Partycjonowanie

W najnowszych wersjach XAMPP partycjonowanie jest domyślnie włączone. Sprawdź to poleceniem:

SQL
SHOW PLUGINS WHERE Name = 'partition';

Zadanie 1: Partycjonowanie typu RANGE (Zakres dat)

Scenariusz: Tworzysz system logów systemowych. Logi szybko rosną, więc chcesz je dzielić według lat.

Zadanie: Utwórz tabelę `logs` z kolumnami `id` (INT), `message` (TEXT) oraz `log_date` (DATETIME). Zastosuj partycjonowanie po roku z kolumny `log_date`.

- Partycja `p_old`: lata przed 2023.
- Partycja `p_2023`: rok 2023.
- Partycja `p_2024`: rok 2024.
- Partycja `p_future`: wszystko powyżej.

Zadanie 2: Partycjonowanie typu LIST (Kategorie)

Scenariusz: Twoja aplikacja obsługuje oddziały sklepu w różnych regionach Polski.

Zadanie: Utwórz tabelę `stores` (id, nazwa, region_id). Podziel tabelę na 3 partycje na podstawie ID regionu:

- `p_polnoc`: regiony 1, 2, 3.
- `p_centralna`: regiony 4, 5, 6.
- `p_poludnie`: regiony 7, 8, 9.

Zadanie 3: Klucz główny a partycjonowanie

Scenariusz: Próbujesz utworzyć partycjonowaną tabelę, ale MySQL zwraca błąd.

Zadanie: Spróbuj utworzyć tabelę `users` (id - PK, email, registration_date), gdzie partycjonowanie odbywa się po `registration_date`.

- Zauważ błąd: "A PRIMARY KEY must include all columns in the partition function".
- **Rozwiązanie:** Napraw tabelę, tworząc klucz złożony `PRIMARY KEY (id, registration_date)`.

Zadanie 4: Partycjonowanie typu HASH (Równomierne rozłożenie)

Scenariusz: Masz tabelę z milionem produktów i chcesz, aby baza danych automatycznie "rozrzuciła" je do 4 plików danych, aby odciążyć dysk.

Zadanie: Utwórz tabelę `products` z kolumną `product_id` (INT, PK). Użyj `PARTITION BY HASH(product_id)` dzieląc dane na dokładnie 4 partycje.

Zadanie 5: Partycjonowanie typu KEY

Scenariusz: Podobnie jak w HASH, ale chcesz, aby MySQL sam zdecydował o algorytmie mieszającym, używając klucza głównego.

Zadanie: Utwórz tabelę `customers` z `customer_uuid` (VARCHAR(36) - PK). Zastosuj `PARTITION BY KEY()` na 5 partycji.

Zadanie 6: Zarządzanie – Dodawanie nowej partycji

Scenariusz: Rok 2024 dobiega końca. Musisz przygotować tabelę z Zadania 1 na rok 2025.

Zadanie: Za pomocą komendy `ALTER TABLE` dodaj nową partycję `p_2025` do tabeli `logs`. *Uwaga: Jeśli użyłeś `MAXVALUE` w Zadaniu 1, musisz najpierw zreorganizować partycję końcową (REORGANIZE PARTITION).*

Zadanie 7: Zarządzanie – Usuwanie starych danych

Scenariusz: Zgodnie z polityką RODO, musisz usunąć stare logi sprzed 2023 roku.

Zadanie: Zamiast używać powolnego `DELETE`, usuń całą partycję `p_old` z tabeli `logs`. Sprawdź, jak szybko operacja została wykonana w porównaniu do usuwania rekordów.

Zadanie 8: Podpartycjonowanie (Subpartitioning)

Scenariusz: Masz bardzo dużą tabelę sprzedaży. Chcesz ją dzielić najpierw po latach, a wewnątrz każdego roku jeszcze na 2 części (np. sprzedaż online i stacjonarna).

Zadanie: Utwórz tabelę `sales` (id, amount, sale_date, store_type_id).

- Główny podział (RANGE) po `YEAR(sale_date)`.
- Podpartycje (HASH) po `store_type_id`.

Zadanie 9: Sprawdzanie wydajności (EXPLAIN)

Scenariusz: Chcesz udowodnić, że partycjonowanie działa i silnik bazy danych nie przeszukuje całej tabeli.

Zadanie:

1. Wstaw do tabeli z Zadania 1 kilka rekordów z różnymi datami.
2. Wykonaj zapytanie: `EXPLAIN SELECT * FROM logs WHERE log_date = '2024-05-10';`.
3. W kolumnie `partitions` wyniku sprawdź, czy MySQL odpytał tylko partycję `p_2024` (zjawisko to nazywa się *Partition Pruning*).

Zadanie 10: Przenoszenie partycji do innej tabeli (Exchange)

Scenariusz: Chcesz przenieść dane z jednej partycji do osobnej tabeli (np. w celu archiwizacji lub szczegółowej analizy).

Zadanie:

1. Utwórz tabelę `logs_archive` o identycznej strukturze co `logs` (ale bez partycjonowania).
2. Użyj komendy `ALTER TABLE logs EXCHANGE PARTITION p_2023 WITH TABLE logs_archive;`.
3. Sprawdź, czy dane z 2023 roku "zniknęły" z tabeli głównej i pojawiły się w nowej.