Engineer

Change data capture

зчитування змін у базі даних для передавання іншим системам

Що це таке

Change data capture, зазвичай скорочено CDC, — це зчитування змін у базі даних для передавання іншим системам у міру того, як ці зміни відбуваються. Українською цей підхід також називають захопленням змін даних. Кожна вставка, оновлення й видалення стає подією, на яку може відреагувати щось інше: копія для звітності, що залишається актуальною, пошуковий індекс, інший сервіс, якому треба знати, що замовлення оформлено. База даних, де відбувається зміна, лишається головним джерелом даних, а решта систем іде за нею.

Знайти зміни можна двома основними способами. Простіший — питати: кожні кілька хвилин виконувати запит, який шукає рядки з часовою міткою «останнього оновлення», новішою за попередній запуск. Інший — читати власний запис бази даних про те, що вона змінила. PostgreSQL веде такий запис у журналі попереднього запису (WAL), і документація формулює правило, на якому він тримається: зміни у файлах даних записують лише тоді, коли записи журналу, що їх описують, уже збережено на постійному носії. Оскільки журнал і так містить кожну зміну в тому порядку, у якому вона сталася, його читання дає повну історію без жодного звернення до таблиць. Саме другий підхід, CDC на основі журналу, сьогодні зазвичай і мають на увазі під цим терміном.

У PostgreSQL цей механізм називається логічним декодуванням. Документація описує його як процес, що витягає всі збережені зміни в таблицях бази даних у зв’язний формат, який можна тлумачити без детального знання внутрішнього стану бази. Разом працюють три частини. Сервер має записувати в WAL достатньо подробиць, і це визначає параметр `wal_level` : його типове значення, `replica` , достатнє для копіювання даних між серверами, а `logical` додає те, що потрібно для декодування. Слот реплікації представляє один потік змін з однієї бази даних, які відтворюються

для клієнта в тому порядку, в якому їх було зроблено, і пам’ятає, докуди клієнт уже дочитав. Плагін виводу (output plugin) перетворює внутрішні записи WAL на формат, потрібний клієнтові.

Більшість команд не пише клієнта самостійно. Debezium описує себе як набір конекторів-джерел для Apache Kafka Connect, кожен з яких читає зміни з певної бази даних через її власні механізми CDC. Конектор для PostgreSQL читає через `pgoutput`, який документація Debezium називає вбудованим плагіном виводу PostgreSQL, або через `decoderbufs`, плагін, який підтримує спільнота Debezium. Debezium також може працювати як бібліотека всередині JVM-застосунку або як Debezium Server, що надсилає події змін до таких систем, як Amazon Kinesis, Google Cloud Pub/Sub і Apache Pulsar.

Застосуйте поняття до рішення

Застосуйте «Що таке change data capture?» 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

Опитування здається дешевшим, і для невеликої таблиці, що змінюється рідко, часто так і є. Документація Debezium перелічує, що вміє читання журналу і чого не вміє опитування. Воно фіксує кожну зміну, тоді як запит бачить лише стан рядка на момент виконання, тож рядок, який тричі змінився між двома запусками, з’явиться один раз. Воно фіксує видалення, тоді як видаленого рядка просто немає в наступному результаті. Воно не потребує додаткового стовпця в моделі даних, як-от поля «Last Updated», яке кожен шлях запису має не забути заповнити. Воно може передати попередній стан рядка й метадані, наприклад ідентифікатор транзакції, що внесла зміну. І воно доставляє зміни з дуже малою затримкою без навантаження на процесор, яке дає часте опитування: для PostgreSQL і MySQL Debezium оцінює цю затримку в діапазоні мілісекунд.

Друга причина — узгодженість між базою даних і повідомленнями, які отримують інші системи. Debezium називає подвійний запис (dual writes) іншим підходом, з яким себе порівнює: застосунок зберігає зміну у своїй базі даних, а потім надсилає про неї повідомлення. Якщо процес зупиниться між цими двома кроками, база даних і потік повідомлень розійдуться, і ніщо не зафіксує, хто з них помиляється. CDC прибирає другий запис. Застосунок фіксує зміну один раз, а повідомлення виводиться з того, що справді було зафіксовано. Патерн outbox іде ще далі: застосунок записує саме повідомлення в таблицю outbox у тій самій транзакції, що й зміну, а CDC публікує рядки цієї таблиці. Outbox Event Router від Debezium описує мету як уникнення розбіжностей між внутрішнім станом сервісу та подіями, які споживають інші сервіси.

