**Người soạn: Lê Huy**

**Ngày soạn: 04/01/2024**

**Topic: Insert 10 triệu record vào MySQL**

**BÁO CÁO**

1. **Yêu cầu:**

Chèn10 triệu record vào bảng trên với điều kiện như dưới. Tìm cách tối ưu tốc độ để chèn nhanh nhất có thể.

Viết ra 1 phương án thơ ngây nhất, tìm hiểu có vấn đề gì ko, sau đó cải thiện dần, đưa ra thời gian chạy của từng phương án, sau mỗi lần cải thiện thì tăng lên được bao nhiêu

* ID không trùng
* FirstName, LastName, Address tìm thư viện Faker nào đó để sinh ngẫu nhiên (https://fakerphp.github.io/)
* Birthday kiểu Apr-03-2015 (Kiểu string)

1. **Tổng hợp kết quả (xếp theo thời gian thực hiện từ chậm đến nhanh):**

*Ghi chú: Kết quả thời gian là lần test có kết quả nhanh nhất đối với một phương án.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Phương án** | **Thời gian**  **(giây)** | **Toàn vẹn dữ liệu** | **Bảo mật** | **Ưu điểm** | **Nhược điểm** | **Đề xuất** |
| Single Insertion | 5950.94 | Bình thường | Bình thường | Đơn giản | Cực kỳ chậm đối với dữ liệu lớn | Sử dụng với dữ liệu nhỏ |
| Prepared Statement | 1292.82 | Ok | Ok | An toàn đối với SQL Injection, câu query được compile 1 lần | Tốc độ vẫn không tối ưu bằng bulk insert | Sử dụng khi có yêu cầu security và dữ liệu không lớn |
| Bulk Insertion using Temporary Table | 1043.206 | Ok | Bình thường | Thao tác với dữ liệu | Tốc độ vẫn không tối ưu bằng bulk insert | Sử dụng khi có yêu cầu thao tác dữ liệu, tránh can thiệp bảng chính |
| Bulk Insertion with Transaction | 635.91 | Ok | Bình thường | Hài hòa giữa tốc độ và an toàn dữ liệu | Vẫn còn overhead cost, chưa đảm bảo security bằng prepared statement | Sử dụng khi vừa muốn đảm bảo tốc độ và vừa an toàn dữ liệu |
| Parallel Insert with Transaction (10 threads) | 576.96 | Bình thường | Bình thường | Cùng lúc thực hiện nhiều tiến trình | Phụ thuộc phần cứng, setup phức tạp và tính toàn vẹn dữ liệu chưa cao | Sử dụng khi quản lý server tốt |
| Bulk Insertion without Transaction | 560.58 | Thấp | Bình thường | Nhanh và khá đơn giản | Toàn vẹn dữ liệu chưa cao | Sử dụng khi dữ liệu lớn và ưu tiên tốc độ |
| Load Data Infile | 133.375 | Thấp | Bình thường  (raw data) | Nhanh nhất | Nguy cơ SQL injection, toàn vẹn dữ liệu thấp | Sử dụng khi dữ liệu lớn và đã được chuẩn hóa, ưu tiên tốc độ |

1. **Chi tiết:**
2. **Nhận định một số yếu tố ảnh hưởng đến hiệu suất:**

Khi insert số lượng lớn record vào MySQL sử dụng PHP, có nhiều yếu tố có thể ảnh hưởng đến hiệu suất như:

* Phần cứng: CPU, Memory, RAM, tốc độ I/O của ổ cứng.
* Kết nối server.
* Tùy chỉnh thông số server, MySQL.
* *Batch-size của từng câu lệnh.*
* *Ảnh hưởng của transaction.*
* *Các phương thức tối ưu câu query.*

Trong phạm vi tìm hiểu và khả năng, trong báo cáo sẽ thử nghiệm một số phương án xoay quanh các các yếu tố batch-size của từng câu lệnh, ảnh hưởng của transaction và các phương thức tối ưu câu query. Không tùy chỉnh các thông số khác của server, MySQL, PHP

