DATA ENGINEERING · BETRIEB

EINEN DATENFLUSS WIEDER AUFNEHMEN, OHNE SEINE WIRKUNGEN ZU VERVIELFACHEN

Veröffentlicht am 10. November 2026 · Version 1.0.0 · Aktualisiert am 10. November 2026
Online-Artikel:
https://datanalysis.re/ressources/data-engineering/datenfluss-wiederaufnehmen-ohne-doppelte-wirkungen

Eine Verarbeitung bricht ab. Die Datenbank hat den Bestellstapel gespeichert, aber die Bestätigung ist nie bei der Verarbeitung angekommen. Soll der Schreibvorgang beim Neustart wiederholt werden? Ohne geeigneten Mechanismus kann dieser erneute Versuch jede Bestellung ein zweites Mal anlegen.

Das Problem besteht nicht nur darin, die Verarbeitung wieder zu starten. Es gilt festzustellen, was tatsächlich ausgeführt wurde, was unsicher bleibt und was wiederholt werden kann, ohne das erwartete Ergebnis zu verändern. Das hier dargestellte Szenario ist fiktiv; die vorgeschlagenen Prüfungen müssen an die Architektur des Datenflusses angepasst werden.

Das Scheitern der Antwort vom Scheitern der Operation unterscheiden

Wenn eine Anfrage in ein Timeout läuft, weiß der Client, dass er keine verwertbare Antwort erhalten hat. Er weiß nicht unbedingt, ob das Zielsystem die Operation ausgeführt hat. Systematisch neu zu versuchen, ist daher nicht immer folgenlos.

Die Engineering-Empfehlungen von AWS beschreiben diese Mehrdeutigkeit und den Nutzen idempotenter APIs: Derselbe Aufruf kann erneut versucht werden, ohne die beabsichtigten Wirkungen zu vervielfachen. Sie benennen auch die Grenzen, die mit der Identität der Anfrage, ihrer Gültigkeitsdauer und Änderungen der Absicht zusammenhängen. [1]

Für unseren Bestellstapel muss erkennbar sein, dass ein neuer Versuch derselben Operation entspricht und nicht einer neuen, legitimen Bestellung.

Der fachlichen Absicht eine Identität geben

Ein stabiler Schlüssel kann je nach gewähltem Umfang den Stapel, die Bestellung oder die auszuführende Operation identifizieren. Er muss über alle Versuche derselben Anfrage hinweg gleich bleiben. Umgekehrt muss eine neue fachliche Anfrage eine andere Identität erhalten können, auch wenn manche Werte gleich sind.

Nur den Inhalt einer Anfrage zu vergleichen, reicht nicht immer: Zwei tatsächlich unterschiedliche Bestellungen können dieselben Artikel und dieselben Mengen enthalten. Die Wahl des Schlüssels sollte daher mit der fachlich verantwortlichen Person und der verantwortlichen Person der Quellanwendung besprochen werden.

Außerdem ist festzulegen, was geschieht, wenn ein bereits bekannter Schlüssel mit einem anderen Betrag oder Empfänger zurückkommt. Die Verarbeitung darf eine Korrektur, einen Aufruffehler und eine identische Wiederholung nicht verwechseln. Die Idempotenz-Dokumentation einer API, zum Beispiel von Stripe, ist für ihre genauen Regeln heranzuziehen, statt ein universelles Verhalten anzunehmen. [2]

Den Schreibvorgang schützen, nicht nur vorher prüfen

Eine Prüfung „Gibt es diese Bestellung schon?“, gefolgt von einem Einfügen, kann ein Duplikat durchlassen, wenn zwei Versuche gleichzeitig laufen. Der Schutz muss den Moment abdecken, in dem die Wirkung gespeichert wird.

Je nach System kann dies auf einer Eindeutigkeitsbedingung, einer Transaktion oder einem Idempotenz-Mechanismus der Ziel-API beruhen. Die Wahl hängt von der Datenbank-Engine, dem Datenmodell und den zu schützenden Wirkungen ab. Eine Prüfung auf Anwendungsebene ersetzt nicht automatisch eine atomare Schreibgarantie. Eindeutigkeitsbedingungen gehören zu den dokumentierten Mechanismen von PostgreSQL; ihr Umfang muss der gewählten fachlichen Identität entsprechen. [3]

Auch die Antwort beim Wiederanlauf muss festgelegt werden. War die Anfrage bereits erfolgreich, sollte der Client eine verwertbare Referenz abrufen können, statt in einem unbestimmten Zustand zu bleiben.

Jede externe Wirkung prüfen

Dass die Bestelltabelle kein Duplikat enthält, garantiert nicht, dass es keine doppelte Benachrichtigung, Rechnung oder Schreibung in einem anderen System gibt. Jede Wirkung muss identifiziert und in ihrem eigenen Umfang getestet werden.

