**BÁO CÁO**

**Tên:** Tô Kha Vỹ

**Link github:** <https://github.com/ToKhaVy/ten_mill_recs>

**Đề bài:** Tạo 1 bảng USER có các cột:ID (int), FirstName (varchar255), LastName (varchar255), Address (text), Birthday (varchar255)

Yêu cầu:

- 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. Chèn 10 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

2. Export toàn bộ dữ liệu trên ra định dạng CSV, 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

3. Export toàn bộ dữ liệu trên theo định dạng CSV theo điều kiện Address, Birthday và sắp xếp tăng giảm theo Birthday

※ 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

4. Với mỗi phương án trong yêu cầu 1, 2, 3 thì làm file báo cáo so sánh về thời gian chạy

※ Source code làm xong với mỗi phương án thì commit lên GIT để so sánh

※Deadline: Hết thứ 5 tuần sau(4/1/2024)

1. **Tạo một bảng user**

CREATE TABLE user (

ID INT PRIMARYKEY,

FirstName VARCHAR(255),

LastName VARCHAR(255),

Address TEXT,

Birthday VARCHAR(255)

);

1. **Tạo file CSV**

* 100.000 records: 26 sec.
* 1.000.000 records: 576,6581 sec (~10 min)
* 10.000.000 records: 3.179,9215 sec (~53 min)

1. **Chèn 10 triệu records từ file CSV vào Database**
   1. **Các yếu tố ảnh hưởng đến tốc độ chèn**
      * Phần cứng: như CPU, RAM, tốc độ đọc ghi của ổ cứng SSD/HDD.
      * Cấu hình của server MySQL.
      * Đỗ trễ mạng.
      * Kích thước giao dịch.
      * Tốc độ chèn dữ liệu vào DB của câu truy vấn `INSERT`
        + Kết nối DB (3)
        + Gửi câu truy vấn tới server DB (2)
        + Phân tích câu truy vấn (2)
        + Chèn dữ liệu theo hàng (1 \* số hàng dữ liệu)
        + Chèn index (1 \* số index)
        + Đóng DB (1)
   2. **Các cách chèn dữ liệu vào Database**
      * **Single insert:** Insert từng dòng dữ liệu từ file CSV vào DB thông qua vòng lặp While.

* **Kết quả:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **100.000** | **1.000.000** | **10.000.000** |
| **Single insert** | 58,68s | 1.368,69s  ~23min | 6.097,65s  ~101min |

* Đây là cách đơn giản nhất.
* Tốc độ chậm vì mỗi khi thực hiện một câu lệnh Insert, PHP sẽ gửi yêu cầu tới MySQL sever để kết nối với bảng dữ liệu cần được chèn dữ liệu. Lặp lại nhiều lần sẽ tốn thời gian kết nối DB ở mỗi lần làm tăng thời gian INSERT.
  + - **Batch insert:** Insert một khối dữ liệu trong một lần truy vấn.
* **Kết quả:** nhanh hơn khoảng 10 lần so với Single insert.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **100.000** | **1.000.000** | **10.000.000** |
| **Batch insert** | | | | |
| **Batch size** | **1.000** | 5,44s | 58,22s | 730,73s  ~12,2min |
| **5.000** | 6,57s | 71,92s | 813,1s  ~14min |
| **10.000** | 7,11s | 76,72s | 782,72s  ~13min |

* Tốc độ đã cải thiện hơn vì đã giảm được số lần mở kết nối đến DB.
* Nếu tăng số lượng dữ liệu cho một lần INSERT thì sẽ ảnh hưởng đến khả năng xử lý dữ liệu của server và làm giảm tốc độ chèn.
  + - **Transaction:** Lock nhiều câu lệnh INSERT vào chung một giao dịch để giảm tài nguyên quản lý giao dịch.
* **Kết quả:** nhanh hơn khoảng 1,2 lần so với Batch insert.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **100.000** | **1.000.000** | **10.000.000** |
| **Batch insert + Transaction** | | | | |
| **Batch size** | **1.000** | 5,34s | 50,67s | 636,03s  ~10,5min |
| **5.000** | 5,24s | 51,16s | 591,54s  ~9,8min |
| **10.000** | 7,11s | 51,19s | 518,42s  ~8,5min |

* Việc lock nhiều câu lệnh INSERT vào chung một giao dịch giúp tiết kiệm được tài nguyên MySQL server bỏ ra để thực thi giao dịch đó
  + - **Prepared statement:**
* **Kết quả:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **100.000** | **1.000.000** | **10.000.000** |
| **Prepared statement** | 56,91s | 569,34s  ~9,5min | 5.756,48s  ~96min |

* Tốc độ chèn đã được cải thiện hơn so với Single insert. Việc đã phân tích câu truy vấn sau đó chỉ chèn dữ liệu mà không cần phải phân tích lại đã giảm được thời gian insert dữ liệu.
  + - **Load Data Infile:** Đây là phương án có tốc độ tối ưu nhất (141 giây ~ 2,4 phút). Yêu cầu file dữ liệu phải được chuẩn hóa.

1. **Xuất 10 triệu records từ DB ra file CSV.**
   1. **Các yếu tố ảnh hưởng đến tốc độ xuất dữ liệu.**
      * Độ phức tạp của câu truy vấn: Những truy vấn với mức độ logic phức tạp sẽ gây mất thời gian xuất dữ liệu hơn.
      * Index: Index được thiết kế tốt có thể tăng tốc quá trình truy xuất dữ liệu.
      * Độ trễ mạng:
      * Tải trọng của server: Nếu truy xuất dữ liệu trong khi server đang thực thi nhiều truy vấn khác sẽ khiến cho server chịu tải nặng và tài nguyên cung cấp cho việc thực thi truy xuất dữ liệu sẽ giảm.
      * Phần cứng: như CPU, RAM, tốc độ đọc ghi của ổ cứng SSD/HDD.

**Bảng so sánh tổng hợp**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **100.000** | **1.000.000** | **10.000.000** |
| **Single insert** | 58,68s | 1.368,69s  ~23min | 6.097,65s  ~101min |
| **Batch insert** | | | |
| 1000 | 5,44s | 58,22s | 730,73s  ~12,2min |
| 5000 | 6,57s | 71,92s | 813,1s  ~14min |
| 10.000 | 7,11s | 76,72s | 782,72s  ~13min |
| **Batch insert + Transaction** | | | |
| 1000 | 5,34s | 50,67s | 636,03s  ~10,5min |
| 5000 | 5,24s | 51,16s | 591,54s  ~9,8min |
| 10.000 | 7,11s | 51,19s | 518,42s  ~8,5min |
| **Prepared statement** | 56,91s | 569,34s  ~9,5min | 5.756,48s  ~96min |
| **LOAD DATA INFILE** | | | 141s ~2,35min |

|  |  |
| --- | --- |
| **Batch Export 10.000.000 records** | |
| **100.000** | 1.041,9s  ~17min |
| **200.000** | 569,03s  ~9,5min |
| **300.000** | 404,58s  ~6,8min |
| **400.000** | 331,98s  ~5,5min |
| **500.000** | 277,56s  ~4,6min |
| **550.000** | 279,42s  ~4,6min |
| **560.000** | 251,8s  ~4,2min |
| **570.000** | 258,4s  ~4,3min |
| **580.000** | Tràn bộ nhớ |
| **600.000** | Tràn bộ nhớ |