COLE.VN

Connecting knowledge

COLE.VN
Connecting knowledge

Turning SQL

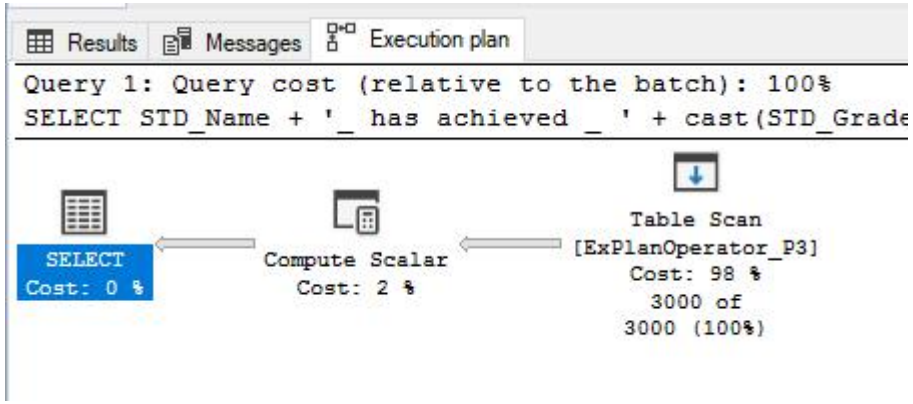
Trình bày: Nguyễn Văn Phúc

NỘI DUNG CHÍNH

1. Nguyên nhân gây chậm truy vấn SQL
2. Công cụ
3. Tối ưu hóa câu lệnh SQL
4. Execution Plan (Kế hoạch thực thi)
5.
6.

Execution Plans (Kế hoạch thực thi)

- Execution Plans là lược đồ thể hiện các bước khác nhau để search engine của sql server lấy dữ liệu từ các bảng.
- Khi thực thi một mệnh đề query (select statements), search engine của sql server sẽ sinh ra nhiều Kế hoạch thực thi khác nhau và chọn kế hoạch tốt nhất để thực thi và trả lại kết quả cho người dùng.
  - *Estimated Execution Plan* (Kế hoạch thực thi ước tính): Sinh ra trước khi một câu query được thực thi
  - *Actual Execution Plan* (Kế hoạch thực thi thực tế): Kế hoạch thực thi được dùng để thực hiện câu query select