1. **Quá trình thử nghiệm và tối ưu:**

* **Tóm tắt:**
* Tạo database theo yêu cầu:

CREATE TABLE `user` (

id INT PRIMARY KEY,

first\_name VARCHAR(255),

last\_name VARCHAR(255),

address TEXT,

birthday VARCHAR(255)

);

* Sử dụng Faker để tạo file user.csv gồm 10 triệu record.
* Lựa chọn sử dụng MySQLi để thao tác với MySQL Server để tận dụng những tính năng được tối ưu cho riêng MySQL, bao gồm hiệu suất và tốc độ.
* Đọc dữ liệu từ file user.csv bao gồm 10 triệu record có sẵn.
* Với mỗi phương án, sử dụng hàm *microtime()* để tính toán thời gian chạy từ bắt đầu cho đến kết thúc quá trình.
* Test từ phương án đơn giản nhất và xoay quanh các yếu tố tác động. Từ đó tìm các cách để tối ưu.
* **Quá trình thực hiện như sau:**
  1. **Thực hiện từng câu lệnh INSERT INTO (Single Insert Statement):**
* **Phương pháp:**
  + Cá nhân đánh giá đây là cách đơn giản nhất để test và làm cơ sở để tối ưu.
  + Sử dụng vòng lặp while để đọc lần lượt từng dòng file user.csv.
  + Mỗi vòng lặp sử dụng 1 câu lệnh INSERT INTO để insert từng record vào database.
* **Kết quả: 5950 giây** (khoảng 99 phút)
* **Hiệu suất:** Đây là phương án đơn giản và có kết quả chậm nhất do hệ thống phải xử lý tuần tự từng câu lệnh INSERT INTO theo quy trình, tiêu tốn tài nguyên để xử lý (overhead cost). Đối với dữ liệu lớn gồm 10 triệu record, thời gian xử lý lâu do hệ thống liên tục thực hiện quá trình kết nối và thao tác giữa PHP và MySQL Server.

Đối với phương án này, việc kết hợp với transaction cũng không làm cải thiện hiệu suất vượt trội vì vậy không test phương án kết hợp với transaction cũng như các thông số khác.

* **File:** single\_insert.php
  1. **Insert gộp (Bulk Insert Statement):**
* **Phương pháp:**
  + Check max\_allowed\_packet tại MySQL để tìm khoảng batch-size tối đa (số lượng record ở mỗi câu INSERT INTO).
    - Đây là thông số biểu thị dung lượng tối đa của một single network packet mà MySQL Sever có thể xử lý.
    - Sau khi kiểm tra, max\_allowed\_packet là 32MB ~ 33,554,432 bytes.
    - Ước tính mỗi record có size khoảng 500 bytes => batch-size lớn nhất khoảng 33,554,432/500 ~ 67000 records.
  + Sử dụng vòng lặp while để đọc lần lượt từng dòng file user.csv.
  + Tùy chỉnh batch-size để để tìm phương án tốt nhất.
* **Kết quả:**

|  |  |
| --- | --- |
| **Batch-size** | **Kết quả** |
| 1000 records | 698.745 giây |
| 5000 records | 560.578 giây |
| 10000 records | 563.864 giây |
| 30000 records | 647.006 giây |
| 60000 records | 741.573 giây |

* **Hiệu suất:** Thay vì sử dụng 01 câu lệnh INSERT INTO cho 01 record, việc nhóm nhiều record vào từng câu lệnh INSERT INTO khiến giảm đáng kể overhead để xử lý từng giao dịch. Tốc độ xử lý tăng đáng kể so với phương án single insert, chỉ bằng khoảng 1/10.

