MERGE i UPSERT: jak to robi każdy silnik

> Upsert wstawia nowy wiersz albo aktualizuje istniejący, w zależności od tego, czy taki wiersz już jest w tabeli, a każdy z czterech silników tej książki oferuje na to własną, niekompatybilną składnię — ta strona przechodzi przez każdą z nich osobno.

Skróty użyte na tej stronie (2)

Skrót	Co znaczy
ISO	International Organization for Standardization: organizacja, która wydaje normy techniczne używane na całym świecie
SQL	Structured Query Language: język zapytań do relacyjnych baz danych

Fiszka

ZMIENIA SYSTEM

Standardowy MERGE (Oracle, MSSQL, PostgreSQL od 15)

```
MERGE INTO klienci AS cel
USING (VALUES ('Kowalski Handel', 'Kraków', '6762345678')) AS zrodlo (nazwa, miasto,
nip)
    ON cel.nip = zrodlo.nip
WHEN MATCHED THEN
    UPDATE SET miasto = zrodlo.miasto
WHEN NOT MATCHED THEN
    INSERT (nazwa, miasto, nip) VALUES (zrodlo.nazwa, zrodlo.miasto, zrodlo.nip);
```

`MERGE` zestawia tabelę docelową ze źródłem danych po warunku podanym w `ON` — tu po numerze NIP, jako kluczu naturalnym — i dla każdego dopasowania wykonuje jedną z gałęzi:

`WHEN MATCHED` aktualizuje istniejący wiersz, `WHEN NOT MATCHED` wstawia nowy. Cała logika mieści się w jednym poleceniu, więc silnik widzi ją jako jedną, atomową operację, zamiast dwóch osobnych poleceń wykonywanych jedno po drugim z osobna.

Przykład:

```
MERGE 1
```

`Kowalski Handel` już istnieje z tym samym NIP-em, więc trafia w gałąź `WHEN MATCHED` — jego `miasto` zostaje zaktualizowane, jeśli się zmieniło, a `nazwa` i `nip` pozostają bez zmian.

MSSQL = Microsoft SQL Server: silnik bazy danych Microsoftu, na nim w Zielonej stoi księgowość Optima · **SET** = Switch Embedded Teaming: zespół kart sieciowych zbudowany w przełączniku wirtualnym Hyper-V; łączy dwie karty w jedno łącze bez ustawień po stronie przełącznika fizycznego, więc bez LACP

ZMIENIA SYSTEM

INSERT ... ON CONFLICT DO UPDATE (PostgreSQL)

```
INSERT INTO klienci (nazwa, miasto, nip)
VALUES ('Kowalski Handel', 'Kraków', '6762345678')
ON CONFLICT (nip) DO UPDATE SET miasto = EXCLUDED.miasto;
```

PostgreSQL przed wersją 15 nie miał `MERGE` w ogóle i do dziś ten zapis jest w PostgreSQL bardziej rozpowszechniony — próbuje wstawić nowy wiersz, a gdy natrafi na konflikt z ograniczeniem unikalności na kolumnie wskazanej w `ON CONFLICT`, wykonuje zamiast tego `UPDATE`. `EXCLUDED` to pseudo-tabela reprezentująca wiersz, który silnik próbował wstawić — odpowiednik `zrodlo` z `MERGE`, tylko nazwany inaczej.

Przykład:

```
INSERT 0 1
```

Ta sama liczba `INSERT 0 1` pojawia się niezależnie od tego, czy polecenie w praktyce wstawiło nowy wiersz, czy zaktualizowało istniejący — `INSERT ... ON CONFLICT` zawsze raportuje się jak `INSERT`, nawet gdy realnie wykonał aktualizację.

SET = Switch Embedded Teaming: zespół kart sieciowych zbudowany w przełączniku wirtualnym Hyper-V; łączy dwie karty w jedno łącze bez ustawień po stronie przełącznika fizycznego, więc bez LACP

ZMIENIA SYSTEM

INSERT ... ON DUPLICATE KEY UPDATE (MySQL)

```
INSERT INTO klienci (nazwa, miasto, nip)
VALUES ('Kowalski Handel', 'Kraków', '6762345678')
ON DUPLICATE KEY UPDATE miasto = VALUES(miasto);
```

MySQL rozwiązuje ten sam problem własną składnią — `VALUES(miasto)` odwołuje się do wartości, którą to samo polecenie próbowało wstawić, analogicznie do `EXCLUDED` w PostgreSQL. Warunek konfliktu nie jest tu jawny — silnik sam sprawdza każde ograniczenie `UNIQUE` albo `PRIMARY KEY` tabeli i odpala gałąź `UPDATE`, gdy którekolwiek z nich zostanie naruszone.

Przykład:

```
Query OK, 2 rows affected (0.01 sec)
```

MySQL liczy aktualizację istniejącego wiersza jako dwa dotknięte wiersze, nie jeden — to udokumentowana osobliwość tego polecenia w tym silniku, nie błąd.

