Engineer

ETL

🔄 システム間でデータを移し、途中で形を整える処理

どのようなものか

ETLは、extract、transform、loadの略です。データを生み出すシステムからコピーし、別のシステムが求める形と意味へ整え、そこへ書き込みます。多くの場合、書き込み先はデータウェアハウスです。近い方式のELTも、三つの段階は同じですが、先に生データを読み込み、転送先の中で変換します。両方を含め、ETL/ELTと書くチームも多くあります。

各文字が、別の仕事を表します。抽出は、元のデータを読みます。テーブル全体、エクスポートファイル、API、前回から変わった行だけ、といった対象です。変更した行は、`updated_at`のタイムスタンプや、変更データキャプチャー、CDCで見つけます。CDCは、変更のたびにデータベース自身が残す記録を読みます。変換では、データを整理し、整合させます。文字列を日付や数値に変え、単位やタイムゾーンを統一し、重複を除き、参照テーブルでコードを引き、ファクトテーブルやディメンションテーブルなど、転送先が使うモデルへ組み替えます。読み込みは、結果を書き込みます。空のテーブルへの一括挿入か、既存行を更新し、新しい行を加えるマージです。

最後の二段階の順序で、処理する場所が決まります。従来のETLは、転送先へ何かが届く前に、別のサーバーで変換します。ウェアハウスの保存領域と計算資源が、乏しく高価だった時代には、合理的でした。ELTは、転送先のステージングテーブルへ生データを置き、SQLで変換します。生の複製が残るため、規則が変わっても、再抽出せず、履歴全体へ変換をやり直せます。代償は、誰も求めていなかった列を含め、元のシステムが送ったすべてがウェアハウスへ入り、そこで管理が必要になることです。

PostgreSQLは、各段階のための道具を備えます。`COPY`は、テーブルと、テキスト、CSV、バイナリー形式のファイルの間で行を移し、最速の読み込み方法です。サーバー上のファイルを読むには、スーパーユーザー権限か、`pg_read_server_files`ロールが必要です。一方、psqlの`\copy`コマンドは、クラ

イアント側のファイルを読みます。`INSERT ... ON CONFLICT` と `MERGE` は、新しい行を既存の行へ組み込みます。論理デコーディングは、先行書き込みログを、CDCで使う行変更のストリームへ変えます。PL/pgSQLのプロシージャーなら、変えるデータの近くへ、変換処理を置けます。

概念を判断に生かす

「ETLとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 月末に誰かが表計算ファイルをコピーしなくても、決まった時刻に必要な場所へ数値が届く仕組みをつくります。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 14部門を支える共通インフラ計画 — 14 共通インフラ計画の対象となった研究部門

14部門を支える共通インフラ計画 お問い合わせ →

なぜ重要なのか

事業上、問う価値のある問いの多くは、複数のシステムのデータを必要とします。注文は一つのデータベース、支払いは事業者のエクスポート、顧客レコードはCRMにあります。何らかの処理が、同じ規則の下で一か所へ集めるまでは、答えを毎回、人が組み立てます。やり方も、担当者で変わります。その役割がETLです。ウェアハウスの正しさは、そこへデータを入れる処理の正しさに左右されます。

変換は、データの意味を事業側が明記する場所でもあります。どの行を売上と数えるか、深夜零時の支払いをどの日に含めるか、二つの顧客レコードが一致したら、どちらを優先するか。一度、読み込みで決めれば、後のすべてのレポートに同じ答えが適用されます。各レポートに任せると、答えがずれ、二つのダッシュボードに違う合計が出ます。

抽出の仕方で、処理量の増え方も決まります。全件の再読み込みは、毎回すべてをコピーします。単純で、昨日の誤りを今夜上書きするため、自己修正できます。ただし、処理の負担はデータ全体の大きさとともに増えます。差分の読み込みは、変化したものだけを移すため、処理の負担は単位時間あたりの変更量に応じて増えます。前回どこで止まったかを覚え、元で削除された行を見つけ、古いタイムスタンプで遅れて届く行も受け入れる必要があります。多くのシステムは、全件の再読み込みから始め、夜間の処理時間内に収まらなくなったら、テーブルごとに切り替えます。

継続的な負担は、読み込みが、予告なく変わるデータソースを相手に、通常は夜、人の監視なしで動くコードであることです。何を実行し、何行を移したかを示すログ、実行が止まったときのアラートが必要です。特に重要なのは、同じ入力で実行を繰り返しても、一度だけ実行したときと、転送先をまったく同じ状態にする性質です。その性質があれば、どんな失敗の後でも、人が先に調べずに、再試行できます。

実際の活用例

複数のPostgreSQLの開発・本番環境で、分断されたパイプラインが、事業上重要な報告を遅らせていました。置き換えるために、600本のETL/ELTパイプラインがPL/pgSQLで作成されました。日常的な読み込みが数百になる規模のETLです。主キーと外部キーの制約で、変換中の参照整合性を守り、一意索引と複合索引で結合を支えました。例外処理とトランザクション制御で、失敗した実行をロールバックしました。差分の読み込みは、CDCとタイムスタンプによる差分を使い、実行ログと監査で、追跡やデバッグを支えました。納品した仕事は、変換ロジック、制約、差分の読み込み、実行ログです。記録は、比較ベンチマークやエラー率の計算を示すものではありません。