Третя причина — порядок. Слот реплікації віддає зміни в тому порядку, у якому їх було зроблено на сервері-джерелі, згрупованими за транзакцією, що їх зробила, як показує приклад нижче. Споживач, який застосовує їх у цьому порядку, крок за кроком відтворює той самий стан. Для ETL-завантаження це робить інкрементальне завантаження чесним: нічне завдання може взяти саме ті зміни, що надійшли після попереднього запуску, разом із видаленнями, замість перезавантажувати цілі таблиці чи покладатися на стовпець часової мітки, який одна з гілок коду забула оновити.

Як це виглядає на практиці

Найменший робочий приклад використовує лише PostgreSQL. Йому потрібні `wal_level = logical` у конфігурації сервера, що набирає чинності лише після перезапуску, і вільний слот реплікації; `max_replication_slots` за замовчуванням дорівнює 10. Сесія нижче працює від імені суперкористувача, як і власний приклад документації, і використовує `test_decoding` — плагін виводу з того прикладу, який друкує кожну зміну окремим рядком тексту.

```
-- postgresql.conf: wal_level = logical (діє лише після перезапуску)
SELECT * FROM pg_create_logical_replication_slot('shop_cdc', 'test_decoding');

BEGIN;
INSERT INTO orders (id, status) VALUES (42, 'pending');
INSERT INTO outbox (id, event) VALUES (7, 'нове замовлення 42');
COMMIT;

SELECT data FROM pg_logical_slot_peek_changes('shop_cdc', NULL, NULL);
```

```
-- BEGIN 10301
-- table public.orders: INSERT: id[integer]:42 status[text]:'pending'
-- table public.outbox: INSERT: id[integer]:7 event[text]:'нове замовлення 42'
-- COMMIT 10301

SELECT data FROM pg_logical_slot_get_changes('shop_cdc', NULL, NULL);
-- ті самі чотири рядки; виклик їх споживає, тож повторний нічого не поверне

SELECT pg_drop_replication_slot('shop_cdc');
```

Замовлення та його повідомлення в outbox виходять разом, у межах транзакції, яка їх записала. `pg_logical_slot_peek_changes` читає, не зсуваючи слот, тому наступний виклик поверне ті самі рядки, а `pg_logical_slot_get_changes` їх споживає. `CREATE TABLE` дав би лише порожні `BEGIN` і `COMMIT`, бо логічне декодування не передає DDL. Останній рядок найважливіший: забутий слот і далі утримує ресурси сервера.

У продакшні читачем стає Debezium. Під час першого запуску його конектор для PostgreSQL робить узгоджений знімок (snapshot) таблиць, зміни яких відстежує, а потім передає потоком зміни від тієї точки, де знімок було зроблено. Типове значення `snapshot.mode`, `initial`, робить знімок лише тоді, коли збережених офсетів ще немає, тож після перезапуску конектор продовжує з останньої зафіксованої позиції. Кожна подія містить образи рядка `before` і `after`, блок `source`, часову мітку й код `op`: `c` — створення, `u` — оновлення, `d` — видалення, `r` — рядок, прочитаний під час знімка, і `t` — truncate. Після видалення конектор також надсилає tombstone — подію з тим самим ключем і значенням null, щоб log compaction у Kafka міг прибрати попередні повідомлення з цим ключем.

Обидва патерни з цієї статті трапляються на власних сторінках компанії. У 2022–2024 роках засновниця розробила 600 пайплайнів ETL/ELT на PL/pgSQL у кількох середовищах розробки та виробничої експлуатації PostgreSQL, зокрема з інкрементальним завантаженням за допомогою CDC і дельт на основі часових міток. Нотатка Продати останній квиток рівно один раз будує флеш-розпродаж навколо outbox: кожен крок записує своє повідомлення в outbox у тій самій транзакції, що й зміну, change data capture публікує outbox, а оскільки повідомлення може надійти більше ніж один раз, кожен споживач ідемпотентний, з ідентифікатором замовлення як ключем.

