Person candidate (DEC)
- Contract candidate
- Relationship

The Person step executes first.

If the fiscal code matches, the existing Golden Record is reused.

Only attributes governed by SAP are considered (for example, work email).

Then, and only then, the Contract is created.

```
{
  "_id": "GOLD-CTR-001",
  "entity_type": "Contract",
  "view": {
    "contract_id": "CTR-2024-ALPHA",
    "amount": 150000,
    "dec": {
      "person_id": "GOLD-PER-001",
      "role": "DEC"
    }
  }
}
```

The contract never stores a person by value.

It stores a **link to truth**

Why Split-and-Link is irreversible (and must be)

Once entities are split and linked by Golden IDs:

- identity becomes stable,
- relationships become precise,
- queries become deterministic.

Reversing this later is almost impossible.

That is why this pattern must be applied **at ingestion time** not retrofitted.

What this chapter deliberately does not cover

This chapter does not explain:

- how identity resolution works internally,
- how conflicts are resolved,
- how history is managed,
- how data quality gates are enforced.

Those topics depend on ingestion being correct.

They are covered in the following chapters.

The architectural takeaway

If ingestion is treated as plumbing, MDM fails.

If ingestion is treated as a **semantic decision point**, everything else becomes simpler.

Split early.

Link explicitly.

Delay truth until governance can act.

That is how you prevent data from lying to you before you even start asking questions.

Chapter II – Identity Resolution & Matching

Determining who is who without assuming identity

Why identity resolution is where most MDM systems silently break

After ingestion and split-and-link, the system is full of **candidate entities**

Some are new.

Some refer to the same real-world person.

Some look similar but are not the same.

At this point, many systems make a fatal shortcut:

- they rely on a single identifier,
- they assume uniqueness,
- they collapse matching and merging into one step.

This chapter describes a different approach, based on one principle:

Identity resolution is a decision process, not a lookup.

What identity resolution is (and what it is not)

Identity resolution answers one question only:

Does this candidate entity correspond to an existing Golden entity?

It does not:

- merge data,
- overwrite attributes,
- resolve conflicts,
- update truth.

Its output is not a record.

Its output is a **decision**

This separation is intentional and non-negotiable.

The three possible outcomes of identity resolution

For any candidate entity, identity resolution must produce exactly one of these outcomes:

1. **Link to an existing Golden Record**
2. **Create a new Golden Record**
3. **Escalate for manual resolution**

There is no fourth option.

Anything else introduces ambiguity into the system.

Identity is never binary

In the real world:

- identifiers are missing,
- data is incomplete,
- spelling varies,
- systems disagree.

Therefore, identity resolution cannot be binary.

It must be **signal-based**

Each attribute contributes evidence.

No single attribute decides identity on its own.

Matching signals: evidence, not truth

A matching signal is a bounded contribution toward identity confidence.

Typical signals include:

- Strong signals
 - fiscal code
 - national ID
 - unique external identifier
- Medium signals
 - email
 - phone number
- Weak signals
 - first name
 - last name
 - address

Each signal has three properties:

- **strength** (how much it can influence the decision),
- **reliability** (how often it lies),
- **independence** (how correlated it is with other signals).

Weak signals are intentionally capped.

They can support identity, but never establish it.

Why “find by email” is not identity resolution

Email looks unique.

In practice, it is not.

- shared mailboxes exist,
- personal emails change,
- typos are common,
- the same email can appear in different contexts.

Treating email as a primary key produces:

- false positives,
- irreversible merges,
- legal and operational issues.

In this architecture, email is a **medium-strength signal** nothing more.

Conceptual matching evaluation

Identity resolution evaluates a candidate against multiple Golden Records.

