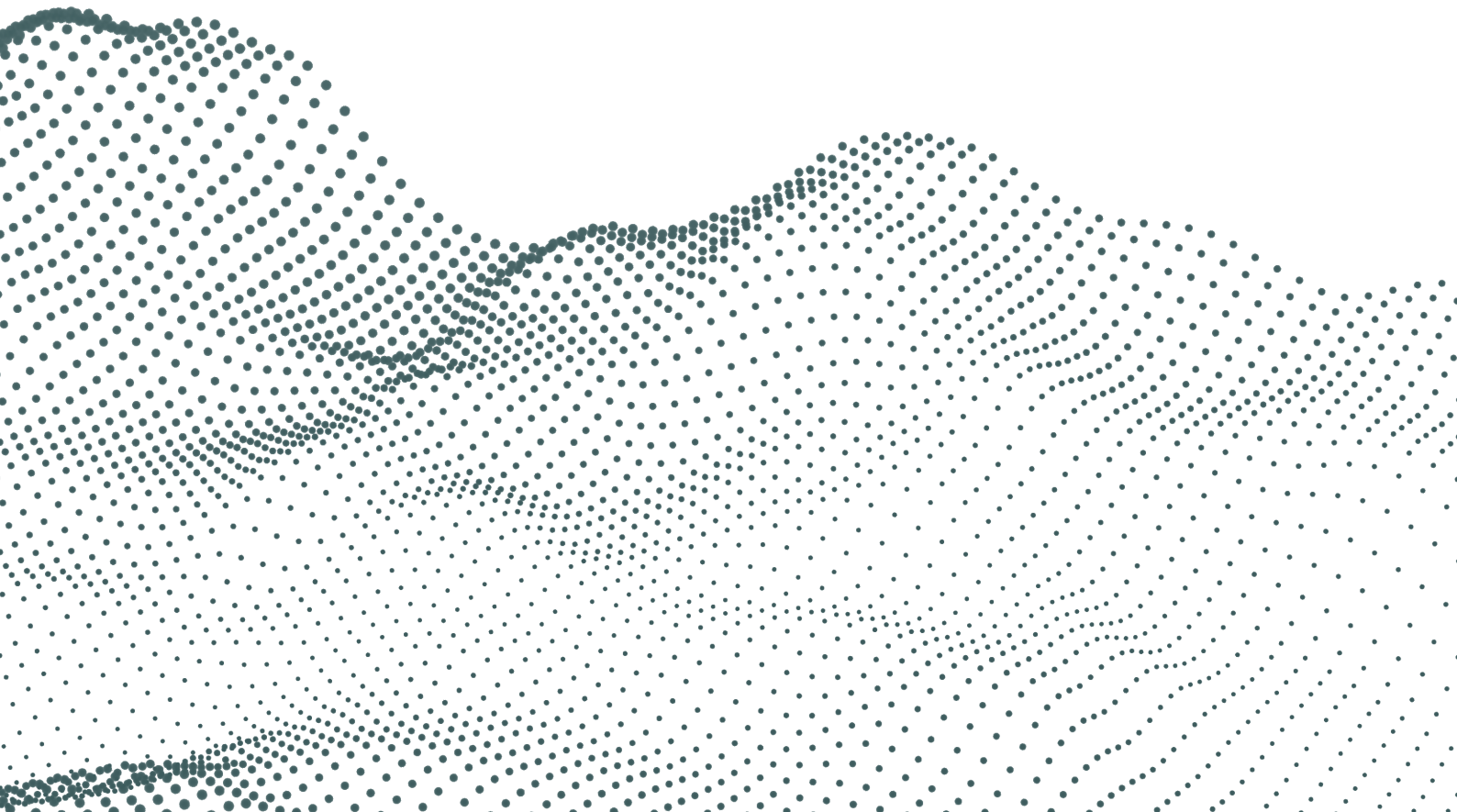
mohr stade

Transformation und Management von SQL-Workflows mit Dataform

Whitepaper



Inhalt

Veröffentlichung dieses Whitepapers: März 2023

1. Vorwort

Wenn das Unternehmen und somit das Data Team wächst, werden Themen wie Dokumentation, Sicherung von Codequalität und der Zugang zu Informationen immer wichtiger, da mehr Struktur und Abstimmung gefragt sind. Der organisatorische Aufwand für die Sicherstellung einer hohen Datenqualität auch bei zunehmender Komplexität der Datenarchitektur steigt. In diesem Zusammenhang können Tools wie Dataform helfen, indem sie eine Schnittstelle bieten, über die alle Transformationsprozesse orchestriert, verwaltet und ausgeführt werden können. Mit Dataform können Data Teams komplexe Transformationsprozesse zentral über das Data Warehouse steuern und so die Qualität einzelner Daten-Pipelines erhöhen und das Ausfallrisiko von Transformationsprozessen minimieren. In diesem Whitepaper zeigen wir anhand der Google-Cloud-Umgebung, welche Probleme Dataform im Transformationsprozess und Management komplexer SQL-Workflows löst, welche Kernfunktionalitäten das Tool bietet und wann ein Einsatz sinnvoll ist.

2. Über Dataform

Dataform ist ein Open Source Framework zur Erstellung und zum Management von SQL-Workflows. Das Tool nutzt hierfür Standards aus der Software-Entwicklung wie die Git-Versionskontrolle und Tests. Seit Ende 2020 ist Dataform nativer Bestandteil von BigQuery in der Google Cloud. Zuvor konnte Dataform als Integrated Development Environment (IDE) auch in Cloud-Data-Warehouse-Lösungen wie Amazon Redshift, Snowflake, Azure SQL Data Warehouse und Postgres genutzt werden.

