Engineer

ETL

🔄 перенесення даних між системами зі зміною їхньої форми

Що це таке

ETL — скорочення від extract, transform, load, тобто «видобути, перетворити, завантажити», — це робота, у якій дані копіюють із систем, що їх створюють, надають їм форми й значення, яких очікує інша система, і записують туди, найчастіше у сховище даних. Близький варіант, ELT, зберігає ті самі три кроки, але спершу завантажує сирі дані, а перетворює їх уже всередині цільової системи, тому часто пишуть ETL/ELT, маючи на увазі обидва.

Кожна літера — окрема робота. Видобування читає джерело: цілу таблицю, файл експорту, API або лише рядки, змінені після попереднього запуску, які знаходять за часовою міткою `updated_at` чи через change data capture (CDC), тобто читання власного журналу змін бази даних. Перетворення — це місце, де дані очищують і узгоджують: текст стає датами й числами, одиниці виміру й часові пояси зводять до одного вигляду, дублікати прибирають, коди звіряють із довідковими таблицями, а результат переформовують у модель цільової системи, наприклад у таблиці фактів і вимірів. Завантаження записує результат — або масовою вставкою в порожню таблицю, або злиттям, яке оновлює наявні рядки й додає нові.

Порядок двох останніх кроків визначає, де виконується робота. Класичний ETL перетворює дані на окремому сервері ще до того, як вони потрапляють у цільову систему, і це мало сенс, коли місце й обчислення у сховищі даних були дефіцитними й дорогими. ELT складає сирі дані в проміжні (staging) таблиці всередині цільової системи й виконує перетворення там, засобами SQL. Сира копія залишається, тож коли правило змінюється, перетворення можна запустити знову по всій історії без повторного видобування. Ціна в тому, що все надіслане джерелом, зокрема стовпці, яких ніхто не просив, тепер лежить у сховищі даних, і цим треба там керувати.

PostgreSQL має інструмент для кожного кроку. `COPY` переносить рядки між таблицею й файлом у текстовому, CSV- чи двійковому форматі й залишається найшвидшим способом завантаження. Щоб прочитати файл на сервері, потрібні права суперкористувача або роль `pg_read_server_files`, а команда psql `\copy` читає файл на боці клієнта. `INSERT ... ON CONFLICT` і `MERGE` вливають нові рядки в наявні. Логічне декодування перетворює журнал попереднього запису (WAL) на потік змін рядків для CDC, а процедура PL/pgSQL дає змогу тримати перетворення поруч із даними, які воно змінює.

Застосуйте поняття до рішення

Застосуйте «Що таке ETL?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Цифри з'являються там, де потрібні, за розкладом, і ніхто не копіює таблицю наприкінці місяця.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Один інфраструктурний план для 14 підрозділів — 14 дослідницьких підрозділів, охоплених спільним інфраструктурним планом

Один інфраструктурний план для 14 підрозділів Зв'язатися з нами →

Чому це важливо

Більшість запитань, які варто ставити бізнесу, потребують даних із кількох систем. Замовлення лежать в одній базі даних, платежі — в експорті провайдера, клієнти — у CRM. Поки щось не зведе їх в одне місце за одним набором правил, кожну відповідь складають вручну, і вона залежить від того, хто складає. ETL і є тим, що їх зводить, а помилка в завантаженні переходить у кожен звіт, побудований на ньому.

Перетворення — це ще й місце, де бізнес записує, що означають його дані. Які рядки вважати продажем, до якого дня належить платіж опівночі, який запис перемагає, коли два клієнти збігаються: вирішене один раз у завантаженні, це правило діє для кожного звіту далі. Якщо залишити це кожному звітові, відповіді розходяться, і два дашборди показують дві суми.

Спосіб видобування визначає, як завантаження зростає. Повне перезавантаження щоразу копіює все. Воно просте й виправляє себе саме, бо вчорашню помилку сьогодні вночі буде перезаписано, але його вартість зростає разом із загальним обсягом даних. Інкрементне завантаження переносить лише змінене, тож його вартість зростає з темпом змін. Воно мусить пам’ятати, де зупинився попередній запуск, помічати рядки, видалені

в джерелі, і приймати рядки, що надходять пізно зі старими часовими мітками. Більшість систем починають із повних перезавантажень і переходять на інкрементні таблиця за таблицею, коли ночі перестає вистачати.

Постійна ціна полягає в тому, що завантаження — це код, який працює без нагляду, зазвичай уночі, проти джерел, що змінюються без попередження. Йому потрібне журналювання, яке показує, що виконалося і скільки рядків перенесено, сповіщення, коли запуск зупиняється, і понад усе одна властивість: повторний запуск із тими самими вхідними даними має залишати цільову систему точно такою, якою її залишив би один запуск. Завантаження з такою властивістю можна перезапустити після будь-якого збою без попередньої перевірки.

Як це виглядає на практиці

Фрагментовані конвеєри затримували критично важливу для бізнесу звітність у кількох середовищах розробки та виробничої експлуатації PostgreSQL, і для їх заміни написали 600 конвеєрів ETL/ELT на PL/pgSQL: ETL у масштабі сотень регулярних завантажень. Обмеження первинного та зовнішнього ключів захищали референційну цілісність під час трансформацій, унікальні й композитні індекси підтримували з’єднання, а обробка винятків із транзакційним контролем відкочувала запуск у разі збою. Інкрементальні завантаження використовували CDC і дельти на основі часових міток, а журналювання виконання й аудит допомагали відстежувати роботу та налагоджувати її. Результат роботи охоплює логіку трансформацій, обмеження, інкрементальне завантаження та журналювання виконання; запис не підтверджує результатів порівняльного бенчмарку чи розрахунку частоти помилок.

