**Báo cáo thực hành NHibernate**

Mục lục

[1) Cơ sở dữ liệu đã tạo: 1](#_Toc15711248)

[2) Tối ưu các câu truy vấn 2](#_Toc15711249)

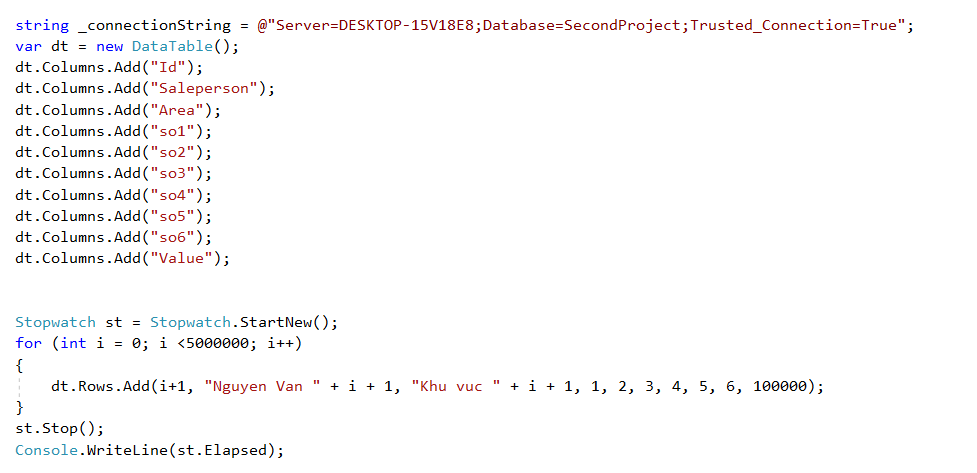
[3) Thực hiện mutil task 2](#_Toc15711250)

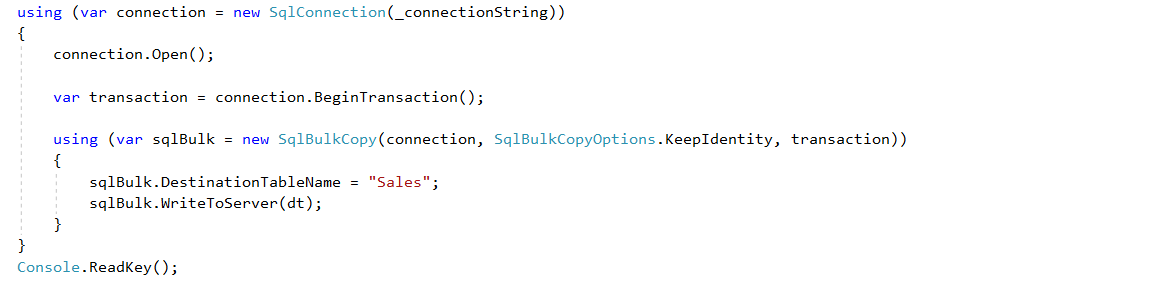
[4) Hiệu năng 2](#_Toc15711251)

[5) Tìm hiểu về Table partitioning (kỹ thuật phân chia bảng). 4](#_Toc15711252)

# Cơ sở dữ liệu đã tạo:

* Sales (Id, Saleperson, Area, so1, so2, so3, so4, so5, so6, Value).
* Đã thử: Dùng vòng lặp sử dụng phương thức Save để Insert 5tr record sẽ mất hàng giờ để hoàn thành.
* Sử dụng SqlBulkCopy insert 5tr record mất thời gian là: 00:00:21.6946393.