Eine Architektur, die mehrere Systeme koordiniert, muss unter Umständen Zwischenzustände, verzögerte Versuche und Korrekturen behandeln. Ein lokaler Schutz darf nicht als globale Garantie einer Verarbeitung „genau einmal“ dargestellt werden. Die Methode für den Wiederanlauf muss ausdrücklich sagen, was sie garantiert und welche Fälle von einer Person zu prüfen bleiben.

Die Unsicherheit vor dem ersten Vorfall testen

Eine nützliche Abnahme beschränkt sich nicht auf den Normalablauf. Die zugehörige Checkliste schlägt vor, die folgenden Situationen mit fiktiven Daten und im Voraus festgelegten erwarteten Ergebnissen zu testen.

Testsituationen und erwartete Prüfungen

Testsituation	Was zu prüfen ist
Bestätigung nach dem Schreiben verloren	Die Wiederholung derselben Anfrage erzeugt keine zusätzliche Wirkung; das ursprüngliche Ergebnis ist abrufbar.
Zwei nebenläufige Versuche	Der Schutz hält der Nebenläufigkeit stand, nicht nur zwei aufeinanderfolgenden Aufrufen.
Gleicher Schlüssel, geänderter Inhalt	Das vereinbarte Verhalten ist explizit: Ablehnung, Konflikt oder Korrekturprozess.
Späte Wiederholung	Die Grenzen der Schlüsselaufbewahrung und der Wiederaufnahme der Historie sind bekannt.
Eine Wirkung gelingt, eine andere scheitert	Der Teilzustand ist sichtbar, und der Wiederanlauf führt bereits ausgeführte Wirkungen nicht blind erneut aus.
Unsicheres Ergebnis im Zielsystem	Vor einem neuen Versuch sind eine verantwortliche Person und eine Abgleichmethode benannt.

Ein bestandener Test belegt das unter seinen Bedingungen beobachtete Verhalten. Er entbindet nicht davon, Abweichungen zu überwachen, Versionen zu dokumentieren und den Umfang neu zu prüfen, wenn sich der Datenfluss ändert.

Den Wiederanlauf als Teil des Dienstes vorbereiten

Für einen konkreten Datenfluss muss das Team die fachliche Anfrage benennen können, ihren Schlüssel, das System, das für ihren Zustand maßgeblich ist, wie lange eine Wiederholung unterstützt wird und wer für Ausnahmen verantwortlich ist. Es muss auch wissen, welche Ausgaben auszusetzen sind, wenn das Ergebnis unsicher ist.

Bei DATANALYSIS besteht die Integrationsarbeit darin, diese Regeln mit den Schreibvorgängen, den Prüfungen und den davon abhängigen Verwendungen zu verbinden. Ziel ist ein überprüfbarer, an den Datenfluss angepasster Wiederanlauf statt einer Schaltfläche „erneut starten“, deren Folgen unbekannt bleiben.

Stecken wir den Wiederanlauf eines konkreten Datenflusses ab. Bringen Sie seine Quelle, seine Ziele und ein Unterbrechungsszenario mit. Wir können die zu schützenden Wirkungen, die nützlichen Tests und die Verantwortlichkeiten für den Wiederanlauf festlegen.

→ Wiederanlauf meines Datenflusses abstecken: https://datanalysis.re/contact?lang=de

→ Ausfüllbare Checkliste (PDF): https://datanalysis.re/downloads/data-reprise-checklist.de.pdf

→ HTML-Checkliste: https://datanalysis.re/ressources/data-engineering/datenfluss-wiederaufnehmen-ohne-doppelte-wirkungen/checkliste

Im selben Dossier

Wenn Daten ihre Bedeutung ändern: Änderungen zwischen Anwendungen vorbereiten:
https://datanalysis.re/ressources/data-engineering/daten-bedeutung-aenderungen-zwischen-anwendungen

Eine Datenplattform modernisieren, während das Unternehmen weiterarbeitet:
https://datanalysis.re/ressources/data-engineering/datenplattform-modernisieren-laufender-betrieb

Verwandte Expertise

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

Quellen und Umfang

Primärquellen am 10. November 2026 eingesehen. Beispiel und Checkliste sind unsere eigene didaktische Anwendung: weder ein Kundenfall noch eine vom gewählten System unabhängige Garantie. Es wird hier kein Stripe- oder PostgreSQL-Konnektor beansprucht.

[1] AWS Builders' Library | Making retries safe with idempotent APIs (Mehrdeutigkeit einer abgelaufenen Anfrage, idempotente APIs, Grenzen bei Identität der Anfrage, Gültigkeitsdauer und Änderungen der Absicht)
https://aws.amazon.com/builders-library/making-retries-safe-with-idempotent-APIs/

[2] Stripe | Idempotent requests (Idempotenz-Regeln dieser API, nachzulesen statt ein universelles Verhalten anzunehmen)
https://docs.stripe.com/api/idempotent_requests

[3] PostgreSQL 18 | Constraints (Eindeutigkeitsbedingungen, deren Umfang der gewählten fachlichen Identität entsprechen muss)
https://www.postgresql.org/docs/18/ddl-constraints.html

Fiktives Lehrszenario; keine Kundenergebnisse.