**BÁO CÁO**

**Tên:** Tô Kha Vỹ

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

**Đề bài:** Chuyển source code ở task 1 từ MySQL sang SQLite, và so sánh khi chạy source code phiên bản tốt nhất của mình trên 2 hệ quản trị CSDL này thì performance khác nhau như thế nào.

1. **Tạo một bảng user**

CREATE TABLE user (

         ID         INT PRIMARY KEY,

         FirstName  TEXT,

         LastName   TEXT,

         Address    TEXT,

         Birthday   TEXT )

**File:** sqlite/sqlite\_create\_tb.php

1. **Khác nhau giữa SQLite và MySQL**

|  |  |  |
| --- | --- | --- |
|  | **MySQL** | **SQLite** |
| 1. **Cấu trúc** | Hệ thống quản lý CSDL client-server, hỗ trợ nhiều kết nối đồng thời, phù hợp với ứng dụng quy mô lớn | Một CSDL không cần máy chủ, được nhúng trực tiếp vào ứng dụng, lựa chọn tốt cho lưu trữ local hoặc dự án nhỏ và trung bình. |
| 1. **Triển khai** | Triển khai như một máy chủ riêng biệt và tương tác với user thông qua mạng. | Nhúng trực tiếp vào ứng dùng và CSDL được lưu trữ dưới dạng tệp trên ổ đĩa. |
| 1. **Kết nối đồng thời** | Hỗ trợ nhiều kết nối đồng thời | Được thiết kế đơn giản, dùng trong các tình huống ít kết nối đồng thời |
| 1. **Server** | Yêu cầu một máy chủ riêng biệt. Cung cấp một CSDL tập trung mà nhiều user có thể truy cập đồng thời | Hoạt động như một DB Engine không server, tự chứa, việc triển khai và quản lý dễ dàng hơn. |
| 1. **Kiểu dữ liệu và tính năng** | Hỗ trợ một loạt các kiểu dữ liệu và tính năng nâng cao, phù hợp cho các ứng dụng phức tạp. Cung cấp các tính năng như thủ tục lưu trữ, triggers và views. | Hỗ trợ một phần của kiểu dữ liệu so với MySQL và là một hệ thống nhẹ hơn. Không hỗ trợ thủ tục lưu trữ và triggers theo cách giống MySQL |
| 1. **Cấu hình** | Yêu cầu cấu hình quản trị, đặc biệt là khi làm việc với các CSDL lớn và hệ thống phân tán | Cần ít cấu hình hơn. Thường được coi là một CSDL không cần cấu hình, giúp dễ dàng thiết lập và sử dụng. |
| 1. **Các trường hợp sử dụng** | Phù hợp cho các ứng dụng cso tính mở rộng, đồng thời cao | Phù hợp cho các hệ thống nhúng, ứng dụng di động, ứng dụng web nhỏ đến trung bình. |

1. **Các yếu tố ảnh hưởng đến tốc độ import dữ liệu trong SQLite**
   1. **Kích thước giao dịch:** Kích thước của mỗi giao dịch có thể ảnh hưởng đến tốc độ. Giao dịch lớn hơn có thể cải thiện hiệu suất. Nhưng giao dịch lớn cũng có thể làm giảm hiệu suất do vượt quá khả năng chịu tải của ứng dụng.
   2. **Indexes:** Việc sử dụng indexes có thể làm chậm quá trình insert. Có thể tạm ngưng dùng indexes trước khi insert và sau đó khôi phục lại.
   3. **WAL mode:** Chế độ Write-Ahead Logging có thể cải thiện hiệu suất insert ở mức độ cao.
   4. **Journal mode:** Chế độ nhật ký thích hợ cho DELETE, TRUNCATE hoặc PERSIST để tối ưu hiệu suất.
   5. **Kích thước bộ nhớ cache:** Tăng kích thước bộ nhớ cache có thể giúp cải thiện tốc độ insert.
   6. **Tốc độ ổ đĩa và phần cứng.**
2. **Khi commit một giao dịch trong SQLite**
   1. **Durability (tính bền vững):** các thay đổi được thực hiện trong giao dịch trước khi commit sẽ trở thành bền vững và sẽ không bị mất khi có sự cố hệ thống hoặc khởi động lại
   2. **Atomicity (tính nguyên tử):** nếu một phần của giao dịch thất bại, toàn bộ giao dịch sẽ bị rollback và không có thay đổi nào được áp dụng.
   3. **Consistency (tính nhất quán):** dữ liệu tuân thủ các ràng buộc (constraints) và quy tắc của CSDL.
   4. **Isolation (tính cô lập):** trong quá trình giao dịch, các thay đổi chỉ là hiển thị đối với giao dịch đó và không ảnh hưởng đến các giao dịch khác. Sau khi commit, thay đổi mới có thể thấy được bởi giao dịch khác.
3. **Khi commit 1 khối giao dịch so với commit từng câu Insert**
   1. **Commit 1 khối giao dịch**
      * **Ưu điểm:** Hiệu suất cao hơn vì chỉ có một lệnh commit duy nhất.
      * **Nhược điệm:** Nếu gặp sự cố, toàn bộ giao dịch sẽ bị rollback, không giữ lại bất kỳ thay đổi nào
   2. **Commit từng câu Insert**
      * **Ưu điểm:** Nếu một câu Insert gặp vấn đề, chỉ ảnh hưởng đến câu đó và không làm ảnh hưởng đến các câu khác. Dữ liệu đã được commit vẫn nguyên vẹn
      * **Nhược điểm:** Làm chậm hiệu suất do số lượng lệnh commit nhiều hơn.
4. **Import 10 triệu records vào SQLite.**
   1. **Batch Insert + Transaction**