□□ TYLKO ODCZYT

REPLACE INTO (MySQL)

```
REPLACE INTO klienci (id, nazwa, miasto, nip) VALUES (2, 'Kowalski Handel', 'Kraków',
'6762345678');
```

`REPLACE` to starsza, prostsza alternatywa dostępna tylko w MySQL i MariaDB — zamiast aktualizować pasujący wiersz, usuwa go w całości i wstawia nowy w jego miejsce. Różnica ma znaczenie praktyczne: każda kolumna, której nie wymieniono w `REPLACE`, wraca do wartości domyślnej zamiast zachować to, co miała wcześniej, a wyzwalacze `DELETE` i `INSERT` odpalają się osobno zamiast jednego wyzwalacza `UPDATE`.

Przykład:

```
Query OK, 2 rows affected (0.01 sec)
```

Po co to istnieje

> **Pojęcie: upsert.** Operacja, która wstawia nowy wiersz, jeśli pasujący jeszcze nie istnieje, albo aktualizuje istniejący, jeśli już jest — nazwa to zlepek słów „update" i „insert". Naiwna implementacja tej logiki po stronie aplikacji — najpierw `SELECT`, sprawdzenie czy wiersz istnieje, potem warunkowo `INSERT` albo `UPDATE` — wygląda poprawnie w izolacji, ale staje się wyścigiem przy współbieżności: dwie sesje mogą wykonać `SELECT` jednocześnie, obie zobaczyć brak wiersza, i obie spróbować `INSERT`, z których jedna nieuchronnie naruszy ograniczenie unikalności, jeśli takie istnieje, albo — gorzej — obie się powiodą, jeśli ograniczenia nie ma, zostawiając dwa duplikaty tam, gdzie miał być jeden wiersz.

Wszystkie konstrukcje z tej strony rozwiązują ten wyścig, bo sprawdzenie i decyzja (wstawić czy zaktualizować) dzieją się w jednej, atomowej operacji silnika, a nie w dwóch osobnych poleceniach z oknem czasowym między nimi, w którym inna sesja mogłaby zdążyć wstawić ten sam wiersz.

Jak to działa

Rozwiń: jak to działa pod spodem

Wymóg indeksu unikalnego. Żadna z tych konstrukcji nie zadziała bez ograniczenia `UNIQUE` albo `PRIMARY KEY` na kolumnie, po której rozstrzygamy dopasowanie — `ON CONFLICT (nip)` wymaga indeksu unikalnego na `nip`, `ON DUPLICATE KEY` wymaga dowolnego klucza unikalnego naruszanego przez nowy wiersz, a `MERGE ... ON cel.nip = zrodlo.nip` bez unikalności na `nip` po prostu dopasuje więcej niż jeden wiersz i da nieprzewidywalny wynik zamiast błędu. To ograniczenie jest mechanizmem, dzięki któremu silnik w ogóle potrafi rozstrzygnąć, czy dany wiersz „już istnieje" — bez unikalności odpowiedź na to pytanie mogłaby brzmieć „i tak, i nie" jednocześnie.

```
ERROR:  there is no unique or exclusion constraint matching the ON CONFLICT specification
```

Ten błąd pojawia się dokładnie wtedy, gdy kolumna wskazana w `ON CONFLICT` nie ma pasującego ograniczenia unikalności — silnik odmawia zgadywania, którego z wielu potencjalnie pasujących wierszy dotyczyłaby aktualizacja.

Znane pułapki MERGE w MSSQL. Standardowy `MERGE` w SQL Server ma udokumentowaną historię błędów przy współbieżnym wykonaniu — dwie sesje wykonujące ten sam `MERGE` na tych samych danych jednocześnie potrafią, w zależności od poziomu izolacji i wersji silnika, obie przejść przez gałąź `WHEN NOT MATCHED` i obie spróbować wstawić ten sam wiersz, co kończy się naruszeniem klucza zamiast oczekiwanej aktualizacji. Zalecenie Microsoftu w takich scenariuszach to opakowanie `MERGE` w `TRANSACTION` z jawnym poziomem izolacji `SERIALIZABLE` albo blokadą tabeli (`WITH (HOLDLOCK)`) na czas operacji, a nie poleganie na samym `MERGE` jako gwarancji atomowości pod obciążeniem. `INSERT ... ON CONFLICT` w PostgreSQL nie ma tej klasy

problemów — jest zaprojektowany od podstaw jako operacja atomowa na poziomie pojedynczego wiersza.

Częste błędy

MERGE bez ON MATCHED / NOT MATCHED w obu kierunkach.

