

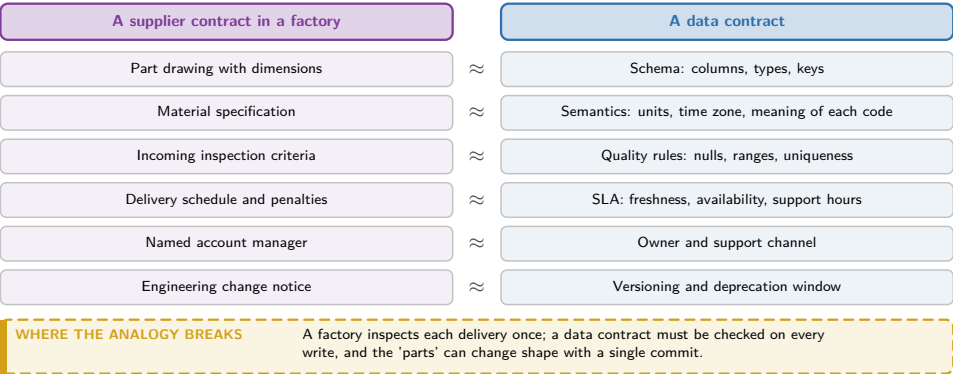


Data Contracts

Data engineering for AI | What a data contract holds, the Open Data Contract Standard (ODCS v3, as of 2026) annotated, dbt model contracts, breaking vs non-breaking changes, versioning and deprecation windows, enforcement in CI, at write and at read, ownership

#DataContracts #ODCS #dbt #SchemaEvolution #DataOwnership

What a contract is



A data contract is an API specification for a table

Block	Holds	Failure without it
Schema	columns, logical and physical types, keys, nullability	a renamed column turns a feature into nulls
Semantics	units, currency, time zone, enum meanings, grain (one row per what)	cents read as euros; UTC read as local time
Quality	rules with thresholds (null rate, uniqueness, ranges, row counts)	bad batches flow into training
SLA	freshness, update frequency, availability, retention	models train on stale or partial days
Ownership	producer team, contact, support hours, escalation	nobody fixes a broken feed
Usage and limits	purpose, allowed uses, PII classification, licence	personal data reused for a purpose never agreed

The six blocks of a contract and what goes wrong without each

A contract in ODCS and in dbt

contracts/orders.odcs.yaml

```
1  apiVersion: v3.0.2
2  kind: DataContract
3  id: commerce-orders
4  name: orders
5  version: 2.1.0
6  status: active
7  domain: commerce
8  description:
9    purpose: Orders for churn and demand models
10   limitations: No card data. Amounts in EUR.
11  schema:
12    - name: orders
13      physicalName: silver.orders
14      properties:
15        - name: order_id
16          logicalType: string
17          primaryKey: true
18          required: true
19        - name: customer_id
20          logicalType: string
21          required: true
22          classification: restricted
23        - name: amount_eur
24          logicalType: number
25          quality:
26            - type: sql
27              query: SELECT COUNT(*) FROM orders
28              WHERE amount_eur < 0
29              mustBe: 0
29          quality:
30            - type: sql
31              query: SELECT COUNT(*) FROM orders
32              mustBeGreaterThan: 100000
33      slaProperties:
34        - property: latency
35          value: 4
36          unit: h
37  team:
38    - username: orders-team
39      role: owner
```

1 semantic version: major on breaking changes

2 purpose and limits: what ML may and may not do with it

3 grain and key: one row per order

4 classification drives PII handling (sheet 05)

5 column rule: no negative amounts

6 table rule: volume floor catches partial loads

7 SLA: data at most 4 h old

8 owner to page when a rule fails

ODCS v3-style contract for an orders table (abridged; check field names against the spec version you validate with)

