Engineer

ETL

🔄 déplacer des données entre systèmes en les remodelant

Ce que c’est

ETL signifie extract, transform, load : extraire, transformer, charger. Ce travail consiste à copier les données depuis les systèmes qui les produisent, à leur donner la forme et le sens attendus par un autre système, puis à les y écrire, le plus souvent dans un entrepôt de données. Une variante proche, ELT, conserve ces trois étapes mais charge d’abord les données brutes et les transforme dans le système cible. Beaucoup d’équipes écrivent ETL/ELT pour désigner les deux.

Chaque lettre correspond à un travail distinct. L’extraction lit une source : une table entière, un fichier exporté, une API ou seulement les lignes modifiées depuis la dernière exécution. Elle les repère par un horodatage `updated_at` ou par la capture des changements de données, CDC, qui lit le journal que la base tient de chaque modification. La transformation nettoie et rapproche les données : convertir le texte en dates et en nombres, harmoniser unités et fuseaux horaires, supprimer les doublons, rechercher les codes dans des tables de référence et remodeler le résultat selon le modèle cible, par exemple en tables de faits et de dimensions. Le chargement écrit le résultat, soit par insertion en masse dans une table vide, soit par fusion mettant à jour les lignes existantes et ajoutant les nouvelles.

L’ordre des deux dernières étapes détermine où le travail s’exécute. L’ETL classique transforme sur un serveur distinct avant que quoi que ce soit n’arrive à destination. Cela avait du sens quand le stockage et le calcul de l’entrepôt étaient rares et coûteux. L’ELT dépose les données brutes dans des tables intermédiaires de la cible, puis les y transforme en SQL. La copie brute reste disponible : si une règle change, on peut retraiter l’historique sans l’extraire une seconde fois. En contrepartie, tout ce qu’a envoyé la source, y compris les colonnes que personne n’a demandées, réside désormais dans l’entrepôt et doit y être gouverné.

PostgreSQL dispose de ses propres outils pour chaque étape. `COPY` transfère des lignes entre une table et un fichier texte, CSV ou binaire ; c'est le moyen de chargement le plus rapide. Lire un fichier sur le serveur exige les droits de superutilisateur ou le rôle `pg_read_server_files`, tandis que la commande psql `\copy` lit le fichier côté client. `INSERT ... ON CONFLICT` et `MERGE` intègrent les nouvelles lignes aux données existantes. Le décodage logique transforme le journal d'écriture anticipée en flux de modifications de lignes pour la CDC. Une procédure PL/pgSQL peut garder la logique de transformation au plus près des données qu'elle modifie.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que l'ETL ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des chiffres disponibles là où il faut, à la fréquence prévue, sans recopier un tableur à la fin du mois.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Un plan d'infrastructure pour 14 départements — 14 départements de recherche sur un même plan d'infrastructure

Un plan d'infrastructure pour 14 départements Contactez-nous →

Pourquoi c'est important

La plupart des questions utiles à une entreprise nécessitent des données de plusieurs systèmes. Les commandes se trouvent dans une base, les paiements dans l'export d'un fournisseur, les fiches clients dans un CRM. Tant que rien ne les réunit selon un ensemble commun de règles, chaque réponse est assemblée à la main, et la méthode change avec la personne qui s'en charge. L'ETL remplit ce rôle. Un entrepôt ne peut être plus juste que les chargements qui l'alimentent.

La transformation est aussi l'étape où l'entreprise consigne le sens de ses données. Quelles lignes comptent comme vente, à quel jour rattacher un paiement de minuit, quelle fiche l'emporte lorsque deux fiches clients correspondent : une décision prise une fois dans le chargement s'applique à tous les rapports en aval. Si chaque rapport en décide, les réponses divergent et deux tableaux de bord affichent deux totaux.