Pominięcie gałęzi `WHEN NOT MATCHED` w `MERGE` sprawia, że polecenie ciche pomija wiersze źródłowe bez pasującego odpowiednika w tabeli docelowej, zamiast je wstawić — bez żadnego błędu, bo brak tej gałęzi jest składniowo poprawny, po prostu nic dla takich wierszy nie robi.

ON CONFLICT DO UPDATE bez wskazania kolumny konfliktu.

```
ERROR:  ON CONFLICT DO UPDATE requires inference specification or constraint name
```

`ON CONFLICT DO UPDATE` musi wiedzieć, które ograniczenie brać pod uwagę — samo `ON CONFLICT DO UPDATE SET ...` bez `(nip)` albo nazwy ograniczenia po `ON CONFLICT` nie wystarcza, gdy tabela ma więcej niż jedno ograniczenie unikalności.

Nadpisanie kolumn niewymienionych w REPLACE INTO.

```
REPLACE INTO klienci (id, nazwa) VALUES (2, 'Kowalski Handel - nowa nazwa');
```

Kolumny `miasto` i `nip` nie zostały wymienione, więc `REPLACE` — usuwając stary wiersz i wstawiając nowy — zapisuje w nich wartości domyślne albo `NULL`, zamiast zachować to, co było wcześniej. `INSERT ... ON DUPLICATE KEY UPDATE` tego problemu nie ma, bo aktualizuje tylko kolumny wprost wymienione w klauzuli `UPDATE`.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — wyścig bez upsertu

Dwie sesje aplikacji jednocześnie próbują zarejestrować tego samego klienta po raz pierwszy, każda wykonując najpierw `SELECT id FROM klienci WHERE nip = '6641234567'`, a dopiero po otrzymaniu pustego wyniku — `INSERT`. Tabela `klienci` ma ograniczenie `UNIQUE` na `nip`. Co się

stanie, jeśli obie sesje wykonają `SELECT` niemal jednocześnie, zanim którakolwiek zdąży wstawić swój wiersz?

Odpowiedź

Obie sesje zobaczą pusty wynik `SELECT`a, bo w chwili jego wykonania żadna z nich jeszcze nic nie wstawiła, i obie przejdą do `INSERT`. Pierwsza z nich wstawi wiersz bez przeszkód, druga naruszy ograniczenie `UNIQUE` na `nip` i dostanie błąd — dokładnie ten sam typ błędu co przy powtórnym `INSERT` na stronie INSERT: jeden wiersz, wiele wierszy, INSERT ... SELECT. To jest wyścig opisany w sekcji „Po co to istnieje": naiwne sprawdzenie `SELECT`em nie daje żadnej gwarancji, że stan się nie zmieni między odczytem a zapisem.

Pytania kontrolne

1. Dlaczego `INSERT ... ON CONFLICT DO UPDATE` wymaga istnienia indeksu unikalnego na kolumnie podanej w `ON CONFLICT`, a nie działa na dowolnej kolumnie?

Odpowiedź

Bez ograniczenia unikalności silnik nie ma sposobu, żeby jednoznacznie rozstrzygnąć, czy nowy wiersz „koliduje" z którymś istniejącym — bez unikalności więcej niż jeden istniejący wiersz mógłby teoretycznie mieć tę samą wartość, więc pytanie „który z nich zaktualizować" nie miałoby jednoznacznej odpowiedzi.

2. Czym różni się `REPLACE INTO` od `INSERT ... ON DUPLICATE KEY UPDATE` w MySQL, poza długością zapisu?

Odpowiedź

`REPLACE` usuwa cały istniejący wiersz i wstawia nowy w jego miejsce, więc każda kolumna niewymieniona w poleceniu wraca do wartości domyślnej, a wyzwalacze `DELETE` i `INSERT` odpalają się osobno. `INSERT ... ON DUPLICATE KEY UPDATE` modyfikuje tylko kolumny jawnie wymienione w klauzuli `UPDATE`, zachowując resztę wiersza bez zmian, i odpala wyzwalacz `UPDATE`, nie parę `DELETE`+`INSERT`.

Powiązane: INSERT: jeden wiersz, wiele wierszy, INSERT ... SELECT · UPDATE: i dlaczego zawsze piszemy WHERE zanim napiszemy SET · Ograniczenia: NOT NULL, UNIQUE, CHECK, FOREIGN KEY, DEFAULT · Poziomy izolacji i co który naprawdę gwarantuje

Źródła: dokumentacja PostgreSQL, rozdział „INSERT ... ON CONFLICT” i „MERGE” · dokumentacja Microsoft, „MERGE (Transact-SQL)” · dokumentacja MySQL, „INSERT ... ON DUPLICATE KEY UPDATE Statement” i „REPLACE Statement” · dokumentacja Oracle, „MERGE” · ISO/IEC 9075-2:2016 § 14.10

Revision #47
Created 2026-09-24 22:42:15 UTC by Tomasz Konieczny
Updated 2026-10-26 14:11:18 UTC by Tomasz Konieczny