2.1 Die wichtigsten Features und Funktionen

Dataform nimmt die Rolle eines “Modeling Layers” über dem Data Warehouse ein und orchestriert SQL-Workflows, die auf Daten aus dem Data Warehouse ausgeführt werden. Grundsätzlich ist die Nutzung von Dataform kostenlos. Lediglich die bei der Abfrage verwendeten Rechenressourcen werden berechnet. Für die Entwicklung von SQL-Transformationen in Dataform wird mit SQLX eine Erweiterung von SQL verwendet. Diese macht das Schreiben und Ausführen von SQL-Abfragen mit JavaScript möglich. Durch die Integration von JavaScript können so Variablen in SQL-Abfragen verwendet werden, um komplexe Transformationen durchzuführen und Beziehungen zwischen einzelnen Modellen herzustellen. Modelle können dabei lokal mit einem Code-Editor bearbeitet und mithilfe des Command Line Interface (CLI) oder bei Nutzung der Google Cloud direkt über die Google Cloud Console ausgeführt werden. Im Folgenden werden die grundlegenden Funktionen des Tools vorgestellt.

2.1.1 Versionskontrolle

Ein Dataform-Projekt lässt sich mit einem Git-Repository wie bspw. GitHub oder GitLab verbinden und somit zur Versionskontrolle des erstellten Codes nutzen. Dies erlaubt es, mit mehreren Entwicklern gleichzeitig an einem einzigen Projekt zu arbeiten, ohne dass Versionen überschrieben werden. Je Branch, in der ein Entwickler tätig ist, wird eine Kopie des Projekts und des Projektverlaufs dargestellt.

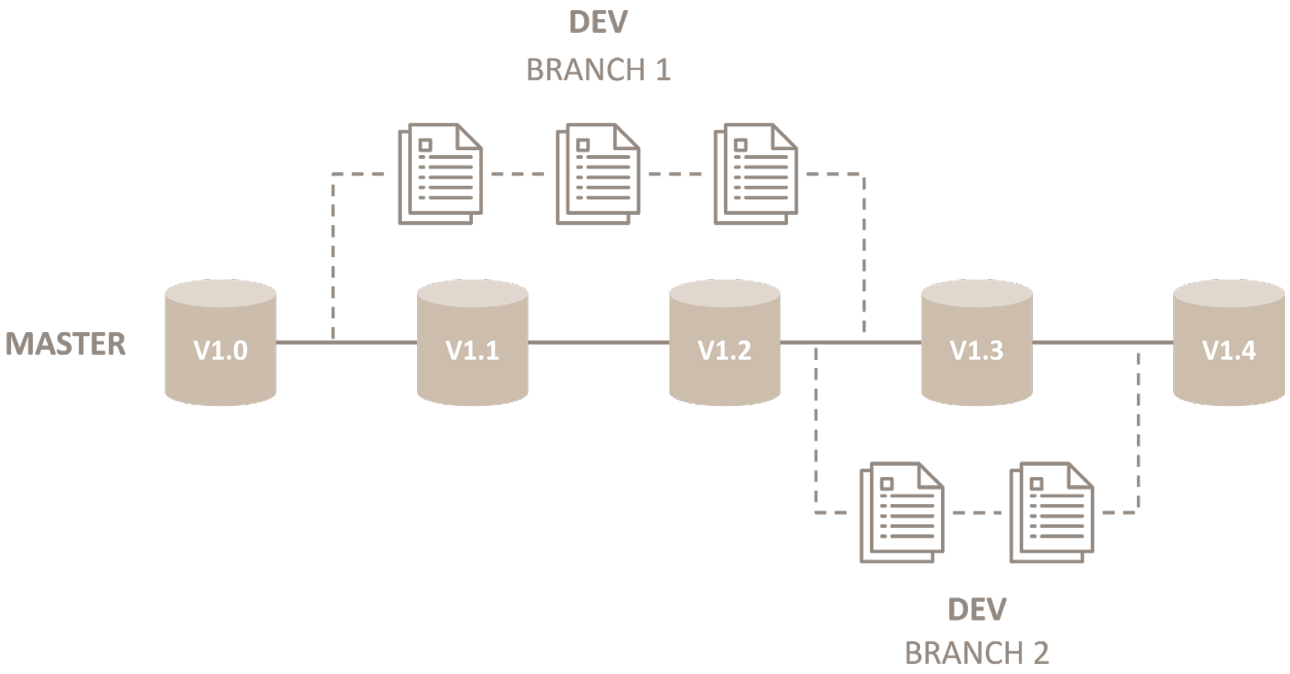


Abbildung 1: Git Version Control (vereinfacht)

Änderungen werden erst dann in der Main Branch des Repository zusammengeführt, wenn sie validiert wurden. Somit werden alle Änderungen eines Codes im Dataform-Projekt dokumentiert und sind chronologisch einsehbar.
Im Code-Editor von Dataform werden Änderungen am Code farblich hervorgehoben und als Konflikte markiert, um Abweichungen zum Code in der Main Branch aufzuzeigen. Bevor Änderungen im Code in die Main Branch übertragen werden können, müssen sie mit einem Kommentar versehen und genehmigt werden. Mithilfe von Git können Entwickler somit sämtliche Änderungen im Projektverlauf an einem Ort einsehen. Dies gewährleistet eine effizientere Zusammenarbeit von Entwicklern und Analysten und stellt sicher, dass alle SQL-Workflows und Transformationen stets auf dem aktuellsten Stand sind.