Le mode d'extraction détermine comment le coût augmente. Un rechargement complet recopie tout à chaque fois. Simple, il se corrige lui-même, puisque l'erreur d'hier est remplacée cette nuit, mais son coût croît avec le volume total des données. Un chargement incrémental ne déplace que les changements ; son coût croît donc avec le rythme des changements. Il doit mémoriser où s'est

arrêtée l'exécution précédente, détecter les lignes supprimées à la source et accepter celles qui arrivent tard avec d'anciens horodatages. La plupart des systèmes commencent par des rechargements complets, puis basculent table par table quand la nuit ne suffit plus.

Le coût récurrent tient à ce que les chargements sont du code exécuté sans surveillance, généralement la nuit, à partir de sources qui changent sans prévenir. Il leur faut des journaux indiquant ce qui s'est exécuté et combien de lignes ont été déplacées, des alertes en cas d'arrêt et, surtout, une propriété : répéter une exécution avec les mêmes données d'entrée devrait laisser la cible exactement dans l'état qu'aurait produit une seule exécution. Un tel chargement peut être relancé après tout échec sans vérification préalable par une personne.

À quoi cela ressemble en pratique

Des pipelines fragmentés retardaient les rapports essentiels à l'activité dans plusieurs environnements PostgreSQL de développement et de production. 600 pipelines ETL/ELT ont été écrits en PL/pgSQL pour les remplacer : de l'ETL à l'échelle de centaines de chargements courants. Des contraintes de clé primaire et étrangère protégeaient l'intégrité référentielle pendant les transformations. Des index uniques et composites soutenaient les jointures, et la gestion des exceptions avec contrôle transactionnel annulait l'exécution en cas d'erreur. Les chargements incrémentaux utilisaient la CDC et des deltas fondés sur les horodatages ; la journalisation des exécutions et l'audit facilitaient le suivi et le débogage. Le travail livré comprend la logique de transformation, les contraintes, le chargement incrémental et la journalisation des exécutions. Le dossier n'établit ni benchmark comparatif ni calcul de taux d'erreur.

Une migration est un ETL exécuté une fois. Le passage d'une application cartographique de Microsoft SQL Server à PostgreSQL avec PostGIS a fait transiter plus de 50 millions d'enregistrements spatiaux par des pipelines ETL. Les transformations ont converti toutes les coordonnées vers le système de référence EPSG:4326. Une fois les données chargées, avec des index GiST et SP-GiST et une configuration ajustée côté cible, les requêtes spatiales, auparavant de 2,5 secondes en moyenne, répondaient en moins de 250 millisecondes, dix fois plus vite. La charge CPU du backend a diminué de 65 %.

À travers une frontière réseau, six pipelines ETL/ELT illustrent chaque lettre du nom. L'extraction consistait à télécharger à intervalles programmés des archives compressées depuis un serveur accessible seulement par un VPN IPsec. Des

scripts Bash et Python triaient les fichiers Parquet, CSV et .bak qu'elles contenaient selon leur type et la disposition des colonnes ; les sauvegardes .bak étaient restaurées dans une instance locale de MSSQL Server. Les données étaient chargées dans des schémas intermédiaires MSSQL et PostgreSQL avec bcp, psql ou SSIS selon le format, nettoyées, normalisées et enrichies par des procédures PL/pgSQL et T-SQL, puis compressées avec gzip et acheminées par un bucket Google Cloud Storage vers Google BigQuery. Le travail qui prenait au moins trois heures à la main s'exécutait en moins de 20 minutes de bout en bout. La synchronisation est passée d'hebdomadaire à quotidienne.