Де виникають проблеми

Ніщо не обмежує, скільки може утримувати слот. Слот тримає кожен шматок WAL, який його споживач ще не підтвердив, і типові налаштування PostgreSQL не ставлять цьому жодної межі. `max_slot_wal_keep_size` дорівнює `-1`, тобто слоти можуть утримувати необмежений обсяг WAL, а `idle_replication_slot_timeout` дорівнює `0`, що вимикає анулювання слотів, якими ніхто не користується. Тож конектор, який зупинився під час невдалого розгортання, може заповнити диск через довгий час після того, як про нього всі забули. Стежте за `pg_replication_slots`, щоб бачити неактивні слоти, і встановіть `max_slot_wal_keep_size` на обсяг, який диск може витримати, пам'ятаючи, що споживач, який відстане сильніше, може вже не змогти продовжити. Відновлення означає новий слот, а новий слот починає з поточного моменту: Debezium попереджає, що видалений і створений заново слот може непомітно пропустити події між ними, тож після цього зробіть свіжий знімок.

Тиха таблиця тримає журнал. Документація Debezium описує підступніший варіант. Слот належить одній базі даних, і конектор просуває свою підтверджену позицію лише тоді, коли бачить зміни, які відстежує. Якщо відстежувані таблиці змінюються рідко, а інші таблиці чи інші бази даних на тому самому сервері записують багато, позиція майже не рухається, і WAL за нею продовжує рости. Рішення — heartbeat: `heartbeat.interval.ms` змушує конектор надсилати heartbeat-події із заданим інтервалом, а `heartbeat.action.query` дає змогу регулярно записувати в невелику таблицю у відстежуваній базі даних. Ця таблиця має входити до публікації (publication), яку читає конектор, тож слот просувається, навіть коли більше нічого не відбувається.

Споживачі вважають, що кожна зміна приходить один раз. Логічний слот за звичайної роботи надсилає кожну зміну один раз, але його позиція зберігається лише на контрольних точках (checkpoints), тож після збою він може повернутися до ранішої точки й надіслати останні зміни ще раз. Документація PostgreSQL залишає наслідки на відповідальність клієнта. Споживач, що додає кожну суму до накопиченого підсумку, порахує ці зміни двічі. Зробіть кожного споживача ідемпотентним: застосовуйте кожну зміну як upsert за первинним ключем рядка або записуйте ідентифікатори вже оброблених подій, щоб повторна доставка нічого не змінювала.

Оновлення та видалення приходять без старих значень. Що подія оновлення чи видалення каже про попередній рядок, залежить від replica identity таблиці. З `REPLICA IDENTITY DEFAULT` події Debezium містять щонайбільше попередні значення первинного ключа, а для таблиці без первинного ключа конектор узагалі не надсилає подій оновлення чи видалення. `REPLICA IDENTITY FULL` кладе в `before` усі попередні стовпці, але має ціну: документація PostgreSQL зазначає, що це збільшує обсяг WAL, особливо коли виконується багато операторів `UPDATE` і `DELETE`. Дайте кожній відстежуваній таблиці первинний ключ і перемикайтеся на `FULL` лише для тих таблиць, чиїм споживачам потрібні старі значення.

Що почитати далі

- Debezium Features — що CDC на основі журналу фіксує з того, що пропускають опитування й подвійний запис, а також знімки, фільтри й маскування.
- Debezium connector for PostgreSQL — плагіни виводу, знімки, формат подій змін, replica identity і місце на диску для WAL.
- Logical Decoding Concepts — слоти реплікації, плагіни виводу і чому слот, який нікому не потрібен, слід видалити.
- Logical Decoding Examples — створення слота, перегляд і споживання змін та видалення слота.
- Replication (server configuration) — `max_replication_slots`, `max_slot_wal_keep_size` і `idle_replication_slot_timeout`.
- Outbox Event Router — патерн outbox і те, як Debezium спрямовує рядки outbox у топіки.

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 change data capture?» 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/change-data-capture/