Qua việc thử nghiệm bulk insert với nhiều kích cỡ batch-size khác nhau cho thấy batch-size tác động không nhỏ đến hiệu suất phương án, cụ thể:

* + Batch-size nhỏ (0 ~ 1000 records): Số lượng câu lệnh INSERT INTO vẫn còn nhiều ở mức này, khiến cho overhead cost vẫn còn lớn và chưa tối ưu.
  + Batch-size lớn và rất lớn (30000 ~ 60000 records): Mặc dù số lượng câu INSERT INTO ở mức này là thấp nhất. Tuy nhiên, tổng hòa hiệu suất vẫn không bằng các mức batch-size còn lại do nhiều yếu tố khác như tăng sử dụng memory, tốc độ xử lý phần cứng và các yếu tố khác.
  + Batch-size hợp lý (5000 ~ 10000 records): Qua thử nghiệm, batch-size ở khung này là hợp lý. Tổng hòa hiệu suất giữa overhead cost, tài nguyên phần cứng, ... khiến thời gian insert ở mức tối ưu trong phương án này.

Trên cơ sở kết quả này, sẽ tiếp tục test bulk insert kết hợp với transactions để kiểm tra hiệu suất. Mục đích kiểm tra nhằm thử nghiệm việc các tính năng bảo vệ tiến trình của transaction như locking, isolation có thể cải thiện hiệu suất hay không?

* **File:** bulk\_insert.php
  1. **Insert gộp kết hợp transaction (Bulk Insert Statement using Transactions):**
* **Phương pháp:**

Tạm thời sử dụng 02 mức batch-size tối ưu ở phương án bulk insert nêu trên (5000 và 10000) để test kết hợp transaction, theo 02 phương án:

* + Commit theo từng batch (Each)
  + Commit gộp cả quá trình (All).
* **Kết quả:**

|  |  |  |
| --- | --- | --- |
| **Batch-size** | **Commit** | **Kết quả** |
| 5000 records | Each | 635.913 giây |
| 5000 records | All | 655.801 giây |
| 10000 records | Each | 674.192 giây |
| 10000 records | All | Lock wait time out (do vượt thời gian chờ cho một transaction) |