2.1.2 JavaScript-API

Dataform bietet eine JavaScript-API, mit der alle Aktionen innerhalb eines Dataform-Projekts über JavaScript definiert werden können. Die Erstellung einzelner SQL-Dateien pro Aktion entfällt somit. Zu den Funktionen der JavaScript-API zählen u.a. Declarations, Assertions, Operations und Tests. Zugehörige JavaScript-Dateien müssen im Ordner *includes/* angelegt werden, um wiederverwendbare Funktionen, Konstanten oder Makros zu definieren. Jede Datei im Ordner *includes/* wird für die Verwendung in anderen SQL- oder JavaScript-Dateien verfügbar gemacht.

JavaScript lässt sich innerhalb einzelner SQLX-Dateien über einen JavaScript-Block *js{ }* initiieren, der mittels Inline-JavaScript *${ }* in die SQL eingefügt werden kann, um die SQL-Abfrage dynamisch zu ändern. Zu beachten ist, dass Funktionen, Konstanten oder Makros, die innerhalb einer SQL-Datei definiert werden, nicht auf globaler Ebene im Projekt genutzt werden können. Dataform stellt zusätzlich einige Built-in-JavaScript-Funktionen und -Features zur Verfügung. Built-in-Funktionen weisen besondere Funktionalitäten auf und lassen sich innerhalb einer SQLX-Datei ausführen. Zu den Built-in-Funktionen zählen:

ref()

Die ref()-Funktion dient dazu, einen Verweis auf ein Dataform-Objekt (z.B. eine Tabelle oder einen View) innerhalb eines SQL-Statements zu erstellen. Hierbei wird von Dataform auf den Tabellennamen referenziert.

Im Code-Beispiel würde demnach mit ${ref("events_*")} auf die Daten in der Tabelle *events_* referenziert werden. ref() fügt außerdem den referenzierten Datensatz zu den Abhängigkeiten (Dependencies) für die Abfrage hinzu. Mehr dazu in Kapitel 2.1.3 Dependency Management.

```
config {
type: "declaration",
database: "bigquery-public-data",
schema: "ga4_obfuscated_sample_ecommerce",
name: "events_*"
}
```

Abbildung 2: Code-Beispiel Config

resolve()

Die resolve()-Funktion arbeitet ähnlich wie die ref()-Funktion, allerdings fügt sie den Datensatz nicht zu den Abhängigkeiten (Dependencies) hinzu.

self()

Die self()-Funktion gibt den vollen Namen des verwendeten Datasets (Datenquelle, Schema und Name) zurück. Sollte die Datenquelle, das Schema oder der initiale Name des Datasets im config{} Block überschrieben worden sein, gibt die Funktion self() den vollen und korrekten Namen des Datensatzes zurück.

```
create or replace function ${self()}(tsource string, medium string, campaign string) as (
    case
        when (medium = '(data deleted)')
            then 'data_deleted'
        when (tsource = 'direct' or tsource is null)
            and (regexp_contains(medium, r'^(\(not set\)|\(none\))$') or medium is null)
            then 'direct'
        when regexp_contains(campaign, r'^(.*shop.*)$')
            and regexp_contains(medium, r'^(.*cp.*|ppc|paid.*)$')
            then 'shopping_paid'
```

Abbildung 3: Code-Beispiel self() function Custom Channel Grouping

2.1.3 Dependency Management (SQLX)

In Dataform können über den Ordner */definitions* im config{} Block Abhängigkeiten (Dependencies) definiert werden, die vor der Materialisierung einer Tabelle, eines Data Quality Tests oder eines SQL-Workflows ausgeführt werden sollen. Zusätzlich nutzt Dataform u.a. die bereits erwähnte ref()-Funktion, um Abhängigkeiten zwischen den einzelnen SQL-Dateien herzustellen. Dadurch kann Dataform automatisch anhand von Join-Keys und Daten-Typen Beziehungen managen, wenn auf Quelldaten in einem Modell zugegriffen wird. In Dataform werden diese Funktionen unter anderem genutzt, um mithilfe des Compiled Graphs den Datenfluss durch das Datenmodell zu visualisieren. Bei größeren Projekten, die eine Vielzahl von Datensätzen mit komplizierten Abhängigkeiten enthalten, kann das Dependency Management helfen, den Überblick über die Gesamtstruktur des Projekts zu behalten.

