Replikacja: synchroniczna, asynchroniczna, logiczna, fizyczna

> Replika powtarza wiernie każdą zmianę, jaka trafia do bazy głównej — łącznie z błędem operatora, który właśnie skasował całą tabelę — i to jest powód, dla którego replikacja nigdy nie zastępuje backupu, choćby wyglądała na jego doskonalszą wersję.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
GTID	Global Transaction Identifier: unikalny w całym klastrze numer nadawany każdej transakcji w MySQL i MariaDB, ułatwia przełączenie repliki na nowe źródło bez ręcznego szukania pozycji
RPO	Recovery Point Objective: ile danych wolno stracić przy awarii
WAL	Write-Ahead Log: dziennik PostgreSQL, zapis zmian przed zapisem danych

Fiszka

TYLKO ODCZYT

Sprawdzenie opóźnienia repliki: PostgreSQL

```
SELECT now() - pg_last_xact_replay_timestamp() AS opoznienie;
```

Wykonujemy na serwerze repliki — funkcja zwraca znacznik czasu ostatniej odtworzonej na replice transakcji, a odjęcie go od bieżącego czasu daje bezpośrednio opóźnienie w postaci interwału.

Przykład:

```
produkcja=# SELECT now() - pg_last_xact_replay_timestamp() AS opoznienie;
 opoznienie
------------
 00:00:02.841341
(1 row)
```

□□ TYLKO ODCZYT

Sprawdzenie opóźnienia repliki: Oracle Data Guard

```
SELECT name, value FROM v$dataguard_stats WHERE name = 'apply lag';
```

Widok `v$dataguard_stats` na bazie zapasowej (standby) zawiera metrykę `apply lag` — czas, o jaki zastosowanie odebranego redo na standby pozostaje w tyle za jego wygenerowaniem na bazie głównej.

Przykład:

```
NAME                 VALUE
-------------------- --------------------------
apply lag            +00 00:00:04
```

□□ TYLKO ODCZYT

Sprawdzenie opóźnienia repliki: SQL Server Always On

```
SELECT database_name, synchronization_state_desc, log_send_queue_size, redo_queue_size
FROM sys.dm_hadr_database_replica_states;
```

`log_send_queue_size` i `redo_queue_size` w kilobajtach pokazują, ile danych czeka odpowiednio na wysłanie do repliki i na zastosowanie po jej stronie — rosnące wartości w kolejnych odczytach sygnalizują narastające opóźnienie.

Przykład:

```
database_name   synchronization_state_desc   log_send_queue_size   redo_queue_size
--------------- --------------------------- --------------------- ----------------
produkcja       SYNCHRONIZED                0                     0
```

□□ **SQL** = Structured Query Language: język zapytań do relacyjnych baz danych

TYLKO ODCZYT

Sprawdzenie opóźnienia repliki: MySQL / MariaDB

```
SHOW REPLICA STATUS\G
```

Pole `Seconds_Behind_Source` (w starszych wersjach `Seconds_Behind_Master`) podaje przybliżone opóźnienie repliki w sekundach względem serwera źródłowego.

Przykład:

```
*************************** 1. row ***************************
             Replica_IO_Running: Yes
            Replica_SQL_Running: Yes
       Seconds_Behind_Source: 1
```

show = FortiOS: wypisuje konfigurację, w podstawowej postaci tylko ustawienia różniące się od domyślnych, a z wariantem full-configuration także wartości domyślne

