Odtwarzanie do punktu w czasie (PITR)

> Ktoś wykonał `DELETE` bez `WHERE` o 14:32 — ta strona pokazuje, jak cofnąć bazę dokładnie do 14:31, zamiast tracić wszystko, co wydarzyło się od ostatniego backupu pełnego.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
PITR	Point-In-Time Recovery: odtworzenie bazy do dowolnej chwili, np. 14:31
RMAN	Recovery Manager: narzędzie Oracle do kopii zapasowych
WAL	Write-Ahead Log: dziennik PostgreSQL, zapis zmian przed zapisem danych

Fiszka

ZMIENIA SYSTEM

Odtworzenie do punktu w czasie: PostgreSQL

```
restore_command = 'cp /archiwum/wal/%f %p'
recovery_target_time = '2026-09-25 14:31:00'
```

Ustawiamy te dwa parametry w konfiguracji odzyskiwania nowej instancji uruchomionej na katalogu przywróconym z backupu bazowego — `restore_command` mówi silnikowi, skąd czytać kolejne segmenty WAL potrzebne do odtworzenia (katalog `pg_wal/` z backupu ich nie zawiera w całości, są archiwizowane osobno), a `recovery_target_time` wskazuje moment, w którym proces odtwarzania ma się zatrzymać.

Przykład:

```
LOG:  starting point-in-time recovery to 2026-09-25 14:31:00+02
LOG:  restored log file "000000010000000000000047" from archive
LOG:  redo starts at 0/47000028
LOG:  recovery stopping before commit of transaction 48213, time 2026-09-25
14:31:02.114281+02
LOG:  recovery has paused
HINT:  Execute pg_wal_replay_resume() to continue.
```

WAL = Write-Ahead Log: dziennik PostgreSQL, zapis zmian przed zapisem danych

ZMIENIA SYSTEM

Odtworzenie do punktu w czasie: Oracle RMAN

```
RUN {
  SET UNTIL TIME "TO_DATE('2026-09-25 14:31:00','YYYY-MM-DD HH24:MI:SS')";
  RESTORE DATABASE;
  RECOVER DATABASE;
}
```

Ustawiamy w RMAN docelowy moment poleceniem `SET UNTIL TIME`, przywracamy pliki danych z backupu (`RESTORE DATABASE`), a następnie odtwarzamy z archivelogów wszystkie zmiany aż do wskazanego momentu (`RECOVER DATABASE`) — Oracle automatycznie zatrzymuje odtwarzanie dokładnie na granicy wskazanego czasu.

Przykład:

```
Starting restore at 25-SEP-26
using channel ORA_DISK_1
channel ORA_DISK_1: restoring datafile 00001 to +DATA/orcl/datafile/system.256.1
Finished restore at 25-SEP-26

Starting recover at 25-SEP-26
media recovery complete, elapsed time: 00:01:47
Finished recover at 25-SEP-26
```

RMAN = Recovery Manager: narzędzie Oracle do kopii zapasowych · **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

Odtworzenie do punktu w czasie: SQL Server

```
RESTORE DATABASE NazwaBazy
FROM DISK = 'SciezkaDoPliku\NazwaBazy_full.bak'
WITH NORECOVERY;

RESTORE LOG NazwaBazy
FROM DISK = 'SciezkaDoPliku\NazwaBazy_log.trn'
WITH NORECOVERY, STOPAT = '2026-09-25 14:31:00';

RESTORE DATABASE NazwaBazy WITH RECOVERY;
```

Odtwarzamy backup pełny z opcją `NORECOVERY`, która celowo pozostawia bazę w stanie „w trakcie odtwarzania” gotowym na kolejne pliki, potem odtwarzamy backup logu z opcją `STOPAT` wskazującą dokładny moment zatrzymania, a na końcu zamykamy sekwencję poleceniem `RESTORE DATABASE ... WITH RECOVERY`, które otwiera bazę do normalnej pracy. Pominięcie `WITH RECOVERY` na końcu zostawia bazę zablokowaną do odczytu i zapisu w stanie „Restoring”.