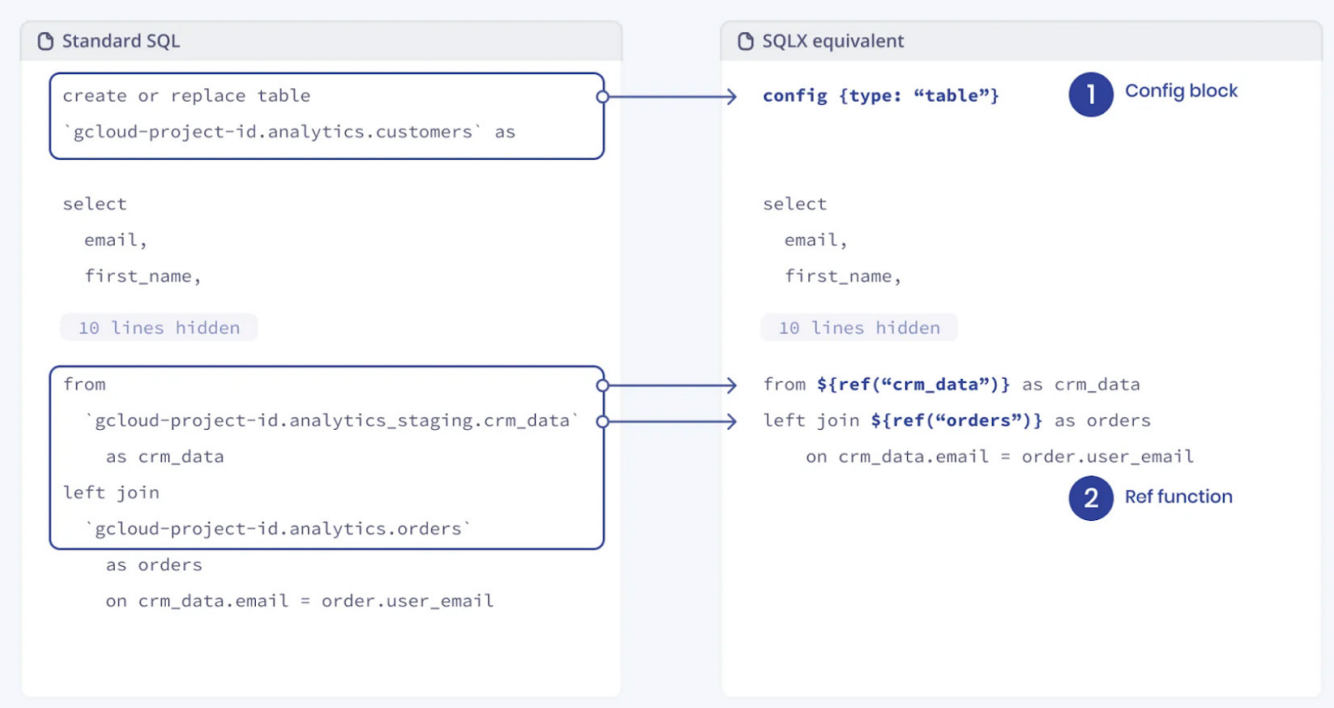


Abbildung 4: SQLX-Schema (ref)

Die ref()-Funktion kann auch verwendet werden, um die Ergebnisse von Datenmodellen mit den Ergebnissen der Quelldaten abzugleichen. In Dataform werden diese Bedingungen als Assertions bezeichnet. So lässt sich beispielsweise überprüfen, ob gewisse Attribute wie eine User-ID oder ein Timestamp einzigartig sind. Sobald Beziehungen zwischen einzelnen Datensätzen und SQL-Workflows hergestellt sind, werden sie im Compiled Graph visualisiert.