Conceptually:

```
{
  "candidate": {
    "first_name": "Giuseppe",
    "last_name": "Verdi",
    "email": "giuseppe.verdi@gmail.com"
  },
  "matches": [
    {
      "golden_id": "GOLD-PER-001",
      "signals": {
        "name_match": 0.6,
        "email_match": 0.9
      },
      "confidence": 0.85
    },
    {
      "golden_id": "GOLD-PER-014",
      "signals": {
        "name_match": 0.4,
        "email_match": 0.0
      },
      "confidence": 0.35
    }
  ]
}
```

This evaluation:

- does not update data,
- does not select attributes,
- does not change truth.

It only answers: **how safe is it to link?**

What confidence really means (and what it does not)

Confidence is often misunderstood.

It is **not**

- a probability of being correct,
- a statistical certainty,
- a guarantee of identity.

Confidence expresses one thing only:

How safe is it for the system to act automatically under current governance rules?

A confidence of 0.85 means:

- the evidence is consistent,
- no strong contradictions exist,
- automatic linking is allowed.

It does not mean “85% chance this is the same person”.

That distinction is essential.

Thresholds are governance, not math

Confidence thresholds are not tuned statistically.

They are defined by governance.

Typical thresholds:

- High confidence: automatic link
- Medium confidence: manual review
- Low confidence: create new Golden Record

These thresholds vary by:

- domain,
- legal constraints,
- operational risk.

They are policy decisions, not algorithmic ones.

First-contact scenario revisited: Web data

Web data is the most common first-contact scenario.

It usually lacks:

- strong identifiers,
- authoritative sources.

If no existing Golden Record reaches the confidence threshold:

- a new Golden Record is created,
- explicitly marked as low-confidence,
- expected to be reconciled later.

This is not a failure case.

It is correct system behavior.

Why identity resolution must precede survivorship

Identity resolution answers **who** we are talking about.

Survivorship answers **which value wins**

Reversing this order causes:

- attributes to be compared across different real-world entities,
- conflicts that are not real,
- irreversible corruption.

Therefore:

No attribute-level decision is made before identity is established.

This rule is absolute.

Manual resolution is not an exception

Some candidates will:

- match multiple Golden Records,
- show conflicting strong signals,
- fall into a grey area.

In these cases, the system must not decide.

Escalation is not a failure path.

It is a **designed outcome**

Manual resolution preserves trust in the system.

Identity resolution does not modify truth

After identity resolution:

- nothing has been merged,
- no attribute has changed,
- no history has been written.

All it does is establish **context**

Only after this context exists can the system safely move forward.

What comes next

Once identity is established, the system can finally ask:

- Which attribute values are authoritative?
- Which conflicts are real?
- Which changes are acceptable?

Those questions belong to the next chapter.

The architectural takeaway

Identity resolution is not about being clever.

It is about being cautious.

If you resolve identity too aggressively, you lose truth.

If you resolve identity too conservatively, you lose usability.

This architecture chooses caution first.

Because once identity is wrong, everything else is wrong forever

Chapter III – Survivorship & Conflict Resolution

Deciding which value wins, and why

Why survivorship exists (and why it is often misunderstood)

After ingestion and identity resolution, the system finally knows **who we are talking about**

Only at this point does a legitimate question emerge:

Which value should represent the truth for this attribute?

Survivorship is not about merging records.

It is about **deciding which attribute value is allowed to speak for the entity**, ~~the~~ explicit governance rules.

Most MDM systems fail here because they:

- apply survivorship too early,
- mix identity and value decisions,
- infer rules instead of declaring them.

In this architecture, survivorship is **deliberate, explicit, and reversible**

Survivorship always comes

after

identity resolution

This rule is absolute.

Before identity resolution:

- attributes belong to candidates,
- conflicts may be artificial,
- values may refer to different real-world entities.

After identity resolution:

- attributes are contextualized,
- conflicts are real,
- decisions are meaningful.

No survivorship rule is evaluated before identity is established.

Violating this rule leads to irreversible corruption.

Survivorship rules are defined, not inferred

Survivorship rules are not guessed by the system.

They are **governance decisions encoded as configuration**

A survivorship rule answers one question only:

Which source is authoritative for this attribute?