Переїзд даних на нову систему — це ETL, що виконується один раз. Перенесення картографічного застосунку з Microsoft SQL Server на PostgreSQL із PostGIS означало провести понад 50 мільйонів просторових записів через ETL-конвеєри, кроки трансформації яких переводили кожну координату в систему відліку EPSG:4326. Самі конвеєри були лише частиною переїзду: разом з індексами GiST і SP-GiST та налаштованою конфігурацією на новому боці він дав просторовим запитам середній час відповіді менш ніж 250 мілісекунд замість 2,5 секунди, тобто у 10 разів швидше, а навантаження на CPU бекенду впало на 65%.

Через межу мережі шість конвеєрів ETL/ELT показують у роботі кожну літеру назви. Видобування складалося із запланованого отримання стиснутих архівів із сервера, доступного лише через IPsec VPN: скрипти на Bash і Python класифікували файли Parquet, CSV і .bak за типом і схемою, а резервні копії .bak відновлювали в локальному екземплярі MSSQL Server. Дані завантажували в проміжні схеми в MSSQL і PostgreSQL за допомогою bcp, psql або SSIS залежно від формату, очищали, нормалізували й збагачували процедурами PL/pgSQL і T-SQL, потім стискали gzip і через бакет Google Cloud Storage переносили в Google BigQuery. Те, що вимагало щонайменше трьох годин ручної роботи, стало займати менш ніж 20 хвилин від початку до кінця, а синхронізація стала щоденною замість щотижневої.

Де виникають проблеми

Повторний запуск завантажує ті самі рядки двічі. Завдання вставляє дані звичайним `INSERT`, зупиняється на півдорозі через мережеву помилку, і планувальник завдань запускає його знову з початку. Рядки, що потрапили в таблицю першого разу, тепер лежать там двічі, і кожна сума, побудована на них, завищена. Дайте цільовій таблиці ключ, що визначає рядок у термінах бізнесу, і вливайте пакет через `INSERT ... ON CONFLICT DO UPDATE`, який документація PostgreSQL описує як атомарну вставку або оновлення навіть за високої конкурентності, або через `MERGE`. Є одна пастка: окрема інструкція `ON CONFLICT DO UPDATE` відмовляється змінювати той самий рядок двічі й повертає помилку cardinality violation, тож пакет із повторюваними ключами спершу треба звести до одного рядка на ключ.

```
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;
```

Масове завантаження зупиняється майже наприкінці й залишає диск заповненішим, ніж до нього. `COPY FROM` записує рядки в таблицю по ходу роботи. Коли команда зупиняється на поганому рядку, уже записані рядки залишаються у видаленому стані: запити їх не бачать, але вони займають диск, доки їх не прибере `VACUUM`, і зупинка ближче до кінця великого файлу може коштувати багато місця. PostgreSQL 17 додав `ON_ERROR ignore`, що пропускає рядки текстового чи CSV-вводу, значення яких не перетворюються на тип свого стовпця, а PostgreSQL 18 додав `REJECT_LIMIT`,

що обмежує, скільки рядків можна пропустити. Це стосується лише перетворення типів: рядок, що порушує обмеження, однаково зупиняє команду. Завантажуйте спершу в проміжну таблицю, перевіряйте дані там і переносьте добрі рядки в робочу таблицю однією транзакцією — так поганий файл не дістанеться до даних, які читають люди.

Change data capture заповнює диск. Логічне декодування читає зміни через слот реплікації, а слот утримує кожен шматок WAL, який ще потрібен його споживачеві, незалежно від того, чи споживач підключений. Коли читач зупиняється, журнал накопичується, доки диск не заповниться, а в крайніх випадках сервер вимикається, щоб запобігти переповненню лічильника транзакцій (transaction ID wraparound). Документація PostgreSQL каже про це прямо: слот, який більше не потрібен, слід видалити. Після збою слот також може надіслати останні зміни ще раз, тож споживач мусить витримувати дублікати, а це та сама властивість безпечного повторного запуску.

Перший ранковий звіт повільний після великого завантаження. Планувальник запитів вирішує, як виконати запит, на основі статистики кожної таблиці, а масове завантаження змінює дані під цією статистикою. Autovacuum може оновити її сам, але не завжди до того, як запуститься перший звіт. Запуск `ANALYZE` як останнього кроку завантаження це виправляє. Для дуже великих завантажень у наявну таблицю посібник PostgreSQL із наповнення бази даних додає, що видалити індекси й зовнішні ключі, а потім створити їх знову може бути вигідно, бо побудувати їх разом швидше, ніж оновлювати рядок за рядком. Поки їх немає, зникає й перевірка помилок, тож перед видаленням унікального індексу варто подумати двічі.

Що почитати далі

- COPY — команда масового завантаження, її формати й параметри і те, що залишає по собі перерване завантаження.
- Populating a Database — поради PostgreSQL щодо великих завантажень: від однієї транзакції та `COPY` до перебудови індексів і запуску `ANALYZE` наприкінці.
- MERGE — вставка, оновлення й видалення однією інструкцією залежно від того, як кожен рядок джерела збігається з цільовою таблицею.
- INSERT — секція `ON CONFLICT`, її атомарний upsert і заборона змінювати той самий рядок двічі.
- Logical Decoding Concepts — як зміни читаються з WAL і чому непотрібний слот реплікації слід видалити.

Словник

Обирайте за фактичними умовами

Надайте репрезентативний набір даних, запитання до нього та поточне вузьке місце. Узгодимо вимірювання якості, пропускної здатності та відновлення.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке ETL?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/etl/