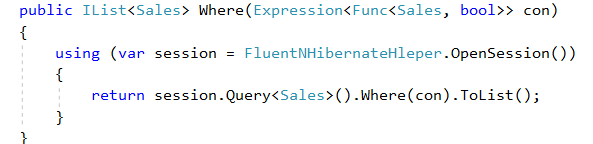




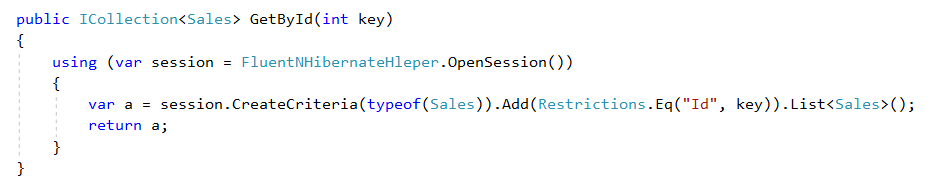
* Đầu tiên, tạo 1 DataTable giống như cấu trúc của bảng trong CSDL. Sau đó tạo vòng lặp ghi vào DataTable này 5tr record.
* Tiếp theo, tạo 1 sqlBulkCopy và sử dụng nó để ghi nội dung của DataTable vào bảng Sales trong CSDL.
* Cơ chế như sau: Tạo kết nối->Mở kết nối->Tạo giao dịch->Chuyển kết nối và các đối tượng giao dịch đến SqlBulkCopy.
* Nên sử dụng lệnh sqlBulk.BatchSize vì khi số lượng record trong nguồn tăng lên, bộ nhớ sử dụng bởi sqlBulkCopy cũng tăng lên do đó nếu sử dụng lệnh này thì có thể giói hạn lại bộ nhớ giảm thiểu sô lượng record ghi được trong mỗi batch do đó sẽ tiêu thụ bộ nhỡ ít hơn. Tuỳ thuộc vào hoàn cảnh mà sử dụng các BatchSize khác nhau. Máy hiện đang sử dụng 5000.

# Tối ưu các câu truy vấn

* Cho 100 Task chạy từ 1->100.
* Việc truy vấn bằng các câu lệnh insert, delete, update sử dụng các phương thức của session interface trong nhibernate. Get dữ liệu thực hiện theo 2 hàm:
* 1 hàm là Lambda expression, thời gian truy vấn:

****

* 1 hàm sử dụng Criteria:

****

* Nhận xét:
* Cơ chế của các phương thức do session interface là openSession->Thực thi câu lệnh ->commit transaction->Đóng session lại. Tuy nhiên, nếu ta lỡ Insert 1 dòng đã có rồi hoặc Update, Delete những dòng không có nó sẽ ra 1 Exception: trả về 1 dòng nhưng thực tế lại có 0 dòng trả về. Vì vậy ta thực hiện 1 câu lệnh try catch những trường hợp exception như thế ta cần rollback transaction lại.
* Đối với lệnh Get khi sử dụng criteria sẽ nhanh hơn khi sử dụng lambda expression. Nhưng việc sử dụng 2 câu lệnh này không có vấn đề so với SQL vì cú pháp khá đơn giản, hỗ trợ bởi IDE thì có thêm các điều kiện truy vấn với SQL.

# Thực hiện mutil task

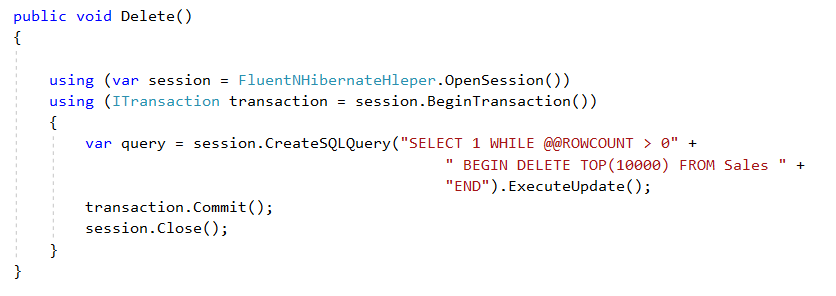
* Thực hiện Get theo Id liên tục trong 100 Task. Mỗi Task thực hiện 10 lần. Id random từ

1->1000000

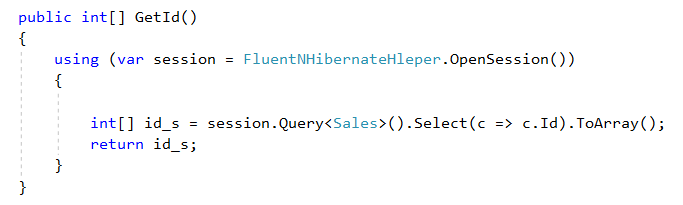
* Đo thời gian: 00:00:07.3393963
* Nhận xét: Các task thực hiện đan xen lẫn nhau khi task này chưa hoàn thành thì task kia đã bắt đầu dẫn đến việc khó kiểm soát dữ liệu. Tuy nhiên nếu chỉ Get Id theo 100 Task thì nó không ảnh hưởng gì đến việc truy xuất dữ liệu. Vì các câu lệnh Get này tương ứng với Select trong database nó sẽ thực hiện chế độ share lock cho phép đồng thời đọc cùng 1 dữ liệu.