Example survivorship configuration for a Person entity:

```
{
  "Person": {
    "cf": { "authoritative_source": "Anagrafe" },
    "first_name": { "authoritative_source": "Anagrafe" },
    "last_name": { "authoritative_source": "Anagrafe" },
    "residence": { "authoritative_source": "Anagrafe" },
    "work_email": { "authoritative_source": "SAP" },
    "email": { "authoritative_source": "Web" },
    "phone": { "authoritative_source": "Web" },
    "mood": { "authoritative_source": "Web" }
  }
}
```

These rules:

- exist before data arrives,
- are stable over time,
- can be audited and explained.

Survivorship is governance expressed as architecture.

Attribute-level governance, not record-level dominance

No system “wins” globally.

Instead:

- **Anagrafe** governs legal identity,
- **SAP** governs operational roles,
- **Web** governs personal preferences and contacts.

Each attribute in the Golden Record exists because:

- its identity is known,
- its source is authorized,
- its inclusion is explainable.

This is why the Golden Record is **composed** not selected.

Conflict types and how they are handled

Once survivorship rules exist, conflicts fall into well-defined categories.

Case 1 – Same attribute, same authoritative source

Example: Web updates a personal email.

```
{
  "source": "Web",
  "payload": {
    "email": "giuseppe.verdii@gmail.com"
  }
}
```

Golden Record already contains:

```
"email": "giuseppe.verdi@gmail.com"
```

Resolution:

- same attribute,
- same governing source,
- recency rule applies.

Result:

- Golden Record updated,
 - previous value archived.
-

Case 2 – Same attribute, different sources, clear authority

Example: SAP sends a name variant.

```
{
  "source": "SAP",
  "payload": {
    "first_name": "G. Verdi"
  }
}
```

Golden Record already has:

```
"first_name": "Giuseppe"
```

Rule:

- first_name governed by Anagrafe.

Resolution:

- SAP value **does not override**
 - value stored as shadow.
-

Case 3 – Competing sources, no deterministic rule

If two sources compete for the same attribute and:

- neither is authoritative,
- or governance rules are missing,

the system must not decide automatically

The conflict is escalated to stewardship.

This is not an exception.

It is correct behavior.

Shadow values: preserving losing truth

Non-winning values are never discarded.

They are preserved as **shadow values** explicitly marked as such.

Example Golden Record with shadow data:

```

{
  "_id": "GOLD-PER-001",
  "entity_type": "Person",
  "view": {
    "first_name": "Giuseppe",
    "last_name": "Verdi",
    "email": "giuseppe.verdi@gmail.com"
  },
  "lineage": {
    "first_name": { "source": "Anagrafe" },
    "email": { "source": "Web" }
  },
  "shadow_values": {
    "first_name": [
      {
        "value": "G. Verdi",
        "source": "SAP",
        "observed_at": "2024-03-10T09:15:00Z"
      }
    ]
  }
}

```

Shadow values provide:

- traceability,
- explainability,
- future reconciliation paths.

Deleting losing values destroys trust.

Authority vs recency vs completeness

Survivorship strategies are **not interchangeable**

Typical strategies:

- **Authority-based** used for legal identity, reference data.
- **Recency-based** used for mutable, user-controlled attributes.
- **Completeness-based** used only when explicitly allowed and low-risk.

These strategies are:

- attribute-specific,
- declared in advance,
- never mixed implicitly.

Manual override as a governed action

Manual intervention is sometimes required.

When it happens, it must be:

- explicit,
- traceable,
- reversible.

Example manual override record:

```
{
  "golden_id": "GOLD-PER-001",
  "attribute": "email",
  "previous_value": "giuseppe.verdi@gmail.com",
  "new_value": "g.verdi@ente.it",
  "reason": "Confirmed by data steward",
  "resolved_by": "steward_02",
  "resolved_at": "2024-03-21T14:32:00Z"
}
```

Manual actions are **data** not side effects.

Survivorship does not erase history

Survivorship decides **what is visible now**

It does not decide:

- what was true before,
- what was observed,
- what was contested.