Przykład:

```
Processed 21344 pages for database 'produkcja', file 'produkcja' on file 1.
RESTORE DATABASE ... WITH NORECOVERY  successfully processed 21344 pages.

Processed 112 pages for database 'produkcja', file 'produkcja_log' on file 1.
RESTORE LOG successfully processed 112 pages in 0.341 seconds (2.564 MB/sec).

RESTORE DATABASE successfully processed 0 pages in 0.012 seconds.
```

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

ZMIENIA SYSTEM

Odtworzenie do punktu w czasie: MySQL / MariaDB

```
mysqlbinlog --stop-datetime="2026-09-25 14:31:00" SciezkaDoPliku/binlog.000047 | mysql -u NazwaUsera -p NazwaBazy
```

Odtwarzamy backup pełny (na przykład z `mysqldump` albo z fizycznej kopii plików) jako punkt startowy, a następnie odgrywamy zawartość binloga narzędziem `mysqlbinlog`, ograniczając odtwarzane zdarzenia opcją `--stop-datetime` do momentu tuż przed błędnym poleceniem — wynik przekazujemy potokiem wprost do klienta `mysql`, który wykonuje te polecenia na przywróconej bazie.

Przykład:

```
Enter password:
Query OK, 0 rows affected (0.00 sec)
Query OK, 1204 rows affected (0.41 sec)
Query OK, 0 rows affected (0.00 sec)
```

mysqlbinlog = narzędzie konwertujące binarną zawartość pliku binloga MySQL/MariaDB na czytelny SQL

Po co to istnieje

Pojęcie: odtwarzanie do punktu w czasie (Point-In-Time Recovery, PITR).
Odtworzenie bazy nie do stanu z chwili wykonania backupu, tylko do dowolnego wskazanego momentu pomiędzy tym backupem a chwilą obecną — sekundy, minuty czy godziny po nim. PITR odpowiada na sytuację, w której backup pełny sam w sobie jest za mało precyzyjny: backup z nocy zawiera stan sprzed błędnego polecenia, ale też sprzed wszystkiego, co poprawnego wydarzyło się między nocą a chwilą awarii, a bez PITR jedyną alternatywą byłoby odtworzenie tego backupu i utrata całego dnia pracy razem z błędem.

PITR jest możliwy tylko wtedy, gdy istnieją jednocześnie dwa elementy: backup bazowy (pełny, ewentualnie uzupełniony różnicowym czy przyrostowym) oraz nieprzerwany łańcuch logu transakcyjnego (temat strony Log transakcyjny: WAL, redo, transaction log) obejmujący cały odcinek od chwili tego backupu aż do wskazanego momentu docelowego. Sam backup bazowy bez logów daje tylko stan sprzed backupu — sam log bez backupu bazowego nie ma punktu startowego, od którego mógłby się odtwarzać. Luka w łańcuchu logów w dowolnym miejscu między backupem a celem zatrzymuje odtwarzanie dokładnie w tym miejscu, tak samo jak przy każdym innym łańcuchu backupów opisanym na stronie Backup pełny, różnicowy, przyrostowy, logi.

Jak to działa

□□ Rozwiń: jak to działa pod spodem

Mechanizm PITR przebiega w każdym silniku według tej samej logiki, mimo różnic w nazwach poleceń: przywracamy backup bazowy jako punkt startowy, odtwarzamy (odgrywamy) z logu transakcyjnego kolejne zmiany aż do wskazanego momentu docelowego, a następnie zatrzymujemy odtwarzanie i otwieramy bazę do normalnej pracy w tym właśnie stanie — każda transakcja zatwierdzona po celu zostaje odrzucona, każda zatwierdzona przed nim jest obecna.

Wybór momentu docelowego. Cel odtwarzania da się wskazać na kilka sposobów w zależności od silnika i sytuacji: konkretnym znacznikiem czasu (najczęstszy przypadek — „odtwórz do 14:31:00”), konkretnym identyfikatorem transakcji (przydatne, gdy znamy dokładnie, która transakcja była ostatnią poprawną), albo nazwanym punktem przywracania ustawionym wcześniej ręcznie tuż przed ryzykowną operacją. Moment docelowy zawsze wybieramy tuż przed feralnym zdarzeniem, nie w jego trakcie — odtworzenie dokładnie na sekundę błędnego `DELETE` bywa niejednoznaczne co do tego, czy ta konkretna transakcja ma się jeszcze zmieścić.

Odtworzenie obok produkcji, nie na niej. Odtwarzanie do punktu w czasie wykonane wprost na działającej bazie produkcyjnej nadpisuje jej bieżący stan nieodwracalnie w trakcie samej procedury — jeśli coś pójdzie nie tak w połowie odtwarzania (zabraknie segmentu logu, cel okaże się źle dobrany), produkcja zostaje w stanie pośrednim, gorszym niż przed rozpoczęciem. Standardową praktyką jest odtworzenie na osobnym serwerze albo do osobnej instancji, zweryfikowanie, że dane w tym punkcie w czasie faktycznie są poprawne — na przykład sprawdzenie, że skasowana tabela znowu ma swoje wiersze — i dopiero wtedy decyzja, czy podmienić produkcję tą odtworzoną kopią, czy wyciągnąć z niej tylko brakujące dane i wstawić je z powrotem do wciąż działającej produkcji.