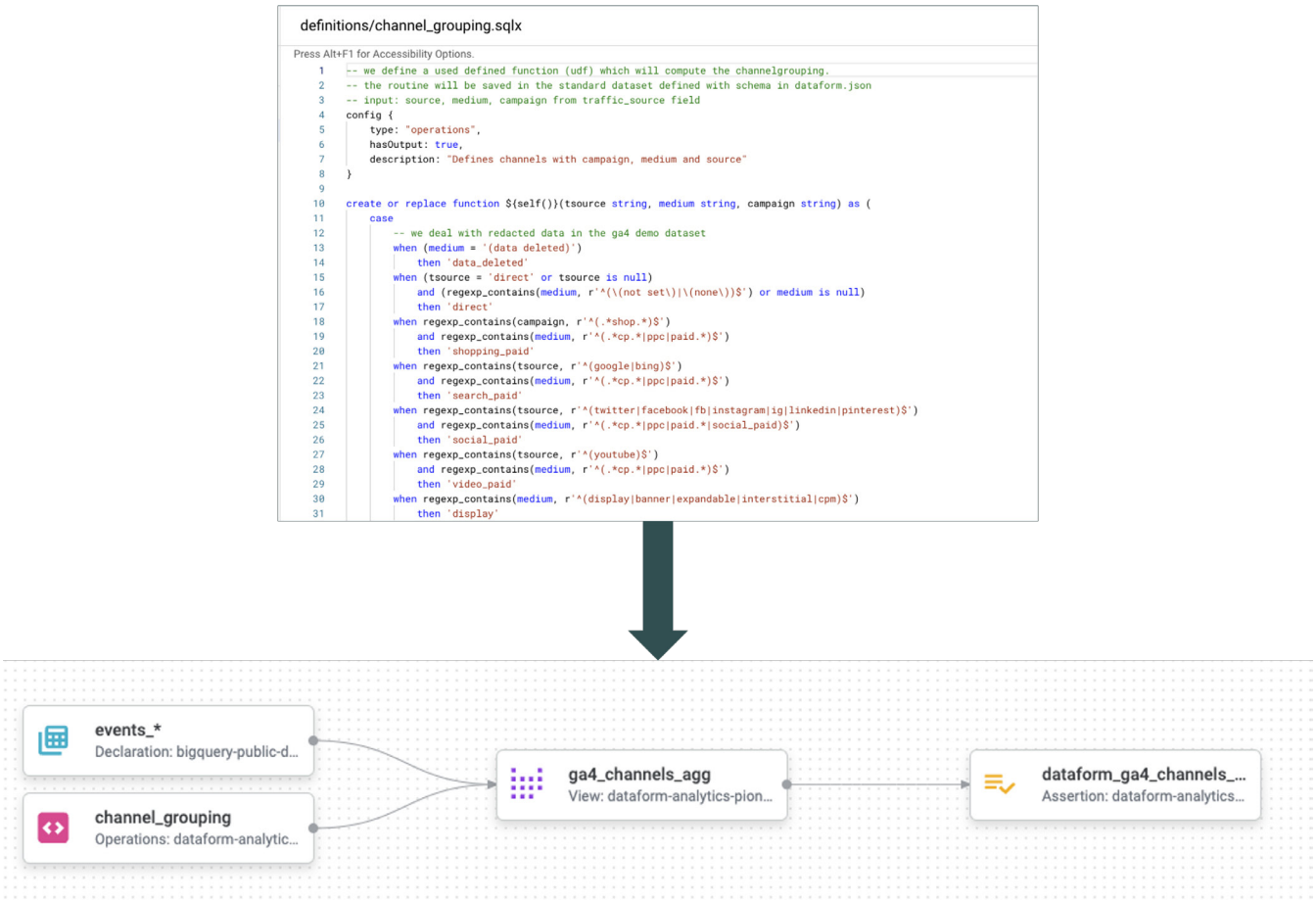


Abbildung 5: Compiled Graph (Data Lineage)

Der Compiled Graph visualisiert die Abhängigkeiten zwischen den verschiedenen Datenquellen, Transformationen und Tabellen innerhalb eines Datenmodells. Der Datenfluss von einer Quelle zur nächsten wird so nachvollziehbar. Durch diese Transparenz können Fehler in den Daten leichter identifiziert und korrigiert werden. Die Navigation durch das Datenmodell wird durch den Graphen erheblich vereinfacht, was die Wartung und Einhaltung von Compliance-Anforderungen erleichtert. Automatisierte Data Quality Tests (Assertions) können zur Überwachung der Datenqualität und zur Gewährleistung konsistenter und korrekter Datenmodelle beitragen.

2.1.4 Data Quality Tests (Assertions)

Assertions sind Bedingungen, die bei der Ausführung von SQL-Transformationen auf Zeilenebene überprüft werden, damit die Daten dem gewünschten Standard entsprechen. Sie dienen der Qualitätskontrolle von SQL-Workflows und stellen sicher, dass die Logik der Datentransformation wie erwartet funktioniert. Nach der Ausführung eines Workflows erstellt Dataform automatisch Tabellen, welche die Ergebnisse der Assertion-Abfragen enthalten.

In Dataform werden Assertions in einem eigenen Verzeichnis (*/definitions*) gespeichert. In diesem Verzeichnis können verschiedene Arten von Assertions selbst definiert oder bereits vordefinierte Assertions wie *uniqueKey(s)*, *nonNull* oder *rowConditions* genutzt werden. Es folgt ein praktisches Beispiel zur Veranschaulichung:

```
config {
  type: "table",
  description: "GA4 Data Export",
  assertions: {
    uniqueKey: ["event_timestamp"],
    nonNull: ["user pseudo id"],
    rowConditions ["ecommerce.purchase.revenue > 0"]
  }
}
SELECT * from $(ref("events_*")}
```

Abbildung 6: GA4 Data Export Assertions

Im Code-Beispiel wurden vordefinierte Assertions verwendet, um die Qualität des täglichen Data Exports aus Google Analytics 4 (GA4) zu gewährleisten. Für die Property *uniqueKey* wurde der *event_timestamp* gewählt. Somit schlägt die Assertion fehl, sobald mehr als eine Zeile im Datensatz mit dem gleichen Timestamp gefunden wird. *nonNull* definiert ein Array von einer oder mehreren Properties, die niemals als *null* ausgewiesen werden dürfen. Über *rowConditions* lassen sich benutzerdefinierte Bedingungen definieren, anhand derer alle Zeilen im Datensatz geprüft werden. Jede Bedingung stellt ein SQL-Statement dar, von dem erwartet wird, dass es als true ausgewertet wird, wenn die Assertion erfolgreich ist. Im Beispiel werden somit nur Unique User berücksichtigt, die zu einem bestimmten Zeitpunkt Umsatz > 0 Euro generiert haben. Der Workflow der Assertions stellt sich wie folgt dar:

