DATA ENGINEERING · GOVERNANCE

WENN DATEN IHRE BEDEUTUNG ÄNDERN

Die Änderung eines Datenflusses zwischen Anwendungen vorbereiten

Veröffentlicht am 10. November 2026 · Version 1.3.0 · Aktualisiert am 10. November 2026
Online-Artikel:
https://datanalysis.re/ressources/data-engineering/daten-bedeutung-aenderungen-zwischen-anwendungen

Ein Update kann korrekt ausgerollt werden, die Verarbeitungen können pünktlich enden, und die Beträge, mit denen die Finanzabteilung arbeitet, können trotzdem falsch werden. Bevor Sie gemeinsam genutzte Daten ändern, müssen Sie daher prüfen, wie die empfangenden Anwendungen sie verstehen werden. Hier ist eine Methode, um diese Prüfung zu organisieren, mit einem Arbeitsblatt für einen konkreten Datenfluss.

Fiktives Beispiel: gleiches Feld, gleicher Typ, andere Einheit

Vorher · invoice_total	Nachher · invoice_total
48.50 Euro	**4850** Cent

Als Euro gelesen: 4.850 €, also hundertmal zu hoch

Normalisiert: 4850 ÷ 100 = 48,50 €

Eine gültige Zahl kann falsch interpretiert werden

Nehmen wir ein fiktives Beispiel. Eine Fakturierungsanwendung übermittelt den Bruttobetrag einer Rechnung im Feld `invoice_total`. Für eine Rechnung über 48,50 Euro sendet sie den numerischen Wert `48.50`. Nach einer Änderung sendet sie `4850`, nun in Cent ausgedrückt. Der Feldname bleibt gleich, und das Schema erlaubt weiterhin eine Zahl.

Eine Prüfung, die nur das Vorhandensein des Feldes und seinen Typ kontrolliert, kann beide Werte akzeptieren. Liest die nachfolgende Verarbeitung jedoch weiterhin Euro, erhält sie 4.850 Euro. In diesem Beispiel ist der interpretierte Betrag hundertmal zu hoch. Die Währung EUR reicht nicht aus, um die Mehrdeutigkeit aufzulösen: Man muss auch die Darstellungseinheit kennen.

Ein Abgleich mit der Quellrechnung oder eine gut gewählte fachliche Regel kann diese Abweichung erkennen. Das Problem ist der Umfang der tatsächlich ausgeführten Prüfung. Ein technischer Status „abgeschlossen“ sagt uns, dass die Verarbeitung gelaufen ist; ob der Betrag stimmt, hängt auch von den vorgesehenen Prüfungen ab.

Die dbt-Dokumentation beschreibt diese Schwierigkeit: Eine Berechnung kann die Ergebnisse einer Spalte ändern, während Name und Typ gleich bleiben. Das Team, das das Modell pflegt, muss die Folgen für seine Nutzer beurteilen. [1]

Verwendungen finden, bevor das Feld geändert wird

Um die Änderung vorzubereiten, schlagen wir vor, von einem Datenelement auszugehen und seinem Weg zu folgen. Wer erzeugt es? Welche Verarbeitungen transformieren es? Welche Anwendungen, Exporte oder Entscheidungen hängen davon ab? Eine Abhängigkeitsansicht ist ein guter Ausgangspunkt, ihre Abdeckung muss aber überprüft werden. Eine exportierte und anschließend manuell weiterverarbeitete Datei kann der technischen Kartierung entgehen.

Stellen wir uns in unserem Beispiel zwei Konsumenten vor: ein Finanz-Monitoring-Dashboard und einen Export für das Forderungsmanagement. Das erste zeigt die Summen an; der zweite dient dazu, eine Liste zu bearbeitender Rechnungen vorzubereiten. Die Änderung muss in beiden Verwendungen getestet werden. Nur das Eintreffen der Daten im Data Warehouse zu validieren, würde die endgültige Lesart ungeprüft lassen.

Halten Sie für jeden Konsumenten eine verantwortliche Person, die anzupassende Verarbeitung und einen erwarteten Nachweis fest. Die verantwortliche Person der Quellanwendung bestätigt die neue Darstellung. Das Integrationsteam prüft die Transformation. Die verantwortliche Person jeder Verwendung bestätigt das erzielte Ergebnis. Eine Person steuert den Übergang und kennt die noch offenen Punkte.

Diese Liste sollte dem Datenfluss angemessen bleiben. Für selten genutzte Daten können wenige Zeilen genügen. Bei einem Betrag, der an mehrere Systeme verteilt wird, sollte eine Abhängigkeit ohne verantwortliche Person geklärt werden, bevor die Umstellung terminiert wird.

Definition und Erwartungen explizit machen

Ein Datenvertrag ist eine Vereinbarung zwischen denen, die Daten erzeugen, und denen, die sie nutzen. Confluent unterscheidet in dieser Vereinbarung Struktur, Einschränkungen, Metadaten, Regeln und Weiterentwicklung. [2] Für unsere Prüfung behalten wir die Informationen, mit denen sich entscheiden lässt, ob die Änderung akzeptabel ist.