Those concerns belong to **temporal versioning** covered in the next chapter.

Architectural takeaway

Survivorship is not a merge algorithm.

It is a governance mechanism.

If identity is wrong, survivorship is dangerous.

If governance is implicit, survivorship is arbitrary.

If shadow values are lost, survivorship is dishonest.

Done correctly, survivorship:

- produces explainable truth,
- preserves uncertainty,
- enables trust at scale.

This is the point where Master Data Management stops being integration

and becomes **truth governance**

Chapter IV – Data Quality & Validation

Protecting truth before it is decided

Why Data Quality is structural, not cosmetic

In Master Data Management, data quality is not about “clean data”.

It is about **protecting the truth construction process**

If invalid or semantically inconsistent data is allowed to participate in:

- identity resolution,
- survivorship decisions,
- historical versioning,

then the system will produce results that are formally consistent but **conceptually false**

Once this happens, no downstream rule can repair the damage.

For this reason, data quality in MDM is not a corrective activity.

It is a **preventive architectural gate**

The role of validation in the overall architecture

Validation is not the first step.

It is not the last step.

It sits **exactly between ingestion semantics and identity resolution**

The canonical flow is:

Raw ingestion (staging)
→ Semantic split into entities
→ DATA QUALITY VALIDATION GATE
→ Identity resolution
→ Survivorship & Golden Record composition
→ History management

Anything that does not pass the validation gate:

- does not participate in identity resolution,
- does not influence confidence scores,
- does not affect survivorship,
- does not touch a Golden Record.

This positioning is non-negotiable.

Raw data is not wrong. It is untrusted.

Consider the following incoming event from SAP:

```
{
  "_id": "raw_7781",
  "source": "SAP",
  "ingested_at": "2024-03-12T09:40:00Z",
  "payload": {
    "contract_id": "CTR-2024-ERR",
    "amount": -1200,
    "dec_cf": "VRDGPP80A01H501U",
    "dec_first_name": "Giuseppe",
    "dec_last_name": "Verdi"
  }
}
```

From a technical perspective, this payload is valid.

From a business perspective, it is not.

A contract with a negative amount is **semantically impossible**

The critical point is this:

the data is not “wrong” because it arrived.

It is **untrusted until validated**

Treating untrusted data as truth is how MDM systems corrupt themselves silently.

Validation rules must be declarative and governed

Validation logic must not be:

- hardcoded,
- scattered across services,
- embedded in business logic.

Validation rules are part of governance and must therefore be:

- explicit,
- readable,
- versioned,
- auditable.

A representative validation configuration might look like this:

```
{
  "Contract": {
    "contract_id": {
      "required": true
    },
    "amount": {
      "type": "number",
      "min": 0,
      "error": "Contract amount must be non-negative"
    }
  },
  "Person": {
    "fiscal_code": {
      "required": true,
      "regex": "^[A-Z0-9]{16}$",
      "error": "Invalid fiscal code"
    },
    "email": {
      "required": false,
      "warning": "Email missing"
    }
  }
}
```

These rules do not decide truth.

They decide **admissibility**

They answer one question only:

Is this data allowed to participate in truth construction?

Validation produces a decision, not a boolean

Validation is not a true / false check.

It produces a **structureddecision** that can be reasoned about, persisted, and measured.

Example validation outcome for the SAP contract above:

```
{
  "entity_type": "Contract",
  "valid": false,
  "errors": [
    "Contract amount must be non-negative"
  ],
  "warnings": []
}
```

This output is:

- human-readable,
 - machine-actionable,
 - suitable for governance metrics.
-

Rejected data goes to quarantine, not to oblivion

Invalid data is never discarded.

