Engineer

ETL

🔄 Daten zwischen Systemen übertragen und dabei umformen

Was es ist

ETL steht für Extract, Transform, Load: Daten werden aus den Systemen kopiert, in denen sie entstehen, in die von einem anderen System erwartete Form und Bedeutung gebracht und dort geschrieben, meist in ein Data Warehouse. Die nahe verwandte Variante ELT behält die drei Schritte bei, lädt aber zuerst die Rohdaten und transformiert sie im Zielsystem. Viele Teams schreiben ETL/ELT, um beides abzudecken.

Jeder Buchstabe steht für eine eigene Aufgabe. Extract liest aus einer Quelle: einer ganzen Tabelle, einer Exportdatei, einer API oder nur den seit dem letzten Lauf geänderten Zeilen. Diese werden über einen Zeitstempel wie `updated_at` oder über Change Data Capture (CDC) gefunden, das die eigenen Aufzeichnungen der Datenbank zu jeder Änderung liest. Bei Transform werden Daten bereinigt und abgeglichen: Text wird in Datumswerte und Zahlen umgewandelt, Einheiten und Zeitzonen werden vereinheitlicht, Duplikate entfernt und Codes in Referenztabellen nachgeschlagen. Anschließend wird das Ergebnis in das Modell des Zielsystems gebracht, etwa in Fakten- und Dimensionstabellen. Load schreibt das Ergebnis, entweder als Massenimport in eine leere Tabelle oder als Zusammenführung, die vorhandene Zeilen aktualisiert und neue hinzufügt.

Die Reihenfolge der letzten beiden Schritte entscheidet, wo die Arbeit läuft. Klassisches ETL transformiert auf einem separaten Server, bevor etwas das Zielsystem erreicht. Das war sinnvoll, als Speicher und Rechenleistung im Warehouse knapp und teuer waren. ELT legt die Rohdaten in Staging-Tabellen im Zielsystem ab und führt die Transformation dort in SQL aus. Die Rohdatenkopie bleibt verfügbar. Ändert sich eine Regel, lässt sich die Transformation deshalb erneut auf die historischen Daten anwenden, ohne diese ein zweites Mal zu

extrahieren. Der Preis: Alles, was die Quelle gesendet hat, einschließlich nicht angeforderter Spalten, liegt nun im Warehouse und muss dort verwaltet und kontrolliert werden.

PostgreSQL bietet für jeden Schritt eigene Werkzeuge. `COPY` bewegt Zeilen zwischen einer Tabelle und einer Datei im Text-, CSV- oder Binärformat und ist der schnellste Weg zum Laden. Das Lesen einer Datei auf dem Server erfordert Superuser-Rechte oder die Rolle `pg_read_server_files`. Der psql-Befehl `\copy` liest die Datei dagegen auf dem Client. `INSERT ... ON CONFLICT` und `MERGE` führen neue Zeilen mit vorhandenen zusammen. Logical Decoding verwandelt das Write-Ahead Log für CDC in einen Strom von Zeilenänderungen, und eine Prozedur in PL/pgSQL kann die Transformation nahe an den Daten halten, die sie verändert.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ETL?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Zahlen kommen termingerecht dort an, wo sie gebraucht werden, ohne dass am Monatsende jemand eine Tabelle kopieren muss.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Ein Infrastrukturplan für 14 Abteilungen — 14 Forschungsabteilungen mit einem gemeinsamen Infrastrukturplan

Ein Infrastrukturplan für 14 Abteilungen Kontakt aufnehmen →

Warum es wichtig ist

Die meisten lohnenden Geschäftsfragen erfordern Daten aus mehreren Systemen. Bestellungen liegen in einer Datenbank, Zahlungen im Export eines Anbieters, Kundendaten in einem CRM. Solange sie nicht unter einheitlichen Regeln an einem Ort zusammengeführt werden, wird jede Antwort von Hand zusammengestellt, und das Vorgehen ändert sich mit der ausführenden Person. Genau diese Zusammenführung übernimmt ETL. Ein Warehouse ist nur so korrekt wie die Ladevorgänge, die es füllen.

Im Transformationsschritt hält ein Unternehmen außerdem fest, was seine Daten bedeuten. Welche Zeilen zählen als Verkauf, welchem Tag wird eine Zahlung um Mitternacht zugerechnet, welcher Datensatz hat Vorrang, wenn zwei Kundeneinträge übereinstimmen? Wird das einmal im Ladevorgang entschieden,

gilt die Antwort für jeden nachgelagerten Bericht. Entscheidet jeder Bericht selbst, laufen die Antworten auseinander, und zwei Dashboards zeigen zwei Summen.