Częste błędy

Odtwarzanie wprost na bazie produkcyjnej. Opisane wyżej w sekcji „Jak to działa” — jeśli coś zawiedzie w trakcie odtwarzania na tym samym serwerze i tej samej bazie, cofnięcia się nie da, a stan pośredni bywa gorszy niż przed rozpoczęciem procedury.

Brak ciągłości archiwizacji logu. PITR jest tak dobre, jak najsłabsze ogniwo łańcucha logów między backupem bazowym a celem — przerwana archiwizacja WAL w PostgreSQL, wyłączony tryb `ARCHIVELOG` w Oracle na kilka godzin, albo skasowany plik binloga w MySQL, zamieniają PITR w odtworzenie tylko do momentu tej przerwy, nie do rzeczywiście pożądanego celu.

Cel odtwarzania ustawiony w złej strefie czasowej albo z opóźnieniem. Znacznik czasu podany w innej strefie niż ta, w której silnik zapisuje logi, przesuwa efektywny cel o kilka godzin bez żadnego komunikatu błędu — odtworzenie kończy się sukcesem, tylko nie tam, gdzie

zakładaliśmy. Weryfikacja strefy czasowej serwera przed ustaleniem `recovery_target_time` czy `STOPAT` jest krokiem, którego pominięcie ujawnia się dopiero po fakcie.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — dobór momentu docelowego

Log aplikacji pokazuje, że błędny `UPDATE` bez `WHERE` wykonał się o 14:32:07, a `COMMIT` tej transakcji zapisał się o 14:32:08. Jaki moment ustawimy jako cel PITR i dlaczego nie ustawiamy go dokładnie na 14:32:07?

Odpowiedź

Ustawiamy cel na moment wyraźnie przed rozpoczęciem transakcji, na przykład 14:32:00 albo nawet 14:31:55, zamiast dokładnie na 14:32:07. Ustawienie celu w samym środku transakcji jest niejednoznaczne co do tego, czy silnik potraktuje ją jako zatwierdzoną, czy nie — bezpieczniejsze jest cofnięcie się o kilka sekund więcej niż minimalnie potrzeba, bo utrata kilku dodatkowych sekund poprawnych danych jest dużo mniejszym problemem niż ryzyko, że błędna transakcja jednak się zmieści.

Pytania kontrolne

1. Dlaczego PITR nie działa, jeśli mamy tylko backup pełny sprzed tygodnia, bez żadnych logów transakcyjnych z okresu późniejszego?

Odpowiedź

PITR wymaga dwóch elementów naraz: punktu startowego (backup bazowy) i ciągłego zapisu wszystkich zmian od tego punktu aż do celu (log transakcyjny). Bez logów backup pełny sprzed tygodnia pozwala odtworzyć wyłącznie stan dokładnie z chwili tego backupu — nie ma żadnego źródła informacji o tym, co się zmieniło w ciągu tygodnia, więc nie da się dojść do żadnego innego momentu niż ten jeden, zapisany w samym backupie.

2. Czym różni się `RESTORE DATABASE ... WITH NORECOVERY` od `RESTORE DATABASE ... WITH RECOVERY` w SQL Server?

Odpowiedź

`NORECOVERY` przywraca dane, ale celowo pozostawia bazę w stanie odtwarzania, gotowym na przyjęcie kolejnych plików backupu (kolejnego różnicowego albo logów) — baza w tym stanie jest niedostępna dla zwykłych połączeń. `RECOVERY` (albo pominięcie tej opcji przy ostatnim kroku) kończy sekwencję odtwarzania i otwiera bazę do normalnej pracy, cofając niedokończone transakcje z ostatniego przywróconego logu — po tym poleceniu nie da się już dołożyć kolejnego pliku backupu do tej samej sekwencji.

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

Źródła: dokumentacja PostgreSQL, rozdział „Recovery Target Settings” · Oracle Database Backup and Recovery Reference, rozdział „RMAN RECOVER Command” · Microsoft Docs, „RESTORE Statements (Transact-SQL)” · dokumentacja MySQL, „Point-in-Time (Incremental) Recovery Using mysqlbinlog”

Revision #43
Created 2026-09-24 23:07:44 UTC by Tomasz Konieczny
Updated 2026-10-26 14:11:54 UTC by Tomasz Konieczny