It is **quarantined**

```
{
  "_id": "quarantine_3321",
  "original_event_id": "raw_7781",
  "entity_type": "Contract",
  "source": "SAP",
  "quarantined_at": "2024-03-12T09:41:02Z",
  "errors": [
    "Contract amount must be non-negative"
  ],
  "payload": {
    "contract_id": "CTR-2024-ERR",
    "amount": -1200
  },
  "status": "OPEN"
}
```

Quarantine serves multiple purposes:

- forensic traceability,
- feedback to source systems,
- manual remediation,
- regulatory defensibility.

An MDM system that silently drops data is as dangerous as one that blindly accepts it.

Warnings are first-class signals, not errors

Not all quality issues should block truth construction.

Example: Web data with missing email.

```
{
  "source": "Web",
  "payload": {
    "first_name": "Giuseppe",
    "last_name": "Verdi",
    "phone": "3331234567"
  }
}
```

Validation result:

```
{
  "entity_type": "Person",
  "valid": true,
  "errors": [],
  "warnings": [
    "Email missing"
  ]
}
```

Golden Record result:

```
{
  "_id": "GOLD-PER-091",
  "entity_type": "Person",
  "view": {
    "first_name": "Giuseppe",
    "last_name": "Verdi",
    "email": null,
    "phone": "3331234567"
  },
  "data_quality_flags": [
    "Email missing"
  ]
}
```

The system:

- does not invent data,
- does not hide incompleteness,
- preserves uncertainty explicitly.

This is intentional.

Why validation must precede identity resolution

If invalid data is allowed into identity resolution:

- confidence scores become distorted,
- false matches increase,
- wrong Golden Records are created.

Once identity is wrong:

- survivorship rules amplify the error,
- history preserves it forever.

Validation is therefore **identity protection** not data hygiene.

This ordering is absolute:

no identity resolution without validation.

Validation is about trust boundaries, not correctness

Validation does not claim that accepted data is correct.

It claims that:

- the data is admissible,
- the data is explainable,
- the data can safely participate in governance.

Truth is decided later.

Validation only decides **who is allowed to speak**

Architectural takeaway

A Golden Record is not protected by merge logic.

It is protected by refusing bad data **before** truth is decided.

If this gate is weak:

- identity resolution becomes unreliable,
- survivorship becomes arbitrary,
- history becomes legally dangerous.

With a strong validation layer:

- matching is stable,
- governance is enforceable,
- truth remains defensible over time.

This is why data quality in MDM is not optional.

It is structural.

Chapter V – Temporal Versioning & History (SCD Type 2)

Truth changes. Memory must not.

Why temporal versioning is not optional in MDM

In Master Data Management, truth is not static.

People change roles.

Emails change.

Residences change.

Responsibilities shift.

If an MDM system only stores the latest version of truth, it is not governing data.

It is overwriting memory.

A production-grade MDM must answer not only:

What is true now?

but also:

What was true at a specific point in time, and why?

This is why **temporal versioning is not an optimization**

It is a foundational capability.

Versioning as a first-class architectural concern

In this architecture, versioning is not implemented as:

- an audit log,
- a change history table,
- a diff-based patch system.

Instead, it follows a **Slowly Changing Dimension Type 2 (SCD Type 2)** applied to Golden Records.

Each meaningful change produces:

- a **new immutable version**
- with a **clearly defined time domain**
- linked to the same logical Golden identity.

Truth evolves.

Memory does not.

The temporal model: `valid_from` and `valid_to`

Each Golden Record version is associated with a time interval:

- `valid_from`: when this version became effective
- `valid_to`: when this version stopped being effective (or null if current)

This time domain defines **exactly when a given truth was valid** enabling unambiguous point-in-time queries.

Conceptually:

```
{
  "golden_id": "GOLD-PER-001",
  "version_id": "v3",
  "valid_from": "2024-03-20T10:15:00Z",
  "valid_to": null,
  "view": {
    "email": "giuseppe.verdi@newmail.com",
    "residence": "Via Milano 22"
  },
  "lineage": {
    "email": { "source": "Web" },
    "residence": { "source": "Anagrafe" }
  }
}
```

This version is immutable.

Any future change creates **v4**, not a mutation of **v3**.