# Hiệu năng

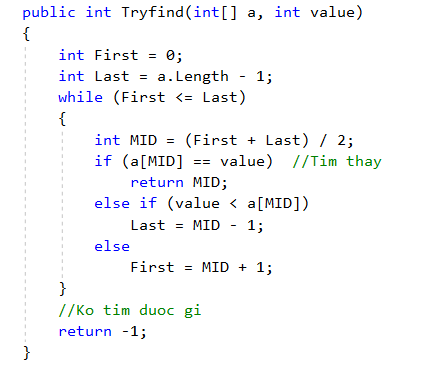
* Giữ mức ổn dịn 5tr record sử dụng câu lệnh T-SQL để xoá tất cả 5tr record cùng lúc:

****

* Thời gian truy vấn là: 
* Thực hiện 10 task mỗi task thực hiện 10 lần Get random theo Id tất cả các trường, thực hiện update các record chia hết cho 2 và xoá các record chia hết cho 4, sau đó insert theo Sales đã cho.
* Sử dụng hàm GetById.
* Các lệnh insert, delete, update sử dụng các phương thức trong session.
* Nhận xét: Quá trình thực hiện đan xen lẫn nhau diễn ra bình thường, không có lỗi xảy ra vì trước đó đã thực hiện việc rollback dữ liệu.
* 1 số giải pháp đã thử (ngoài việc rollback dữ liệu):
* Sử dụng lock: Đảm bảo rằng các trường cân bằng không thể cập nhật đồng thời giữa 2 luồng. Khi cố gắng gọi các phương thức. Giúp tránh deadlock bằng cách chỉ cho lần lượt từng task vào. Lock yêu cầu truyền vào 1 đối tượng object.
* Cơ chế: Khi khoá được giữ, luồng giữ khoá có thể lấy lại và giải phóng khoá. Bất kỳ 1 luông nào khác đều bị chặn không lấy khoá và đợi cho đến khi khoá được giải phóng
* Nhận xét: Giúp tránh được deadlock. Tuy nhiên, việc thực hiện tuần tự sẽ làm giảm hiệu năng của chương trình. Vì vậy, chỉ thực sự sử dụng khi nghi ngờ rằng có deadlock xảy ra.
* Xây dựng hàm kiểm tra tồn tại của Id trong CSDL:

****

**1**) Hàm lấy danh sách Id trong CSDL

****

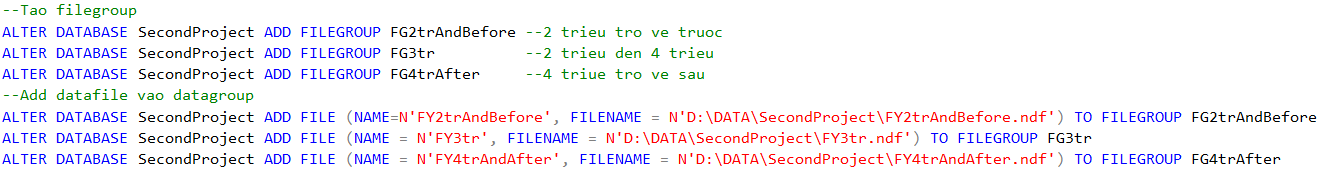
**2**) Giải thuật tìm kiếm nhị phân.

* Thực hiện: Trước hết xây dựng 1 hàm lấy danh sách Id trong CSDL sau đó tiến hành thực hiện việc tìm kiếm theo Id đã lấy nếu tìm thấy thành công thì thực hiện việc Get dữ liệu, thêm, sửa, xoá đã nêu trên nếu không thì cho tiếp tục tìm kiếm:
* Nhận xét: Việc thực hiện này cũng có khả năng giải quyết được vấn đề. Tuy nhiên, sẽ tốn rất nhiều thời gian đối với dữ liệu có hàng triệu records.
* Xây dựng các hàm trên bằng câu lệnh SQL, tuy thời gian có nhanh hơn nhiều lần so với các xử lý trên nhưng có 1 vấn đề nhỏ là nó lặp lại việc thực hiện của mình (chẳng hạn như đã xoá 1 đối tượng sale rồi, nhưng vẫn tiếp tục thực hiện việc xoá hoặc update tiếp theo).