Où les problèmes surviennent

Une relance charge deux fois les mêmes lignes. Une tâche insère avec un simple `INSERT`, s'arrête à mi-chemin sur une erreur réseau, puis l'ordonnanceur la relance depuis le début. Les lignes déjà arrivées sont alors en double, et tous les totaux fondés sur elles sont trop élevés. Donnez à la cible une clé identifiant la ligne en termes métier, puis intégrez le lot avec `INSERT ... ON CONFLICT DO UPDATE`, que PostgreSQL décrit comme une insertion ou mise à jour atomique même sous forte concurrence, ou avec `MERGE`. Une limite : une instruction `ON CONFLICT DO UPDATE` refuse de modifier deux fois la même ligne et lève une violation de cardinalité. Il faut donc d'abord réduire les clés répétées du lot à une seule ligne par clé.

```
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;
```

Un chargement en masse s'arrête vers la fin et laisse le disque plus rempli qu'avant. `COPY FROM` écrit les lignes au fur et à mesure. S'il s'arrête sur une ligne incorrecte, les lignes déjà écrites restent dans un état supprimé : invisibles aux requêtes, elles occupent encore le disque jusqu'à ce que `VACUUM` récupère cet espace. Un échec tardif sur un gros fichier peut ainsi gaspiller beaucoup d'espace. PostgreSQL 17 a ajouté `ON_ERROR ignore`, qui saute les lignes texte ou CSV dont les valeurs ne peuvent être converties vers le type de leur colonne. PostgreSQL 18 a ajouté `REJECT_LIMIT` pour plafonner le nombre de lignes sautées. Cela ne couvre que la conversion de type ; une ligne violant une contrainte arrête

toujours la commande. Charger d'abord dans une table intermédiaire, y vérifier les données, puis transférer les bonnes lignes dans la vraie table en une transaction tient un mauvais fichier à l'écart des données lues par les utilisateurs.

La capture des changements remplit le disque. Le décodage logique lit les changements par un slot de réplication. Celui-ci conserve chaque portion du journal d'écriture anticipée encore nécessaire à son consommateur, que ce dernier soit connecté ou non. Si la lecture s'arrête, le journal s'accumule jusqu'à remplir le disque. Dans les cas extrêmes, le serveur s'arrête pour empêcher le bouclage des identifiants de transaction. La documentation PostgreSQL est explicite : un slot devenu inutile devrait être supprimé. Après un plantage, il peut aussi réémettre des changements récents ; le consommateur doit donc tolérer les doublons. C'est encore la même propriété de réexécution.

Le premier rapport du matin est lent après un gros chargement. Le planificateur choisit comment exécuter une requête à partir des statistiques de chaque table, et un chargement massif modifie les données qu'elles décrivent. Autovacuum peut les actualiser seul, mais pas toujours avant le premier rapport. Exécuter `ANALYZE` en dernière étape règle ce problème. Pour les très gros chargements dans une table existante, le guide PostgreSQL sur le remplissage d'une base ajoute qu'il peut être rentable de supprimer index et clés étrangères, puis de les recréer : les construire en masse est plus efficace que les mettre à jour ligne par ligne. Cela retire les contrôles d'erreur pendant leur absence ; réfléchissez donc à deux fois avant de supprimer un index unique.

Pour aller plus loin

- COPY — la commande de chargement en masse, ses formats et options, et ce qu'un chargement échoué laisse derrière lui.
- Remplir une base de données — les conseils PostgreSQL pour les gros chargements, de la transaction unique et de `COPY` à la reconstruction des index et à l'exécution finale d' `ANALYZE` .
- MERGE — insérer, mettre à jour et supprimer en une instruction, selon la correspondance de chaque ligne source avec la cible.
- INSERT — la clause `ON CONFLICT` , son upsert atomique et l'interdiction de toucher deux fois une ligne.
- Concepts du décodage logique — comment les changements sont lus dans le journal d'écriture anticipée et pourquoi supprimer un slot de réplication inutilisé.

Glossaire

Choisissez selon vos contraintes réelles

Présentez un jeu de données représentatif, les questions auxquelles il doit répondre et le goulot d'étranglement actuel. Nous pouvons convenir de la manière de mesurer la qualité, le débit et la reprise.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que l'ETL ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/etl/