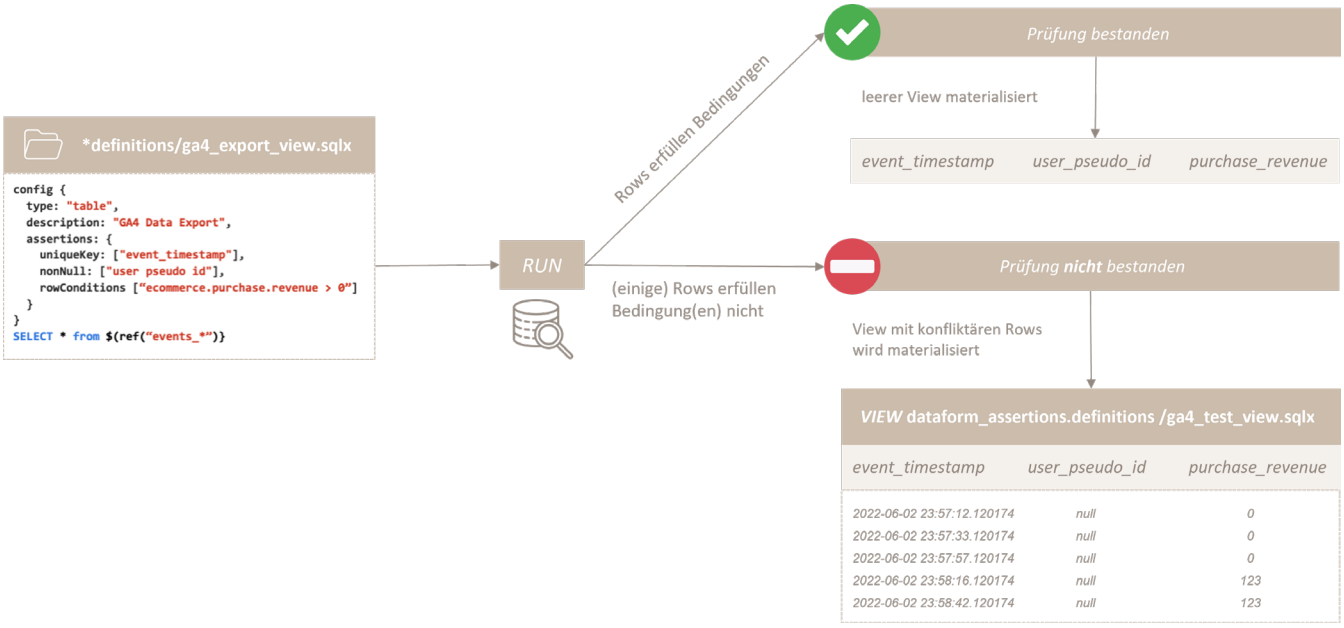


Abbildung 7: Auto-generated Assertions (Workflow)

Die im Ordner */definitions* hinterlegten Assertions werden auf dem ausgewählten Table oder View im Data Warehouse (BigQuery) ausgeführt. Dabei bietet Dataform, wie auch BigQuery, zusätzlich die Möglichkeit eines *dry_run*, um die Größe der Abfrage zu schätzen, ohne Rechenslots zu beanspruchen.

2.1.5 Automatisierte Workflows

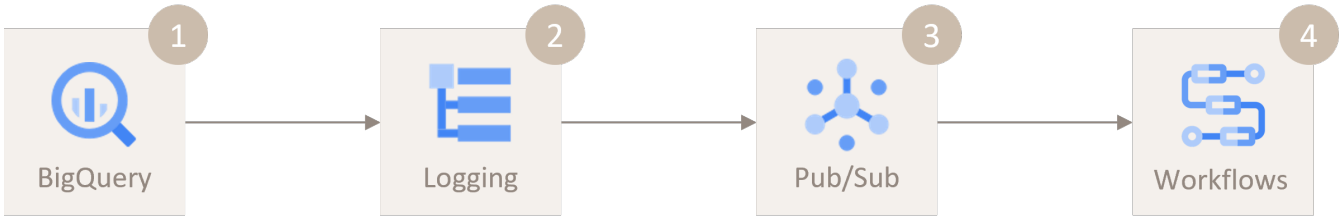


Abbildung 8: Ablauf Workflow-Automatisierung Dataform

Die Automatisierung von Dataform Executions ist aktuell nur mit Workflows auf der Google Cloud Platform möglich. Dabei wird zunächst der aktuelle Stand des Git Repositories geladen und kompiliert. Anschließend wird das Ergebnis genutzt, um den Dataform Workflow auszuführen. Im Code des Workflows muss sowohl der Name des Dataform Workspaces als auch die Branch für die Ausführung gesetzt werden. Im folgenden Workflow-Beispiel wird der Dataform Workspace attribution_ga4 mit der Branch *main* ausgeführt.

```
main:
    steps:
    - init:
        assign:
        - repository: projects/artful-shelter-265814/locations/europe-west4/repositories/attribution_ga4
    - createCompilationResult:
        call: http.post
        args:
            url: ${"https://dataform.googleapis.com/v1beta1/" + repository + "/compilationResults"}
            auth:
                type: OAuth2
            body:
                gitCommitish: main
        result: compilationResult
    - createWorkflowInvocation:
        call: http.post
        args:
            url: ${"https://dataform.googleapis.com/v1beta1/" + repository + "/workflowInvocations"}
            auth:
                type: OAuth2
            body:
                compilationResult: ${compilationResult.body.name}
        result: workflowInvocation
    - complete:
        return: ${workflowInvocation.body.name}
```

Abbildung 9: Dataform Workflow (Beispiel)

Es können nur bestimmte Bestandteile eines Dataform Workspaces ausgeführt werden. Hierfür sind Tags nutzbar, welche in den SQLX-Dateien im Config-Abschnitt gesetzt werden können. Für das Auslösen bzw. Automatisieren des Workflows können entweder verschiedene Events von Diensten wie Pub/Sub oder BigQuery über die Eventarc API genutzt oder eine zeitlich basierende Auslösung mit dem Cloud Scheduler verwendet werden.