What triggers a new version (and what does not)

A new version is created **only when governed truth changes**

Examples that **do create a new version**

- an authoritative source updates an attribute it governs,
- a survivorship decision changes the winning value,
- a manual stewardship action overrides automation.

Examples that **do not create a new version**

- reprocessing the same data,
- duplicate events,
- validation-only metadata updates.

Versioning is intentional, not noisy.

Query 1 – Retrieving the current truth

Most consuming applications need **only the current truth**

This query retrieves the active version of a Golden Record:

```
{
  "golden_id": "GOLD-PER-001",
  "valid_to": null
}
```

This is:

- deterministic,
- fast,
- free of temporal ambiguity.

Applications do not need to understand history to consume truth correctly.

Query 2 – Point-in-time reconstruction (“as-of” query)

Audits, legal checks, and forensic analysis require reconstructing truth **as it was**

Example: What was considered true on 1st March 2024?

```
{
  "golden_id": "GOLD-PER-001",
  "valid_from": { "$lte": "2024-03-01T00:00:00Z" },
  "valid_to": { "$gt": "2024-03-01T00:00:00Z" }
}
```

This query guarantees:

- a single, unambiguous result,
- no reconstruction logic in application code,
- defensible answers in regulated environments.

This is why time must live **inside the data model** not outside.

Metric 1 – Version creation rate

This metric measures **how often truth changes**

```
{
  "period": "2024-03",
  "entity_type": "Person",
  "versions_created": 4821,
  "distinct_golden_ids": 1200,
  "average_versions_per_entity": 4.0
}
```

Interpretation:

- High rate → volatile domain or weak upstream data
- Very low rate → overly rigid governance
- Sudden spikes → upstream incidents or rule changes

This metric tells you **how stable your truth really is**

Metric 2 – Source impact on truth evolution

Not all sources contribute equally to change.

This metric shows **which sources actually modify governed truth**

```
{
  "period": "2024-03",
  "entity_type": "Person",
  "source_impact": {
    "Anagrafe": 68,
    "Web": 21,
    "SAP": 11
  }
}
```

Interpretation:

- Authoritative sources should dominate
- Web should enrich, not destabilize
- Unexpected dominance signals governance drift

This metric closes the loop between **policy and reality**

Why versioning must be immutable

Immutability guarantees that:

- past decisions cannot be altered,
- audits are reproducible,
- trust is cumulative, not fragile.

Once a version is written:

- it is never edited,
- it is never deleted,
- it can only be superseded.

This is the difference between **history** and **logs**

Application-level implications

Applications consuming MDM data benefit directly:

- transactional systems always read the current version,
- analytical systems can reconstruct historical states,
- legal and compliance teams can replay decisions.

No application needs to implement custom history logic.

The architecture absorbs complexity centrally.

Architectural takeaway

Temporal versioning is not about storage.

It is about responsibility.

If your system cannot explain:

- what was true,
- when it was true,
- and why it changed,

then it is not governing truth.

It is merely storing data.

Part III – Final Considerations

From data integration to truth governance

What this architecture really delivers is not a data pipeline.

It is a **truth governance system**

One that:

- models uncertainty,
- resolves conflict explicitly,
- preserves memory,
- exposes responsibility.

Golden Records are a consequence, not the goal. Golden Records emerge when:

- data is validated,
- identity is established,
- survivorship is governed,
- history is preserved.

Without these, a Golden Record is just a snapshot.

Why most MDM initiatives fail

They:

- merge before validating,
- automate before measuring,
- overwrite before remembering,
- hide uncertainty instead of modeling it.

They optimize for simplicity where rigor is required.

Why this pattern scales

Because it:

- separates concerns cleanly,
- makes every decision explainable,
- keeps performance and governance compatible,
- treats time as a first-class dimension.

The one sentence that matters. You are no longer managing data. You are managing truth.

And truth:

- is constructed,
- is governed,
- changes over time,
- must never forget where it came from.

If your system can do that, then you are finally doing Master Data Management.