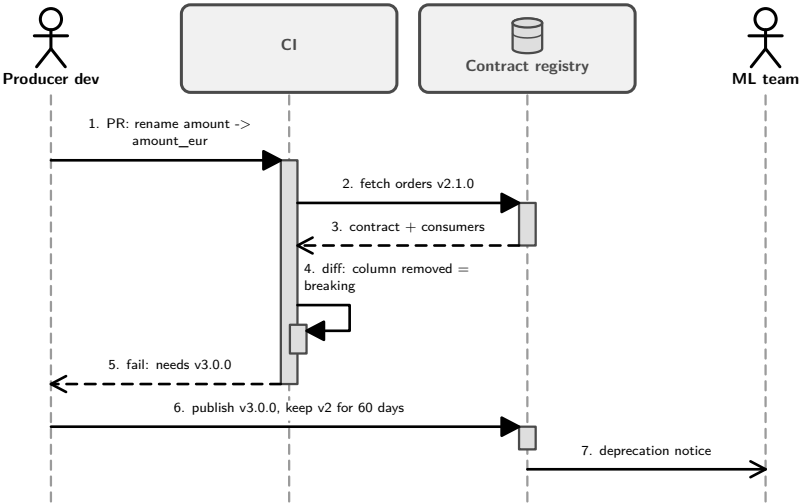
```
The same promise enforced at build time by dbt (1.8+ syntax: data_tests, model contracts)
models:
  - name: orders
    config:
      contract: {enforced: true} # build fails on type drift
    columns:
      - name: order_id
        data_type: varchar
        constraints: [{type: not_null}]
        data_tests: [unique]
      - name: amount_eur
        data_type: numeric
        data_tests:
          - dbt_utils.accepted_range:
              min_value: 0
sources:
  - name: shop
    loaded_at_field: _ingested_at
    freshness:
      warn_after: {count: 2, period: hour}
      error_after: {count: 4, period: hour}
    tables: [{name: orders_raw}]
```

Command	What it does
datacontract lint contracts/orders.odcs.yaml	Validate the contract file against the specification
datacontract test contracts/orders.odcs.yaml	Run schema and quality checks against the server the contract names
datacontract export --format dbt contracts/orders.odcs.yaml	Generate dbt model YAML from the contract (one source of truth)
dbt build --select orders	Build the model, enforce the model contract, run its data tests
dbt source freshness	Compare max(loaded_at_field) with warn/error thresholds

Changes, versions and enforcement

Conditions	Remove	Narrow	Semantics	Add	Docs
Column removed or renamed	Y	N	N	N	N
Type narrowed or nullability tightened	--	Y	N	N	N
Meaning, unit or grain changed	--	--	Y	N	N
Optional column added or type widened	--	--	--	Y	N
Docs or descriptions only	--	--	--	--	Y
Actions					
MAJOR bump, new version side by side	X	X	X		
Deprecation window, notify consumers	X	X	X		
MINOR bump				X	
PATCH bump					X
Notes	consumers fail loudly	values may truncate or fail	silent: same schema, wrong numbers	safe for readers	no data impact

Is this change breaking? What to do



Shift left: a breaking change is caught in the producer's pull request, not in the model's metrics

Point	Mechanism	Catches
In CI (producer PR)	contract lint + compare with the published version; dbt contract on a slim CI build	breaking schema changes before merge
At write	schema registry compatibility (BACKWARD, FULL) for events; table schema enforcement in Delta/Iceberg; quality gate before publishing a partition	bad batches, type drift
At read	consumer-side checks before training (row count, null rate, ranges)	anything the producer missed; defence in depth

Where to enforce: three points, use all of them

Task	Producer team	Data platform	ML consumer	Governance
Define schema and semantics	R A		C	
Agree quality rules and SLA	A	C	R	
Run checks and SLA monitors	I	R A		
Fix a violation at the source	R A		I	
Approve a breaking change	A		C	C
Classify PII and allowed uses	R			A

R

 Responsible

A

 Accountable

C

 Consulted

I

 Informed

Who does what for a contracted dataset (one Accountable per row)

© Worked check and pitfalls

PROBLEM

The orders contract says: null rate of customer_id ≤ 0.1 %, no duplicate order_id, row count within the 30-day mean ± 3 standard deviations, data no older than 4 h at 07:00.