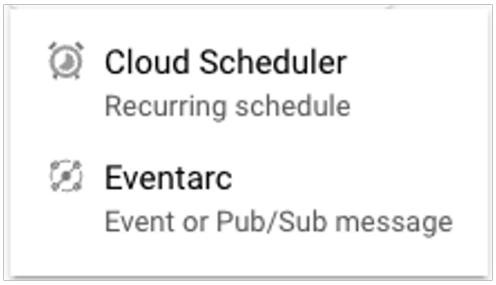


Abbildung 10: Cloud Scheduler und Eventarc

In Bezug auf Automatisierungen mit GA4-Rohdaten ist es sinnvoll, einen Log Sink anzulegen, der die Nachrichten bei Aktualisierung des genutzten GA4-Datasets enthält. Diese Logs können anschließend mit dem GCP-internen Nachrichtendienst Pub/Sub an andere Dienste wie bspw. Workflows weitergeleitet werden. Im Logs Explorer kann der Filter für die Log Sink getestet werden, um die gewünschten Ergebnisse zu erzielen. Als Ziel des Log Sinks wird ein Pub/Sub Topic definiert.

```
1 resource.labels.dataset_id="analytics_262445815"
2 log_name="projects/artful-shelter-265814/logs/cloudaudit.googleapis.com%2Factivity"
3 proto_payload.authentication_info.principal_email="firebase-measurement@system.gserviceaccount.com"
4 proto_payload.resource_name=~"projects/artful-shelter-265814/datasets/analytics_262445815/tables/events_2"
5 proto_payload.authorization_info.permission="bigquery.tables.create"
```

Abbildung 11: Log Explorer

Im obigen Beispiel ist der Filter für den Log Sink eines GA4-Datasets zu sehen. Dabei wird der Intraday Table nicht berücksichtigt, sondern nur die tägliche Aktualisierung der *events_*-Tabellen.

Google empfiehlt nach dem Prinzip “Least Privilege” die Nutzung eigener Service Accounts mit den notwendigen Rollen/Rechten für die unterschiedlichen Dienste. Dies muss für die Umsetzung der Automatisierung berücksichtigt werden. So benötigt der Service Account, welcher den Workflow ausführt, z.B. die Rechte für das Lesen von Pub/Sub Messages.

2.1.6 Incremental Tables

Dataform bietet die Möglichkeit, Incremental Tables zu erstellen. Anstatt die Tabelle jedes Mal mit der gesamten Datenquelle neu aufzubauen, werden mit Incremental Tables inkrementelle Daten aktualisiert. Ein großer Vorteil inkrementeller Datensätze besteht darin, dass Pipelines schneller abgeschlossen und daher häufiger zu geringeren Kosten ausgeführt werden können. Durch das sogenannte Micro-batching wird die Latenzzeit downstream stark verringert. Somit werden nur Änderungen an Daten aus dem letzten Job vorgenommen, wodurch nicht nur Zeit, sondern auch Rechenressourcen eingespart werden. Wichtig dabei ist, dass keine Duplikate in die Tabelle eingehen. Sicherstellen lässt sich dies beispielsweise über den config{} Block. Dort wird ein *rowKey* für jede Zeile definiert, welcher einen einzigartigen Wert widerspiegelt.

```
concat(DATE(extract(year from sessionDate), month, date_diff(sessionDate, firstSessionDate, day)+1),"-",browser,
"-",deviceCategory) as rowKey
```

Abbildung 12: Definition eines rowKey (Beispiel)

Zu den Anwendungsbereichen von Incremental Tables zählt beispielsweise die Aktualisierung von Tabellen mit Streaming-Daten oder Slowly Changing Dimensions (SCD), die sich regelmäßig ändern. Bei SCD wird durch einen Snapshot der Zustand einer Dimension zu einem bestimmten Zeitpunkt abgespeichert, was den Vergleich mit früheren Zuständen dieser Dimension erlaubt. Dadurch können Änderungen in den Daten im Zeitverlauf verfolgt und historische Analysen durchgeführt werden. Zum Beispiel lässt sich so die Änderung des Status eines Nutzers oder aber der sich aktualisierende Lieferstatus einer Bestellung rückblickend nachvollziehen.