Zu klärende Punkte, angewendet auf das fiktive Beispiel

Zu klärender Punkt	Anwendung auf unser fiktives Beispiel
Fachliche Definition	Bruttobetrag der Rechnung, eine Rechnung pro Datensatz.
Einheit und Währung	Euro vor der Änderung; Euro-Cent danach.
Sonderfälle	Negative Gutschriften, Nullbetrag, Stornierung und fehlender Wert, nach expliziten Regeln zu behandeln.
Versionen und Zeitraum	Empfangene Version, Wirksamkeitsdatum und Behandlung historischer Rechnungen.
Konsumenten	Finanz-Monitoring und Export für das Forderungsmanagement, jeweils mit einer verantwortlichen Person.
Prüfungen und Reaktion	Abgleich mit den Quellrechnungen; Stopp oder Ausleitung des Datenflusses bei ungeklärter Abweichung.

Dieser Inhalt kann in einem Referenzverzeichnis, einem Code-Repository oder einer versionierten Spezifikation geführt werden. Der Ablageort muss es den betroffenen Teams ermöglichen, ihn zu finden und die Änderung nachzuvollziehen. Das Dokument muss sich anschließend in ausgeführte Prüfungen und zugewiesene Freigaben übersetzen. Eine schriftliche Regel, die nie an einer Abnahme oder Überwachung teilnimmt, erkennt keine Abweichung.

Der Open Data Contract Standard unterscheidet zudem textlich beschriebene Regeln von ausführbaren Regeln, zum Beispiel in SQL oder mit einer Datenqualitäts-Engine. [3] Diese Unterscheidung hilft zu erkennen, was bereits geprüft wird und was noch umzusetzen ist.

Das Ergebnis in einer gemeinsamen Einheit testen

Ein nützlicher Test vergleicht Werte mit derselben Bedeutung. Beträgt der neue Wert `4850` Cent, ist sein normalisierter Betrag `48.50` Euro. Dieser Betrag ist mit der Referenz in Euro zu vergleichen. Ein direkter Vergleich der beiden Rohzahlen würde eine erwartete Warnung erzeugen, ohne die Umrechnung zu prüfen.

Wählen Sie einen kleinen Satz von Rechnungen mit bekannten erwarteten Ergebnissen. Nehmen Sie die fachlich zulässigen Fälle auf: eine normale Rechnung, eine Gutschrift, einen Nullbetrag und die relevanten Situationen von Stornierung oder fehlendem Wert. Halten Sie für jeden Fall die Eingabedaten, die verwendete Version, das erwartete Ergebnis und das in jedem Konsumenten beobachtete Ergebnis fest.

Prüfungen und erwartete Ergebnisse im fiktiven Beispiel

Prüfung	Erwartetes Ergebnis in unserem Beispiel
Rechnung über 48,50 Euro	`4850` Cent werden nach der Normalisierung zu `48.50` Euro.
Gutschrift über 12 Euro	Sind Gutschriften negativ, wird `-1200` zu `-12.00` Euro.
Fehlende Einheit oder unbekannte Version	Die Verarbeitung wendet die vereinbarte Regel an und meldet den Fall; sie errät die Einheit nicht.
Gleicher Rechnungsstapel	Die normalisierte Summe entspricht der Referenz, mit einer dokumentierten Rundungsregel.
Erneutes Einspielen eines Stapels	Das Ergebnis enthält kein zusätzliches Duplikat.

Die Summe ist nützlich, kann aber Fehler verdecken, die sich gegenseitig ausgleichen. Prüfen Sie auch die Zeilen, ihre Anzahl und ihre Identität. Eine Toleranz braucht einen fachlichen Grund, zum Beispiel eine definierte Rundung; ein Schwellenwert, der nur gewählt wird, um die Abnahme zu bestehen, macht die Prüfung wenig aussagekräftig.

Legen Sie bei Beträgen außerdem fest, wo gerundet wird und wie viele Nachkommastellen erhalten bleiben. Dieselbe Berechnung auf Zeilenebene oder nach der Aggregation kann unterschiedliche Ergebnisse liefern. Diese Entscheidung muss von den betroffenen Konsumenten geteilt werden.

Den Übergang nach den Abhängigkeiten wählen

Wenn alle Konsumenten gemeinsam angepasst werden können, kann eine koordinierte Umstellung passen. Dann müssen die Reihenfolge der Eingriffe, die Tests nach dem Deployment und die Entscheidung festgelegt werden, falls einer davon fehlschlägt. Ein kurzes Zeitfenster entbindet nicht davon, zu wissen, wer die Verarbeitung stoppen kann.