* **Hiệu suất:** 
  + Trong bối cảnh test độc lập theo kịch bản này, ít các yếu tố ảnh hưởng đến tiến trình thì hiệu quả của việc bảo vệ tiến trình không thể bù đắp cho các overhead cost của transaction. Vì vậy, thời gian thực hiện insert có tăng do các bước trong transaction.
  + Tuy nhiên, trong thực tế, những lợi ích về nhiều mặt của transactions (ACID) cần được xem xét đến hơn là lợi ích chỉ về tốc độ insert (như kịch bản này).
* **File:** bulk\_insert\_transaction.php
  1. **Insert gộp kết hợp bảng tạm (Bulk Insert Statement using Temporary Table):**
* **Phương pháp:**
  + Test thử phương án *d.Insert gộp kết hợp transaction* (thông số tương tự nêu trên) và bảng tạm (TEMPORARY TABLE) để xem hiệu suất có được cải thiện:
    - Sử dụng query tạo một bảng tạm tương tự bảng chính cần insert.
    - Insert vào bảng tạm trước, khi quá trình insert vào bảng tạm hoàn tất sẽ thực hiện final insert vào bảng chính.
    - Tính chất bảng tạm chỉ lưu tạm trên memory, chưa thao tác đến I/O của Disk vì vậy tốc độ thao tác có thể giúp cải thiện hiệu suất?
* **Kết quả: 1043.206 giây** (khoảng 17 phút).
* **Hiệu suất:** 
  + Trong bối cảnh test độc lập theo kịch bản này, bảng chính đơn giản, ít ràng buộc, ít thao tác (trước khi insert) dữ liệu. Hiệu suất không có cải thiện so với phương án *d.Insert gộp kết hợp transaction .*
  + Tuy nhiên, trong thực tế, trong trường hợp cần thao tác với dữ liệu trước khi insert hoặc bảng chính có nhiều ràng buộc, index, dữ liệu phức tạp, … thì phương thức này có thể góp phần tăng hiệu suất và mang lại nhiều lợi ích.
* **File:** bulk\_insert\_temporary\_table.php
  1. **Prepared Statement:**
* **Phương pháp:** 
  + Sử dụng prepared statement để test việc biên dịch (compile) câu query 1 lần có làm cải thiện hiệu suất?
  + Sử dụng vòng while để lần lượt đưa giá trị của từng record vào prepared statement.
  + Kết hợp commit transaction theo batch-size 5000 records để test.
* **Kết quả: 1292.821 giây** (khoảng 21 phút).
* **Hiệu suất:** 
  + Việc sử dụng prepared statement giúp cải thiện hiệu suất đáng kể so với single insert. Tuy nhiên, prepared statement vẫn thao tác trên một tác vụ đơn lẻ cho mỗi record. Quá trình sẽ thêm các bước preparing, binding, executing, committing cho mỗi insert và mỗi batch. Vì vậy, trong trường hợp này, không thể có thời gian insert ngắn hơn so với bulk insert được.
  + Tuy nhiên, việc sử dụng prepared statement đã cho thấy việc câu query được compile 1 lần đã làm tăng hiệu suất. Áp dụng với số lượng lớn record vẫn phát huy hiệu quả. Trong thực tế, với yếu tố security luôn cần được chú trọng, đây là phương án nên xem xét đến (trong khi tốc độ vẫn tốt).
* **File:** prepared\_stmt\_insert.php
  1. **Insert đồng thời (Parallel Insert):**
* **Phương pháp:** 
  + Dùng php script chia file user.csv thành nhiều file nhỏ.
  + Dùng php script mẫu để insert dữ liệu với tham số là các file.
  + Sử dụng lệnh Linux để chạy cùng lúc các php script mẫu với tham số là các file để insert dữ liệu. So sánh hiệu suất các trường hợp.
  + Tương ứng với thông số CPU hiện tại của máy (4 Core và 8 Logical processors), test các trường hợp như sau:
    - Chạy 8 script cùng lúc, mỗi script insert 1.250.000 record.
    - Chạy 10 script cùng lúc, mỗi script insert 1.000.000 record.
    - Chạy 12 script cùng lúc, mỗi script insert 850.000 record (trừ script cuối).
  + Kết quả là thời gian của file cuối cùng, kết thúc quá trình.
* **Kết quả:**

|  |  |
| --- | --- |
| **Number of threads** | **Thời gian** |
| 8 | 706.975 giây |
| **10** | **576.957 giây** |
| 12 | 775.732 giây |

* **Hiệu suất:** 
  + Nếu không tính đến quá trình chia từ file gốc (10 triệu record), tốc độ insert hiệu quả. Trong đó, trường hợp sử dụng 10 thread hiệu quả nhất, tương đương bulk insert.
  + Tuy nhiên, để đảm bảo hiệu quả chung, cần kiểm soát nhiều file dữ liệu nhỏ và phải lưu ý khả năng xử lý multi threads của server.
* **File:** process\_file.php, run\_parallel.sh
  1. **LOAD DATA INFILE Statement:**
* **Phương pháp:** 
  + Đây là một phương thức đặc biệt mà MySQL hỗ trợ để thao tác load data lớn, nhiều record từ file vào MySQL.
  + Tuy nhiên, lưu ý thiết lập bảng và câu lệnh phải tương đồng.
* **Kết quả: 133.375 giây** (khoảng hơn 2 phút).
* **Hiệu suất:** 
  + Hiệu suất về tốc độ tối ưu nhất do đã rút ngắn tối thiểu quá trình tương tác giữa server và database.
  + Tuy nhiên, phương án này chỉ phát huy hiệu quả khi dữ liệu đã được chuẩn hóa, thiết lập theo logic để có thể thực hiện phương án này. Lưu ý đến vấn đề security và xung đột kiểu dữ liệu khi áp dụng phương thức này.
* **File:** load\_data\_infile.php