3. Dataform Transformation Workflow

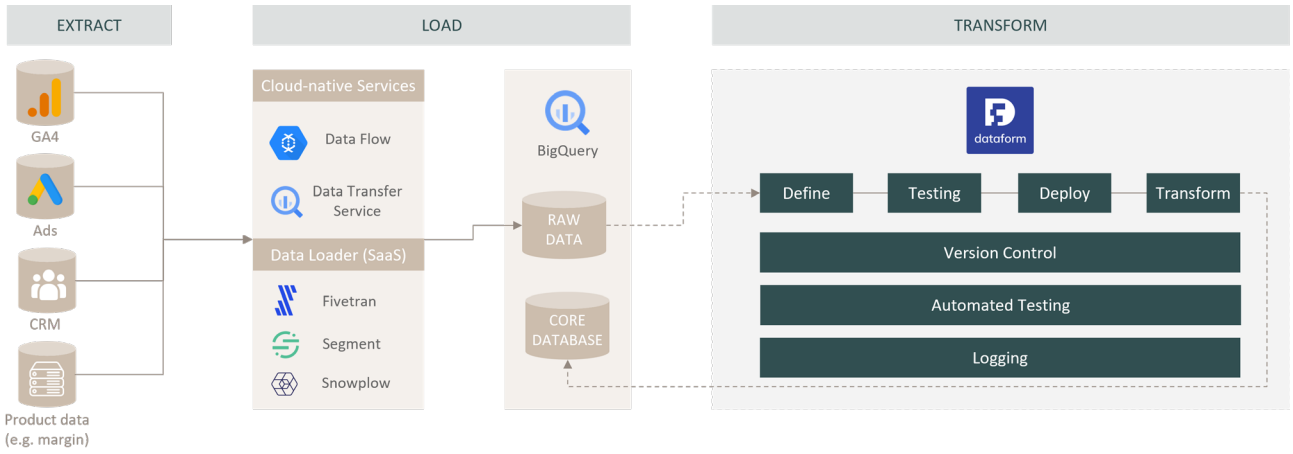


Abbildung 13: Dataform Orchestration und Transformation Workflow

Die Abbildung zeigt den grundlegenden technischen Workflow von Dataform als Bestandteil der Google Cloud. Über Data Loader wie Fivetran, Segment, Snowplow oder GCP-Services wie Dataflow und den BigQuery Data Transfer Service werden die Daten aus den relevanten Quellsystemen in BigQuery geladen. Dort werden die Daten im Rohformat des Quellsystems entgegengenommen und in Form von Tabellen oder Views abgelegt. Ab dieser Stelle fungiert Dataform als eine Art Orchestration und Modeling Layer. Die in Dataform definierten Modelle (SQLX) werden auf die abgelegten Rohdaten im Data Warehouse ausgeführt, ohne dass dabei die Transformation direkt in der Datenquelle vorgenommen wird. Anschließend werden die transformierten und getesteten Daten in einer neuen Tabelle materialisiert. Diese steht downstream für die weitere Verwendung in Marketing- und Analysetools zur Verfügung.

4. Unterschiede zu dbt (Data Build Tool)

Seit Ende letzten Jahres ist Dataform in Google BigQuery integriert und stellt somit einen integralen Bestandteil der Google Cloud Platform (GCP) dar. Funktional gibt es zwischen Dataform und dem Data Build Tool (dbt) kaum Unterschiede, da auch Dataform Git nutzt und dadurch Funktionen wie Version Control und die Dokumentation von Modellen und Transformationen ermöglicht. Der Hauptvorteil von Dataform ist, dass es durch seine Integration in BigQuery keine eigene Infrastruktur benötigt, während dbt auf einem eigenen Server betrieben wird. Zudem nutzt Dataform eine JavaScript-API, um interaktiv mit SQL-Modellen zu arbeiten, während dbt die Templating-Sprache Jinja verwendet, die der Syntax von Python ähnelt. Im Vergleich zu dbt existiert für Dataform jedoch keine so große Community, die das Tool weiterentwickelt und eine Vielzahl von Packages für diverse Use Cases zur Verfügung stellt. Allerdings bietet die Community von dbt zahlreiche Packages wie z. B. ein GA4 Community Package, welches sich mit ein paar Anpassungen leicht an die Dataform-Syntax anpassen lässt. Wenn bereits mit der GCP bzw. BigQuery gearbeitet wird, kann Dataform längerfristig eine bessere Integration bieten. Eine pauschale Aussage lässt sich jedoch nicht treffen und ist abhängig von der individuellen Ausgangslage des Unternehmens und den Kompetenzen im Data Team.

5. Fazit

Durch die Integration von Git bringt Dataform bewährte Standards und Workflows aus der Softwareentwicklung in die Welt der Datenanalyse und Datentransformation, indem es die direkte Ausführung von Code aus dem Repository über ein Command Line Interface (CLI) oder die GCP Console ermöglicht. Somit wird es Entwicklern und Analysten ermöglicht, sich auf die Implementierung der Geschäftslogik in der Datenmodellierung zu konzentrieren. Der Fokus auf Zusammenarbeit und Dokumentation fördert die Demokratisierung von Daten und Informationen im Unternehmen und schafft Transparenz über die Versionierung und Änderungen im Transformationsprozess. Die Verwendung eines SQL Orchestration Tools wie Dataform sollte demnach für jedes größere datengetriebene Unternehmen in Betracht gezogen werden, um einen stabilen Transformationsprozess mit weniger Wartungsaufwand bei gleichzeitig steigender Datenqualität sicherzustellen.

Über mohrstade

Unternehmen

mohrstade ist eine Beratung für Marketing-Technologie in München, Hamburg und Wien. mohrstade ist spezialisiert auf Projekte in den Bereichen Data Collection, Data Management, Analytics, Marketing Activation und Data Visualization. Diese Services bietet mohrstade in zertifizierten Partnerschaften mit Marketing-Software-Herstellern an.

Managing Partner

Patrick Mohr

Co-Founder & Managing Partner

Patrick ist Gründer und Geschäftführer von mohrstade. Bereits während seines Studiums für BWL, Finance und Information (MSc) sammelte er Erfahrungen im Management Consulting. Später arbeitet er als SEA Manager, Data Scientist und Analytics Consultant bei Rocket Internet, Group M und UDG. 2017 baute er schließlich den Münchner Standort von Trakken auf. Parallel arbeitet er als Dozent an Universitäten. Darüber hinaus ist er Co-Organisator von Analytics Pioneers, der größten Analytics Community im DACH-Raum.

patrick@mohrstade.de

Marcus Stade

Co-Founder & Head of Analytics

Marcus ist Gründer von mohrstade und Head of Analytics. Darüber hinaus ist er Co-Organisator von Analytics Pioneers, der größten Analytics Community im DACH-Raum. Zuvor hat er im Bereich Web-Development und Online-Marketing gearbeitet. Auf seinem Blog www.marcusstade.de schreibt er regelmäßig zu Themen der Digitalen Analyse.

marcus@mohrstade.de

mohr stade

Mohr & Stade GmbH
Landwehrstraße 2
80336 München

www.mohrstade.de