Po co to istnieje

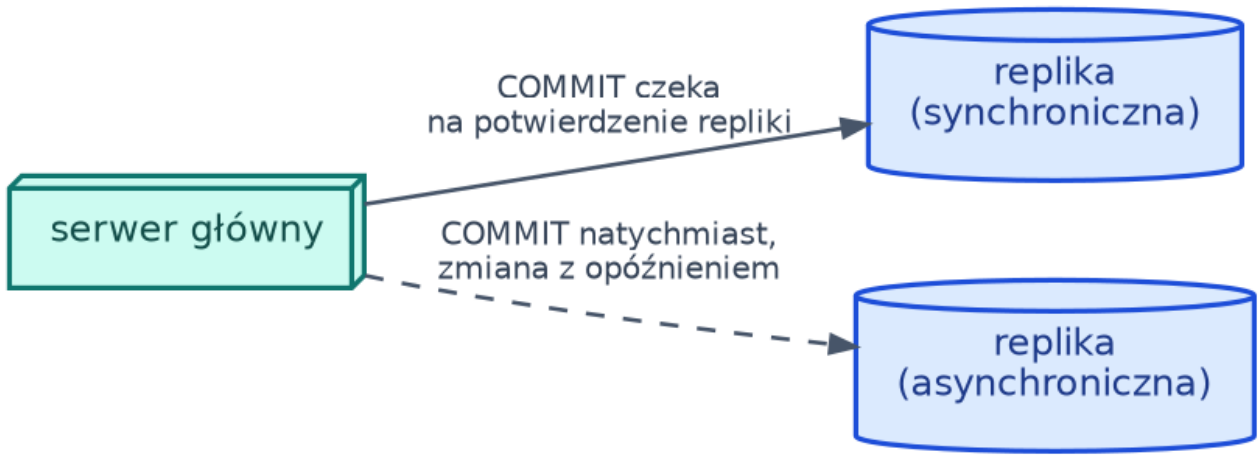
> **Pojęcie: replikacja.** Mechanizm utrzymujący co najmniej jedną dodatkową kopię bazy (replikę), na bieżąco odzwierciedlającą zmiany wykonywane na bazie głównej. Replikacja rozwiązuje dwa różne problemy naraz, w zależności od tego, po co ją wdrażamy: rozłożenie odczytów na kilka serwerów zamiast jednego, oraz gotowość do przejęcia ruchu przez replikę w razie awarii serwera głównego, znacznie szybciej niż pozwalałoby na to odtwarzanie z backupu.

Replikacja i backup wyglądają podobnie — obie utrzymują dodatkową kopię danych — ale odpowiadają na inne kategorie zagrożeń, i to jest różnica, którą trzeba rozumieć, zanim ktokolwiek zamieni jedno na drugie w planie awaryjnym. Backup, opisany na stronie Backup pełny, różnicowy, przyrostowy, logi, jest kopią z przeszłości, celowo opóźnioną względem bieżącego stanu — dzięki temu opóźnieniu backup przetrwa błąd, który właśnie wydarzył się na produkcji. Replika natomiast dąży do tego, żeby być jak najbardziej aktualna, z opóźnieniem liczonym w sekundach albo mniej — a to oznacza, że błędny `DELETE` bez `WHERE` wykonany na bazie głównej dotrze do repliki równie szybko jak każda poprawna zmiana i tam też skasuje te same wiersze. Replika chroni przed awarią sprzętu czy sieci, nie przed błędem logicznym w danych — z tego właśnie powodu żadna architektura oparta wyłącznie na replikacji, bez osobnego, opóźnionego backupu, nie jest kompletnym planem odzyskiwania.

Jak to działa

Rozwiń: jak to działa pod spodem

Synchroniczna kontra asynchroniczna. Pojęcie: replikacja synchroniczna. Tryb, w którym transakcja na bazie głównej zostaje potwierdzona klientowi (`COMMIT` się kończy) dopiero wtedy, gdy odpowiadająca jej zmiana zostanie potwierdzona także przez co najmniej jedną replikę. Gwarantuje to, że żadna zatwierdzona transakcja nie istnieje wyłącznie na serwerze głównym — dokładnie warunek potrzebny do osiągnięcia RPO bliskiego zeru, opisanego na stronie RPO i RTO: dwie liczby, od których zaczyna się każdy plan backupu. Cena tej gwarancji jest wprost widoczna w czasie odpowiedzi: każdy zapis czeka na podróż w obie strony do repliki, więc opóźnienie sieciowe między serwerami wlicza się bezpośrednio do czasu każdego `COMMIT`u — replika oddalona geograficznie o dużym opóźnieniu sieci potrafi spowolnić każdą transakcję zapisującą na tyle, że synchroniczność przestaje być praktyczna.