GIVEN

- Rows today n = 2,400,000; null customer_id = 3,100; duplicate order_id = 0
- 30-day row count mean μ = 2,300,000, standard deviation σ = 60,000
- Newest event time in the batch: 02:30; check time 07:00

1

Null rate

3,100 / 2,400,000 = 0.129 % > 0.1 %

Why: rule fails: about 700 nulls above the 2,400 allowed

2

Uniqueness

duplicates = 0

Why: rule passes

3

Volume band

μ ± 3σ = 2,120,000 to 2,480,000; 2,400,000 inside

Why: rule passes

4

Freshness

07:00 - 02:30 = 4 h 30 min > 4 h

Why: SLA breached by 30 min

5

Decision

quarantine the partition, page the owner, keep yesterday's training snapshot

Why: two of four rules failed

ANSWER

Batch rejected: null rate 0.129 % (limit 0.1 %) and freshness 4.5 h (limit 4 h)

CHECK

✓

Allowed nulls = 0.001 × 2,400,000 = 2,400; observed 3,100

Does today's batch honour the contract?

MYTH

✗

A schema is a contract

✗

Contracts slow producers down

✗

Every table needs a contract

✗

Consumers can always adapt to changes

FACT

✓

A schema is one of six blocks

Same columns with a unit changed from cents to euros pass every schema check and break every model.

✓

They move the cost to where it is cheapest

A failed PR costs minutes; a corrupted training set found weeks later costs a retrain and an incident.

✓

Contract the tables others depend on

Start with sources feeding models and executive metrics; internal staging tables can stay uncontracted.

✓

Breaking changes need a side-by-side version

Models are retrained on a schedule; give a deprecation window, typically weeks.

Beliefs about data contracts

© Tadata | tadata.fr | contact@tadata.codes

✗ Writing the contract after the incident

Teams often write contracts as documentation nobody runs. A contract that is not executed in CI and at write time is a wiki page; generate dbt tests or checks from it and fail the pipeline on violation.

🔗 Interview: 'How would you stop upstream changes from breaking your models?'

Agree a contract with each upstream owner (schema, semantics, quality, SLA, owner), store it in Git, check it in the producer's CI, enforce schema at write, run consumer-side checks before training, and require a major version with a deprecation window for breaking changes.

✓ Recap

Six blocks

schema, semantics, quality, SLA, owner, usage

Semantic changes break

even with an identical schema

Major = breaking

side by side + deprecation window

Enforce three times

CI, at write, at read

One source of truth

generate dbt tests from the contract

Notes to keep

Q1 Six blocks of a data contract?

A Schema, semantics, quality rules, SLA, ownership, usage and limits.

Q2 Which change is breaking although the schema is unchanged?

A A change of meaning, unit or grain (cents to euros, local time to UTC, one row per order to per line).

Q3 What does dbt's contract enforced: true do?

A Fails the build when the model's columns or data types differ from those declared in YAML.

Q4 Version bump for adding an optional column?

A MINOR: existing consumers are unaffected.

Q5 What is a deprecation window?

A A period when old and new versions are both served so consumers migrate before the old one is removed.

Q6 Name the three enforcement points.

A Producer CI, at write (schema registry or table enforcement, quality gate), at read (consumer checks).

Q7 Which standard defines data contracts in YAML, as of 2026?

A The Open Data Contract Standard (ODCS, Bitol project, Linux Foundation), v3.

Q8 What does dbt source freshness compare?

A The newest loaded_at_field value with the warn_after and error_after thresholds.

Q9 Who is accountable for fixing a contract violation?

A The producer team: the fix belongs at the source, not in each consumer.