# Tìm hiểu về Table partitioning (kỹ thuật phân chia bảng).

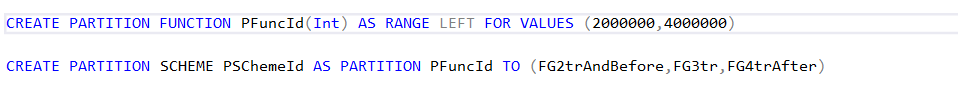
* Định nghĩa: Table partitioning là kỹ thuật phân chia bảng thành từng đoạn nhằm quản lý hiệu quả CSDL với dung lượng lớn, cung cấp 1 phương pháp để chia dữ liệu những bảng lớn với những vùng nhỏ hơn. Bằng phương pháp đó, nó tạo ra 1 phiên quản trị cơ sở dữ liệu dễ dàng hơn khi backup, loading, phục hồi và truy vấn dữ liệu.
* Tiện lợi:
* Có thể backup restore 1 đoạn mà không ảnh hưởng đến các đoạn còn lại.
* Cho phép loại bỏ dữ liệu nguyên 1 đoạn ra khỉ bảng thay vì dùng lệnh DELETE (dùng Switch out), tương tự cho phép thêm dữ liệu từ 1 bảng khác thành đoạn mới (Switch in).
* Cải tiến hiệu năng:
* Khi 1 câu lệnh chỉ cần lấy dữ liệu ở 1 đoạn nào đó thì hệ thống chỉ cần truy cập vào đoạn đó và bỏ qua các đoạn còn lại.
* Khi các đoạn dữ liệu được lưu trữ ở các ổ cứng khác nhau sẽ làm giảm tranh chấp giữa các câu lệnh.
* Phân chia đoạn các bảng dựa trên Partition function và Partition Scheme.

**Ứng dụng vào trong CSDL hiện có:**

****

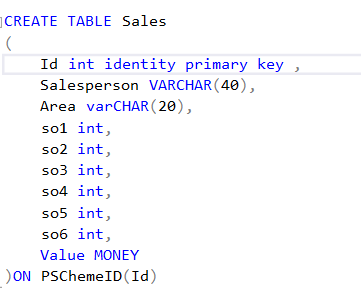
**3**) Tạo file group và add datafile vào filegroup

* Đối với dữ liệu hàng triệu record ta có thể chia làm nhiều filegroup khác nhau để dễ quản lý, mỗi file group lưu trữ 1 đoạn. Ở đây mục đích là lưu trữ Theo Id, Đối với dữ liệu này (Bảng Sales ở mục 1) thì ta sẽ tiến hành tạo 3 file group. FG2trAndBefore sẽ lưu trữ Id 2tr trở về trước, FG3tr sẽ lưu trữ Id từ 2tr001 đến 4tr và FG4trAfter sẽ lưu trữ Id từ 4tr001 trở về sau.
* Ta tiếp tục Add các datagroup vào datafile theo đường dẫn như trên.

****

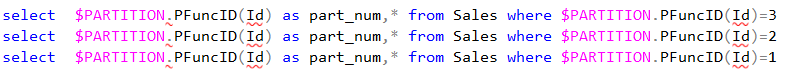
**4**) Tạo partition function và partitionscheme

* Partition function: Để xác định giá trị biên cho các đoạn. Hệ thống dựa vào hàm này để xác định đoạn tương ứng với mỗi record. Dựa vào bảng trên ta có 3 đoạn nên có 2 giá trị biên như đã phân ra trước nên giá trị biên là 2tr và 4tr. Ở hàm function trên ta sẽ sử dụng thuộc tính Id để phân nhóm nên sử dụng kiểu Int.
* Partition scheme: Ánh xạ các đoạn khai báo trong partition function vào các file group.



5) Tạo bảbanrguwr dụng partitition scheme

* Tạo bảng dùng Partition Scheme. Như vậy là dữ liệu Id đã được phân ra từng đoạn nằm trong từng file group (ở đây có 3 file group) ứng với mỗi trường là part\_num.
* Partition number: là Id ứng với từng file group được tạo.
* Object\_id: là một số Id duy nhất cho 1 dối tượng CSDL, được SQL sử dụng nội bộ.

****

**6**) Cú pháp load Các record theo từng filegroup