Przy replikacji synchronicznej COMMIT czeka na replikę, przy asynchronicznej zmiana dochodzi z opóźnieniem (kliknięcie otwiera pełny rozmiar)

> **Pojęcie: replikacja asynchroniczna.** Tryb, w którym transakcja na bazie głównej zostaje potwierdzona klientowi natychmiast po zapisaniu na serwerze głównym, a zmiana trafia do repliki z pewnym opóźnieniem, bez czekania na jej potwierdzenie. `COMMIT` jest szybki, niezależny od odległości do repliki, ale istnieje okno czasowe, w którym zatwierdzona transakcja istnieje wyłącznie na serwerze głównym — awaria

serwera głównego dokładnie w tym oknie oznacza utratę tych ostatnich, jeszcze nieprzesłanych zmian. Większość wdrożeń produkcyjnych, w których replika służy głównie do odczytów albo jako gotowość awaryjna z akceptowalnym niewielkim RPO, wybiera asynchroniczność właśnie ze względu na brak wpływu na czas odpowiedzi zapisów.

Fizyczna kontra logiczna. Pojęcie: replikacja fizyczna. Przenoszenie na replikę tych samych zmian na poziomie stron danych, jakie zapisuje log transakcyjny bazy głównej — replika odtwarza dosłownie te same operacje blokowe, które wydarzyły się na oryginale, w tej samej kolejności. Rezultatem jest replika będąca bajtową kopią całej instancji: te same bazy, te same tabele, ten sam układ na dysku, bez możliwości replikowania tylko wybranego fragmentu ani zmiany czegokolwiek po drodze.

Pojęcie: replikacja logiczna. Przenoszenie na replikę zdarzeń zmiany danych opisanych na poziomie logicznym — konkretny wiersz w konkretnej tabeli został wstawiony, zmieniony albo usunięty — niezależnie od tego, jak te zmiany są fizycznie zapisane na dysku. Replikacja logiczna pozwala na elastyczność niedostępną fizycznej: replikowanie tylko wybranych tabel zamiast całej bazy, replikację między różnymi wersjami silnika, a nawet między różnymi bazami tego samego klastra w różne strony naraz — kosztem większego narzutu na przetwarzanie każdej zmiany po stronie repliki.

Silnik	Mechanizm fizyczny	Mechanizm logiczny
PostgreSQL	streaming replication (WAL) + replication slots	logical replication (publikacja/subskrypcja)
Oracle	Data Guard (redo transport)	GoldenGate
SQL Server	Always On Availability Groups, log shipping	replikacja transakcyjna / merge
MySQL / MariaDB	replikacja oparta o binlog z GTID	ta sama replikacja binlog, filtrowana po tabelach
Sybase ASE	—	Replication Server

Pojęcie: replication slot (PostgreSQL). Mechanizm rezerwujący na serwerze głównym segmenty WAL, dopóki konkretna replika (albo inny subskrybent) ich nie odczyta — gwarantuje, że replika, która na chwilę odłączy się od sieci, po powrocie znajdzie wciąż dostępne segmenty potrzebne do nadgonienia, zamiast luki w historii. Nieużywany slot, do którego nikt się nie podłącza, ma skutek uboczny opisany już na stronie o logu transakcyjnym — silnik trzyma WAL w nieskończoność, dopóki administrator go nie usunie.

> **Pojęcie: GTID (Global Transaction Identifier, MySQL).** Unikalny w skali całego klastra identyfikator nadawany każdej transakcji zapisanej w binlogu, zastępujący starszy sposób śledzenia pozycji repliki przez nazwę pliku i offset. GTID pozwala replice jednoznacznie stwierdzić, które transakcje już zastosowała, niezależnie od tego, z którego serwera źródłowego akurat replikuje — istotne przy zmianie serwera głównego po awarii.

Częste błędy

Traktowanie repliki jako zamiennika backupu. Opisane w sekcji „Po co to istnieje” — replika odtwarza błąd operatora równie wiernie jak poprawną zmianę, więc architektura oparta wyłącznie na replice, bez osobnego backupu z opóźnieniem, nie chroni przed najczęstszą w praktyce przyczyną utraty danych, czyli błędem ludzkim.

Failover kontra switchover. Pojęcie: failover. Awaryjne przełączenie roli serwera głównego na replikę w reakcji na nieoczekiwaną awarię oryginalnego serwera głównego — wykonywane w pośpiechu, często bez pewności, że wszystkie transakcje z serwera głównego zdążyły dotrzeć do repliki, zwłaszcza przy replikacji asynchronicznej. **Pojęcie: switchover.** Planowana, kontrolowana zamiana ról między serwerem głównym a repliką — na przykład przed pracami serwisowymi — wykonywana w sposób gwarantujący, że stary serwer główny dogania replikę co do ostatniej transakcji przed zamianą ról, więc nie traci się niczego. Mylenie tych dwóch pojęć prowadzi do fałszywego poczucia bezpieczeństwa: przetestowany, udany switchover w warunkach kontrolowanych nie dowodzi niczego o tym, jak zachowa się nieplanowany failover w warunkach rzeczywistej awarii sieci czy sprzętu.