Haben die Teams unterschiedliche Zeitpläne, kann eine Phase der Koexistenz nötig sein. In unserem Beispiel könnte die Quelle eine neue Version oder ein neues Feld mit expliziter Einheit bereitstellen. Die Konsumenten migrieren dann nach einem bekannten Zeitplan. Es gilt zu vermeiden, dass dasselbe Datenelement nacheinander „Euro“ und dann „Cent“ bedeutet, ohne die beiden Darstellungen zuverlässig unterscheiden zu können.

dbt erlaubt für versionierte Modelle, parallele Versionen beizubehalten und ein Abkündigungsdatum festzulegen. Die Dokumentation weist auch auf die Kosten dieser Koexistenz hin. [1] Das Prinzip ist über dieses Werkzeug hinaus nützlich: Jede beibehaltene Version braucht identifizierte Nutzer und eine Bedingung für ihre Entfernung. Kopien ohne Zeitplan zu vervielfachen, verschiebt das Problem nur.

Der Übergang muss auch die Historie behandeln. Bleiben bereits erfasste Rechnungen in der alten Darstellung? Werden sie umgerechnet? Wie erkennt man einen alten Stapel, der nach der Umstellung erneut eingespielt wird? Ein Verarbeitungsdatum allein kann unzureichend sein, wenn der Stapel ältere Daten enthält. Eine explizite Version oder eine andere eindeutige Regel muss die Interpretation leiten.

Die Wiederherstellung vor der Freigabe vorbereiten

Das Zurückrollen muss den Code und die bereits verarbeiteten Daten abdecken. Eine alte Verarbeitung wieder in Betrieb zu nehmen, korrigiert nicht automatisch die während des Vorfalls verteilten Beträge. Legen Sie den betroffenen Stapel fest, die auszusetzenden Ausgaben und die Art, wie die Ergebnisse ohne Duplikate neu berechnet oder korrigiert werden.

Vor der Umstellung muss das Team eine konkrete Frage beantworten können: Wenn die Prüfung einer Rechnung fehlschlägt, welche Maßnahme stoppt die Ausbreitung und wer löst sie aus? Je nach Datenfluss kann das ein Zurückhalten, eine Datenquarantäne oder ein Stopp des Konsumenten sein. Die Antwort hängt von den Kosten falscher Daten und den Kosten einer Unterbrechung ab.

Legen Sie auch den Kontrollpunkt nach dem Deployment fest. Die verantwortlichen Personen der Verwendungen müssen ihre Ergebnisse auf demselben Datenumfang bestätigen. Der Abschluss der Änderung kann sich dann auf identifizierte Nachweise stützen, und die Entfernung der alten Version darauf, dass keine Konsumenten mehr von ihr abhängen.

Ihre nächste Änderungsprüfung vorbereiten

Das zugehörige Arbeitsblatt greift diese Vorgehensweise auf: ein ausgefülltes fiktives Beispiel, dann ein Raster, um Ihren Datenfluss zu dokumentieren. Es ersetzt keine Testergebnisse: Es zeigt, was geprüft werden muss, von wem, und wo der Nachweis zu finden ist.

Beginnen Sie mit einem Datenelement, dessen Änderung bereits geplant ist. Bringen Sie das Quellteam und die verantwortlichen Personen der Verwendungen zusammen. Wenn Sie einen Konsumenten, seine erwartete Einheit oder seine verantwortliche Person noch nicht benennen können, haben Sie einen Punkt gefunden, der vor der Auslieferung zu klären ist.

Sie ändern Daten, die von mehreren Anwendungen genutzt werden? Sprechen wir über einen konkreten Datenfluss, um seine Abhängigkeiten, die vorzusehenden Prüfungen und den Übergang abzustecken.

→ Datenfluss mit DATANALYSIS besprechen: https://datanalysis.re/contact?lang=de

→ Ausfüllbares Arbeitsblatt (PDF): https://datanalysis.re/downloads/data-n2-fiche-remplissable.de.pdf

→ HTML-Arbeitsblatt: https://datanalysis.re/ressources/data-engineering/daten-bedeutung-aenderungen-zwischen-anwendungen/arbeitsblatt

Verwandte Expertise

Data Engineering (auf Französisch): https://datanalysis.re/expertises/data-engineering?lang=fr

Data Governance (auf Französisch): https://datanalysis.re/expertises/data-governance?lang=fr

Quellen

Quellen am 10. November 2026 erneut geprüft. Methode und Beispiele dienen der Veranschaulichung; es werden keine Kundenergebnisse dargestellt.

[1] dbt Labs | Model versions (Berechnungsänderung ohne Änderung von Name oder Typ; Versionen und Abkündigung)
https://docs.getdbt.com/docs/mesh/govern/model-versions

[2] Confluent | Data Contracts for Schema Registry (Struktur, Regeln, Metadaten, Weiterentwicklung)
https://docs.confluent.io/cloud/current/sr/fundamentals/data-contracts.html

[3] Bitol | Open Data Contract Standard, Data Quality, version 3.2.0 (textliche Beschreibung gegenüber ausführbarer Prüfung)
https://bitol-io.github.io/open-data-contract-standard/v3.2.0/data-quality/

Fiktives Lehrbeispiel; keine Kundenergebnisse.