移行は、一度だけ実行するETLです。地図アプリをMicrosoft SQL Serverから、PostGISを使うPostgreSQLへ移すために、5,000万件を超える空間レコードをETLパイプラインで運びました。変換段階で、すべての座標をEPSG:4326の参照系へ変えました。転送先へ入った後、GiSTとSP-GiSTの索引と、調整した設定により、平均2.5秒だった空間クエリーが、250ミリ秒未満、十倍の速さで返るようになりました。バックエンドのCPU負荷も65%下がりました。

ネットワークの境界を越える六本のETL/ELTパイプラインでは、名前の各文字が表す仕事を、すべて見られます。抽出では、IPsec VPN経由でしか届かないサーバーから、圧縮アーカイブを予定に従ってダウンロードしました。BashとPythonのスクリプトで、中のParquet、CSV、.bakファイルを、種類と列の構成で分け、.bakのバックアップは、ローカルのMSSQL Serverインスタンスへ復元しました。形式に応じてbcp、psql、SSISで、MSSQLとPostgreSQLのステージングスキーマへ読み込みました。PL/pgSQLとT-SQLのプロシージャーで、整理、正規化、情報の補完を行い、gzipで圧縮し、Google Cloud Storageのバケットを通してGoogle BigQueryへ運びました。手作業で少なくとも三時間かかった仕事が、最初から最後まで20分未満で動き、同期は毎週から毎日へ変わりました。

問題が起きるところ

再試行で、同じ行を二回読み込みます。 単純な `INSERT` で挿入するジョブが、ネットワークエラーで途中で止まり、スケジューラーが最初からやり直します。初回に入った行が二重になり、それに基づくすべての合計が大きくなります。業務上の意味で行を識別するキーを転送先に持たせ、 `INSERT ... ON CONFLICT DO UPDATE` でバッチを取り込みます。PostgreSQLの文書は、同時実行が多い場合でも、原子的な挿入または更新と説明します。 `MERGE` も使えます。ただし、一つの `ON CONFLICT DO UPDATE` 文は、同じ行を二回変えることを拒み、カーディナリティー違反を起こします。キーが重複するバッチは、先にキーごとに一行へまとめる必要があります。

```
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;
```

一括読み込みが終わり近くで止まり、ディスクを以前より埋めます。 `COPY FROM` は、処理しながら行をテーブルへ書き込みます。不正な行で止まると、書いた行は、削除済みの状態で残ります。クエリーには見えませんが、 `VACUUM` が回収するまで、ディスク上にはあります。大きなファイルの後半で失敗すると、大量の領域を無駄にしかねません。PostgreSQL 17で追加された `ON_ERROR ignore` は、テキストやCSVの入力で、値を列の型へ変換できない行を飛ばします。PostgreSQL 18は、飛ばせる行数を制限する `REJECT_LIMIT` を追加しました。対象は、型変換だけです。制約に違反する行では、今もコマンドが止まります。先にステージングテーブルへ読み、そこで確かめ、正しい行を一つのトランザクションで本当のテーブルへ移せば、不正なファイルが、利用者の読むデータへ入りません。

変更データキャプチャーが、ディスクを埋めます。 論理デコーディングは、レプリケーションスロットから変更を読みます。スロットは、利用する側がまだ必要な先行書き込みログを、接続の有無にかかわらず、すべて保持します。読み手が止まると、ログが積み上がり、ディスクが一杯になります。極端な場合は、トランザクションIDの周回を防ぐために、サーバーが停止します。PostgreSQLの文書は、不要になったスロットを削除すべきと明記します。クラッシュ後には、スロットが最近の変更を二度送ることもあります。受け手は重複を許容しなければなりません。これも、実行を繰り返せるという、同じ性質です。

大量に読み込んだ後、朝一番のレポートが遅くなります。 プランナーは、各テーブルの統計から、クエリーの実行方法を選びます。一括読み込みは、その統計の元になるデータを変えます。autovacuumが自動で更新することもありますが、最初のレポートより先とは限りません。読み込みの最後に `ANALYZE` を実行すれば、直せます。既存テーブルへ非常に大量に入れる場合、PostgreSQLのデータベース投入ガイドは、索引と外部キーを削除し、後で作り直す方法も、有効な場合があるとします。一行ずつ更新するより、まとめて作る方が速いためです。ただし、削除中はエラーの検査を失うので、一意索引を消す前には、慎重な検討が必要です。

参考資料

- COPY — 一括読み込みのコマンド、形式、オプションと、失敗した読み込みが残すものです。
- データベースへの投入 — 一つのトランザクションと `COPY` から、索引の再構築、最後の `ANALYZE` まで、大量に読み込む際のPostgreSQLの指針です。
- MERGE — 元の各行が転送先とどう一致するかに従い、一つの文で挿入、更新、削除を行います。
- INSERT — `ON CONFLICT` 句、原子的な挿入または更新、同じ行を二回変更しない規則です。
- 論理デコーディングの概念 — 先行書き込みログから変更を読む方法と、不要なレプリケーションスロットを削除する理由です。

用語集

実際の制約に合わせて選ぶ

代表的なデータセット、そこから答えを得たい問い、現在のボトルネックをお知らせください。品質、処理量、復旧の測り方を合意できます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「ETLとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/etl/