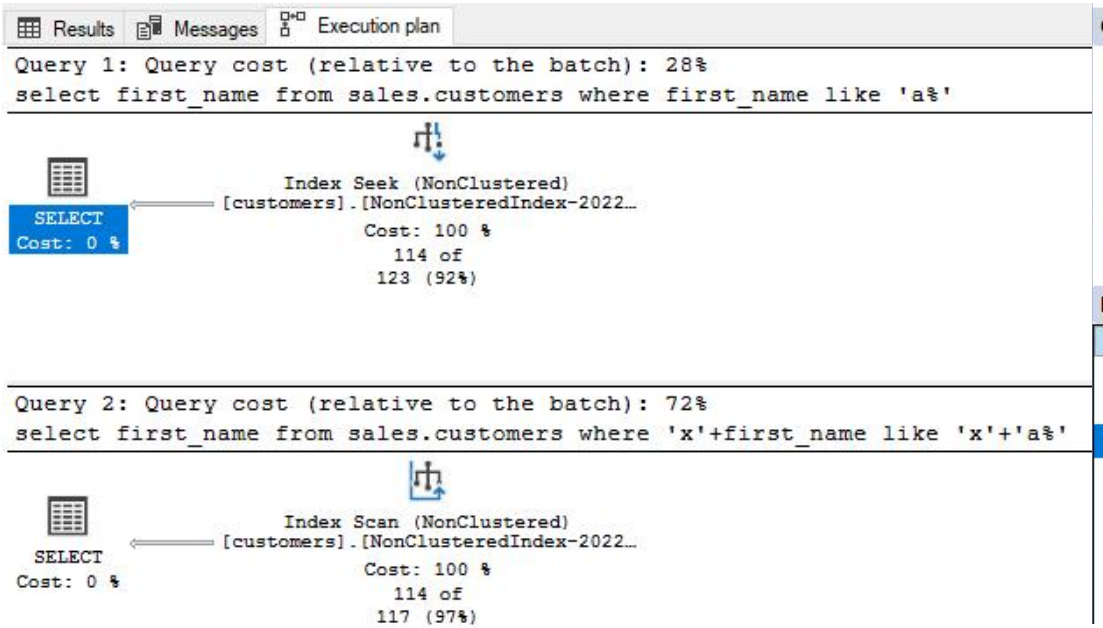


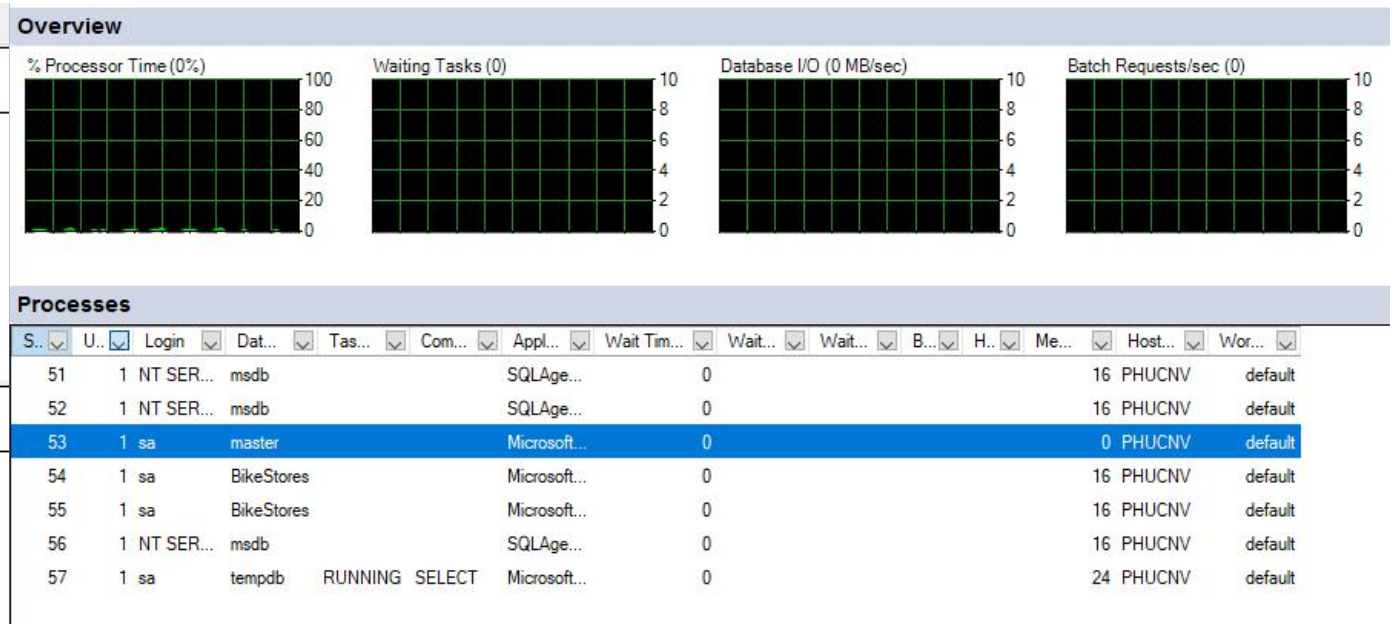
Nguyên nhân gây chậm truy vấn SQL

1. Không có hoặc thiếu Index.
2. Trả về các dữ liệu không cần thiết.
3. Locks hoặc deadlocks.
4. Các câu truy vấn được viết nghèo nàn.
5. Thiếu bộ nhớ.