* **Kết quả**

|  |  |  |
| --- | --- | --- |
| **Batch size** | **SQLite** | **MySQL** |
| 1000 | 605,6 s ~ 10,1 min | 636,03 s ~ 10,5 min |
| **5000** | **535,27 s ~ 8,9 min** | 591,54s ~ 9,8min |
| 10.000 | 586,56 s ~ 9,8 min | 518,42s ~ 8,5min |

* Tốc độ INSERT ở cả hay DB trong trường hợp này là ngang nhau vì sử dụng Transaction commit tất cả các khối insert.

**File:** sqlite/batch\_import.php

* 1. **Batch Insert**
* **Kết quả**

|  |  |  |
| --- | --- | --- |
| **Batch size** | **SQLite** | **MySQL** |
| 1000 | 533,5 s ~8,9 min | 730,73 s ~ 12,2 min |
| **5000** | **532,5 s ~ 8,8 min** | 813,1 s ~ 14 min |
| 10.000 | 521,5 s ~ 8,7 min | 782,72 s ~ 13 min |

* Tốc độ INSERT ở trường hợp này ở SQLite nhanh hơn do SQLite có hỗ trợ AUTO COMMIT.

**File:** sqlite/batch\_import\_no\_transaction.php

1. **Export 10 triệu records ra file CSV**
   1. **Batch export + Transaction**

* **Kết quả**

|  |  |  |
| --- | --- | --- |
| **Batch size** | **SQLite** | **MySQL** |
| 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 | 137,6 s ~ 2,3 min | 279,42s ~ 4,6min |
| 560.000 | 136,1 s ~ 2,26 min | 251,8s ~ 4,2min |
| 570.000 | 136,4 s ~ 2,26 min | 258,4s ~ 4,3min |
| 580.000 | 132,7 s ~ 2,2 min | Tràn bộ nhớ |
| 600.000 | 130,24 s ~ 2,17 min | Tràn bộ nhớ |

* Tốc độ và giới hạn Batch size thự thi SQLite tốt hơn MySQL.

**File:** sqlite/batch\_export.php

1. **So sánh các câu lệnh trong SQLite và MySQL**

|  |  |  |
| --- | --- | --- |
| **Câu lệnh** | **SQLite** | **MySQL** |
| 1. **Transaction** | | |
| Save point | Không hỗ trợ SAVE POINT, không thể tạo các điểm lưu trữ trung gian để phục hồi giao dịch ở mức độ nhỏ. | Hỗ trợ SAVE POINT để tạo điểm lưu trữ trung gian trong gioa dịch, giúp quản lý và phục hồi dữ liệu một cách linh hoạt. |
| Auto commit | Giao dịch sẽ tự động COMMIT nếu không có lệnh ROLLBACK hoặc COMMIT được thực hiện. |  |
| 1. **INSERT** | | |
| Atomacity | Hỗ trợ tính nguyên tử. Mọi thay đổi sẽ được thực hiện hoặc không, nếu xảy ra lỗi thì toàn bộ sẽ bị rollback. | Phụ thuộc vào cấu hình cụ thể của giao dịch. |
| Auto-increment | Dùng cột “AUTOINCREMENT” để định nghĩa cột tự tăng. Tuy nhiên không cần từ khóa này cũng có thể tạo một cột tự tăng. | Dùng “AUTO\_INCREMENT” để định nghĩa cột tự tăng. |
| Giá trị cho cột tự tăng | Cho phép chèn nhưng không sử dụng. | Không cho phép chèn giá trị vào cột tự tăng. |
| Chèn nhiều dòng | Chèn nhiều dòng với cú pháp “INSERT INTO … VALUES (),(),(),…. | Cú pháp tương tự. |
| 1. **UPDATE** | | |
| Cú pháp | UPDATE table\_name  SET column1 = values1,…  WHERE condition; | Cú pháp tương tự. |
| JOIN | Không hỗ trợ. Phải thực hiện UPDATE trên từng bảng. | Hỗ trợ mệnh đề JOIN khi UPDATE. |
| ORDER BY | Không hỗ trợ ORDER BY trong câu lệnh UDATE. | Hỗ trợ ORDER BY trong lệnh UPDATE, giúp xác định thứ tự cụ thể khi cập nhật. |
| 1. **DELETE** | | |
| Syntax cơ bản | Dùng “DELETE FROM tb\_name WHERE condition” để xóa chính xác dữ liệu theo điều kiện. | Cú pháp tương tự. |
| Xóa toàn bộ dữ liệu | Có thể sử dụng câu lệnh DELETE không có mệnh đề WHERE để xóa toàn bộ dữ liệu của bảng. | Xóa toàn bộ dữ liệu dùng lệnh “TRUNCATE TABLE tb\_name;” |
| 1. **SELECT** | | |
| Syntax cơ bản | Cơ bản giống nhau | Cơ bản giống nhau |
| Đặc biệt |  | SELECT INTO OUTFILE  SELECT DUMPFILE |