Die Art der Extraktion bestimmt, wie der Aufwand wächst. Ein vollständiger Neuladevorgang kopiert jedes Mal alles. Das ist einfach und korrigiert sich selbst, weil der Fehler von gestern heute Nacht überschrieben wird. Die Kosten steigen jedoch mit der gesamten Datenmenge. Ein inkrementeller Ladevorgang überträgt nur Änderungen; seine Kosten steigen deshalb mit der Änderungsrate. Er muss sich merken, wo der letzte Lauf aufgehört hat, in der Quelle gelöschte Zeilen erkennen und verspätete Zeilen mit alten Zeitstempeln annehmen. Die meisten Systeme beginnen mit vollständigem Neuladen und stellen Tabelle für Tabelle um, sobald die Nacht dafür nicht mehr ausreicht.

Der laufende Aufwand entsteht dadurch, dass Ladevorgänge unbeaufsichtigt Code ausführen, meist nachts und gegen Quellen, die sich ohne Ankündigung ändern. Sie brauchen Protokolle darüber, was lief und wie viele Zeilen übertragen wurden, Alarme bei einem abgebrochenen Lauf und vor allem eine Eigenschaft: Ein mit derselben Eingabe wiederholter Lauf sollte das Ziel exakt so hinterlassen wie ein einzelner Lauf. Ein solcher Ladevorgang lässt sich nach jedem Fehler wiederholen, ohne dass vorher jemand nachsehen muss.

Wie es in der Praxis aussieht

Zersplitterte Pipelines verzögerten geschäftskritische Berichte in mehreren Entwicklungs- und Produktionsumgebungen mit PostgreSQL. 600 ETL/ELT-Pipelines wurden in PL/pgSQL geschrieben, um sie zu ersetzen: ETL in der Größenordnung Hunderter routinemäßiger Ladevorgänge. Primär- und Fremdschlüsselbedingungen sicherten bei den Transformationen die referenzielle Integrität, eindeutige und zusammengesetzte Indizes unterstützten die Joins, und Ausnahmebehandlung mit Transaktionskontrolle rollte einen fehlgeschlagenen Lauf zurück. Inkrementelle Ladevorgänge nutzten CDC und zeitstempelbasierte Deltas; Ausführungsprotokolle und Auditing unterstützten Nachverfolgung und Fehlersuche. Geliefert wurden Transformationslogik, Integritätsbedingungen, inkrementelles Laden und Ausführungsprotokolle. Der Nachweis enthält weder einen Vergleichsbenchmark noch eine berechnete Fehlerquote.

Eine Migration ist ETL, das einmal läuft. Der Umzug einer Kartenanwendung von Microsoft SQL Server zu PostgreSQL mit PostGIS erforderte, über 50 Millionen Geodatensätze durch ETL-Pipelines zu übertragen. Dabei rechneten die Transformationsschritte jede Koordinate in das Referenzsystem EPSG:4326 um. Nachdem die Datensätze im Ziel angekommen waren, lieferten räumliche Abfragen dort mit GiST- und SP-GiST-Indizes und einer abgestimmten Konfiguration statt durchschnittlich 2,5 Sekunden Ergebnisse in unter 250 Millisekunden, also zehnmal schneller. Die CPU-Last im Backend sank um 65 %.

Über eine Netzwerkgrenze hinweg zeigen sechs ETL/ELT-Pipelines alle drei im Namen enthaltenen Arbeitsschritte in der Praxis. Die Extraktion bestand aus geplanten Downloads komprimierter Archive von einem nur über ein IPsec-VPN erreichbaren Server. Skripte in Bash und Python sortierten die darin enthaltenen Parquet-, CSV- und .bak-Dateien nach Typ und Spaltenlayout; die .bak-Sicherungen wurden in einer lokalen Instanz von MSSQL Server wiederhergestellt. Je nach Format wurden die Daten mit bcp, psql oder SSIS in Staging-Schemas in MSSQL und PostgreSQL geladen, durch Prozeduren in PL/pgSQL und T-SQL bereinigt, normalisiert und angereichert, dann mit gzip komprimiert und über einen Bucket in Google Cloud Storage nach Google BigQuery übertragen. Arbeit, die von Hand mindestens drei Stunden gedauert hatte, lief von Anfang bis Ende in unter 20 Minuten. Die Synchronisierung wurde von wöchentlich auf täglich umgestellt.

Wo es schiefgeht

Ein wiederholter Lauf lädt dieselben Zeilen doppelt. Ein Job fügt mit einem einfachen `INSERT` ein, bricht nach der Hälfte wegen eines Netzwerkfehlers ab, und der Scheduler startet ihn erneut von vorn. Die beim ersten Mal geschriebenen Zeilen stehen nun doppelt in der Tabelle, und jede darauf aufbauende Summe ist zu hoch. Geben Sie dem Ziel einen Schlüssel, der eine Zeile fachlich identifiziert, und führen Sie den Stapel mit `INSERT ... ON CONFLICT DO UPDATE` zusammen. PostgreSQL dokumentiert dies auch bei hoher Nebenläufigkeit als atomare Einfüge- oder Aktualisierungsoperation. Alternativ lässt sich `MERGE` verwenden. Eine Einschränkung: Eine einzelne Anweisung mit `ON CONFLICT DO UPDATE` weigert sich, dieselbe Zeile zweimal zu ändern, und meldet eine Kardinalitätsverletzung. Ein Stapel mit wiederholten Schlüsseln muss daher zuerst auf eine Zeile je Schlüssel reduziert werden.

```
INSERT INTO meter_reading (meter_id, read_at, kwh)
SELECT DISTINCT ON (meter_id, read_at) meter_id, read_at, kwh
FROM staging_meter_reading
```

```
ORDER BY meter_id, read_at, received_at DESC
ON CONFLICT (meter_id, read_at)
DO UPDATE SET kwh = EXCLUDED.kwh;
```