Split-brain. Pojęcie: split-brain (rozjazd ról). Sytuacja, w której z powodu utraty łączności między serwerami więcej niż jeden z nich uznaje się jednocześnie za serwer główny i przyjmuje zapisy niezależnie od pozostałych — po przywróceniu łączności okazuje się, że oba mają zmiany, których drugi nie ma, a scalenie tych dwóch rozbieżnych historii bez utraty danych po żadnej stronie zwykle nie jest już możliwe automatycznie. Split-brain jest ryzykiem szczególnie przy automatycznym failoverze bez mechanizmu rozstrzygającego (arbitra, kworum), który jednoznacznie decyduje, który serwer ma prawo przejąć rolę głównego, gdy komunikacja między nimi się urywa.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — dobór trybu replikacji

System bankowy wymaga RPO równego zeru dla transakcji finansowych, a dwa centra danych łączy światłowód o opóźnieniu w obie strony rzędu dwóch milisekund. Sklep internetowy w innej firmie akceptuje RPO rzędu kilku sekund, a jego replika znajduje się w innym kraju, z opóźnieniem sieci rzędu osiemdziesięciu milisekund. Jaki tryb replikacji dobierzemy w każdym z tych dwóch przypadków i dlaczego zamiana byłaby błędem?

Odpowiedź

System bankowy z RPO zero i niskim opóźnieniem sieci (dwie milisekundy) dobrze znosi replikację synchroniczną — koszt doliczony do każdego `COMMIT`u jest niewielki, a w zamian żadna zatwierdzona transakcja finansowa nie ginie przy awarii. Sklep internetowy z repliką oddaloną o osiemdziesiąt milisekund i akceptowalnym RPO rzędu kilku sekund powinien zostać przy replikacji asynchronicznej — synchroniczność przy takim opóźnieniu sieci dodałaby osiemdziesiąt i więcej milisekund do każdej transakcji zapisującej, co przy dużym ruchu zauważalnie spowolniłoby cały serwis, a korzyść (RPO zero zamiast kilku sekund) nie jest tu wymagana ani opłacalna.

Pytania kontrolne

1. Dlaczego replika nie zastępuje backupu, mimo że obie utrzymują dodatkową kopię danych?

Odpowiedź

Replika dąży do jak najmniejszego opóźnienia względem bazy głównej i odzwierciedla każdą zmianę niemal natychmiast, łącznie z błędną — backup jest celowo opóźniony i istnieje właśnie po to, żeby przetrwać błąd, który dopiero co wydarzył się na produkcji. Replika chroni przed awarią serwera czy sieci, nie przed błędem logicznym w danych, więc architektura bez osobnego backupu nie ma żadnej ochrony przed tym drugim scenariuszem.

2. Czym różni się failover od switchover?

Odpowiedź

Switchover to planowana, kontrolowana zamiana ról między serwerem głównym a repliką, w której stary serwer główny zdąża dogonić replikę co do ostatniej transakcji przed przełączeniem — nic się nie traci. Failover to awaryjne przełączenie w reakcji na nieoczekiwaną awarię

serwera głównego, wykonywane bez gwarancji, że wszystkie transakcje zdążyły dotrzeć do repliki, zwłaszcza przy replikacji asynchronicznej — może się wiązać z utratą ostatnich, jeszcze nieprzesłanych zmian.

Powiązane: Log transakcyjny: WAL, redo, transaction log · RPO i RTO: dwie liczby, od których zaczyna się każdy plan backupu · Backup pełny, różnicowy, przyrostowy, logi · Co monitorujemy w bazie danych

Źródła: dokumentacja PostgreSQL, rozdział „High Availability, Load Balancing, and Replication” i „Logical Replication” · Oracle Data Guard Concepts and Administration · Microsoft Docs, „Always On Availability Groups” · dokumentacja MySQL, rozdział „Replication”

Revision #44
Created 2026-09-24 23:07:46 UTC by Tomasz Konieczny
Updated 2026-10-26 14:11:56 UTC by Tomasz Konieczny