Ein Massenimport bricht kurz vor dem Ende ab und hinterlässt eine vollere Festplatte. `COPY FROM` schreibt die Zeilen nach und nach in die Tabelle. Bricht der Befehl an einer fehlerhaften Zeile ab, bleiben diese Zeilen im Zustand „gelöscht“ zurück: für Abfragen unsichtbar, aber auf der Platte, bis `VACUUM` den Platz freigibt. Ein Fehler kurz vor dem Ende einer großen Datei kann viel Speicher verschwenden. PostgreSQL 17 führte `ON_ERROR ignore` ein. Damit werden Zeilen in Text- oder CSV-Eingaben übersprungen, deren Werte sich nicht in den Typ ihrer Spalte umwandeln lassen. PostgreSQL 18 ergänzte `REJECT_LIMIT`, um die Zahl der übersprungenen Zeilen zu begrenzen. Das gilt nur für Typumwandlungen; eine Zeile, die eine Integritätsbedingung verletzt, stoppt den Befehl weiterhin. Wer zuerst in eine Staging-Tabelle lädt, dort prüft und die guten Zeilen in einer Transaktion in die eigentliche Tabelle überträgt, hält eine schlechte Datei von den Daten fern, die Menschen lesen.

Change Data Capture füllt die Festplatte. Logical Decoding liest Änderungen über einen Replikations-Slot. Dieser hält jeden Teil des Write-Ahead Logs vor, den sein Konsument noch braucht, unabhängig davon, ob ein Konsument verbunden ist. Stoppt der Leser, wächst das Log, bis die Platte voll ist. In extremen Fällen fährt der Server herunter, um einen Überlauf der Transaktions-IDs zu verhindern. Die PostgreSQL-Dokumentation sagt klar: Ein nicht mehr benötigter Slot sollte entfernt werden. Nach einem Absturz kann ein Slot zudem die letzten Änderungen erneut senden. Der Konsument muss daher mit Duplikaten umgehen können; es ist wieder dieselbe Eigenschaft für wiederholte Läufe.

Der erste Bericht am Morgen ist nach einem großen Ladevorgang langsam. Der Planer entscheidet anhand von Statistiken zu jeder Tabelle, wie er eine Abfrage ausführt. Ein Massenimport verändert die Daten, auf denen sie beruhen. Autovacuum kann die Statistiken selbst aktualisieren, aber nicht immer vor dem ersten Bericht. `ANALYZE` als letzter Schritt des Ladevorgangs behebt das. Für sehr große Ladevorgänge in eine vorhandene Tabelle ergänzt der PostgreSQL-Leitfaden zum Befüllen einer Datenbank, dass sich das Entfernen und anschließende Neuanlegen von Indizes und Fremdschlüsseln lohnen kann: Sie gesammelt aufzubauen ist schneller, als sie Zeile für Zeile zu aktualisieren. Während sie fehlen, entfallen jedoch auch die Fehlerprüfungen. Vor dem Entfernen eines eindeutigen Index ist deshalb besondere Vorsicht angebracht.

Weiterführende Lektüre

- COPY — der Befehl für Massenimporte, seine Formate und Optionen sowie die Rückstände eines fehlgeschlagenen Ladevorgangs.
- Befüllen einer Datenbank — PostgreSQL-Empfehlungen für große Ladevorgänge, von einer einzelnen Transaktion und `COPY` bis zum Neuaufbau der Indizes und anschließendem `ANALYZE`.
- MERGE — Einfügen, Aktualisieren und Löschen in einer Anweisung, abhängig davon, wie jede Quellzeile zum Ziel passt.
- INSERT — die Klausel `ON CONFLICT`, ihr atomares Upsert und das Verbot, eine Zeile zweimal zu ändern.
- Konzepte des Logical Decoding — wie Änderungen aus dem Write-Ahead Log gelesen werden und warum ein ungenutzter Replikations-Slot entfernt werden muss.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Bringen Sie einen repräsentativen Datensatz, die Fragen, die er beantworten muss, und den aktuellen Engpass mit. Wir können vereinbaren, wie Qualität, Durchsatz und Wiederherstellung gemessen werden.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ETL?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/etl/