Công cụ

1. Execution Plan
2. Activity Monitor





COLE.VN
Connecting knowledge

Tối ưu hóa câu lệnh SQL

- Index
  - Một chỉ mục trong một Database là tương tự như một chỉ mục trong Mục lục của cuốn sách.
  - Index giúp Database Search Engine tăng nhanh thời gian và hiệu suất thu thập dữ liệu.
  - Index giúp tăng tốc các truy vấn SELECT và các mệnh đề WHERE.
  - Làm chậm việc dữ liệu nhập vào, với các lệnh UPDATE và INSERT.
  - Index có thể được tạo hoặc xóa mà không ảnh hưởng tới dữ liệu.
- Cách dùng Index
  - Search trên trường nào thì có thể xem xét tạo index trên trường đó
  - Index trên nhiều trường

Tối ưu hóa câu lệnh SQL

- Thu hẹp giá trị trả về
  - Tiết kiệm bộ nhớ
  - Tiết kiệm băng thông khi truyền từ server về client

```
select
    *
from Sales.Customer a
    join Person.Person b on b.BusinessEntityID=a.PersonID
```

```
select
    b.FirstName
    ,b.LastName
from Sales.Customer a
    join Person.Person b on b.BusinessEntityID=a.PersonID
```

Tối ưu hóa câu lệnh SQL

- Sử dụng Like không hợp lý sẽ làm chậm truy vấn

```
select * from Person.Person where FirstName like '%C%'
```

```
select * from Person.Person where FirstName like 'C%'
```

- Hạn chế sử dụng biểu thức lên column

```
select
    b.FirstName
    ,b.LastName
from Sales.Customer a
    join Person.Person b
        on b.BusinessEntityID=a.PersonID
where
    b.EmailPromotion = 1
```

```
select
    b.FirstName
    ,b.LastName
from Sales.Customer a
    join Person.Person b
        on b.BusinessEntityID=a.PersonID
where
    b.EmailPromotion + 1 = 2
```

2. Tối ưu hóa câu lệnh SQL

- Chỉ sử dụng DISTINCT và ORDER BY khi thực sự cần thiết

```
select * from Sales.Customer

select * from Sales.Customer order by ModifiedDate
```

- Sử dụng EXISTS để kiểm tra dữ liệu có tồn tại hay không thay vì dùng COUNT
- Sử dụng SQL Procudure
  - Giảm lượng dữ liệu truyền đến Server SP được lưu sẵn ở phía server do đó không cần phải gửi cả câu lệnh SQL dài tới server mà chỉ cần gửi tham số
  - SP được biên dịch ngay ở lần đầu chạy, những lần sau chạy SP sẽ sử dụng lại file đã biên dịch trước đó nên tốc độ sẽ nhanh hơn

2. Tối ưu hóa câu lệnh SQL

- Hạn chế sử dụng sub query, thay thế bằng join

```
select
    b.FirstName
    ,b.LastName
from Sales.Customer a
    join Person.Person b on b.BusinessEntityID=a.PersonID

select
    (select FirstName from Person.Person b where b.BusinessEntityID=a.PersonID) --
b.FirstName
    ,(select LastName from Person.Person b where b.BusinessEntityID=a.PersonID) --b.LastName
from Sales.Customer a
    --join Person.Person b on b.BusinessEntityID=a.PersonID
```

- Tránh dùng CURSOR
  - Cursor không khác một vòng lặp thao tác tới từng bản ghi. Bản ghi đó sẽ bị lock cho tới khi được xử lý xong. Ảnh hưởng đến hiệu suất của chương trình rất chậm

2. Tối ưu hóa câu lệnh SQL

- Chọn loại dữ liệu thích hợp .
  - Ví dụ lưu chuỗi sử dụng loại varchar thay vì sử dụng loại Text. Khi muốn sử dụng loại Text, là khi bạn cần lưu dữ liệu lơn (nhiều hơn 8000 ký tự)
  - Tránh dùng nchar và nvarchar vì cả hai đều tăng bộ nhớ lên gấp đôi so với char và varchar.
- Tránh NULL đối với những trường mà đã cố định độ dài. Trong trường hợp yêu cầu là NULL hãy sử dụng một trường loại varchar với độ dài tùy biến thì vẫn lấy space ít hơn là NULL
- Đa số cột được chọn nên đặt trong non-clustered index. Những index nào không được dùng thì nên xóa đi.
- Tạo ra index trên những cột có giả trị là số thay vì là ký tự. Giá trị số sử dụng ít bộ nhớ hơn ký tự.
- Sử dụng WITH (NOLOCK) trong khi truy xuất dữ liệu từ bất kỳ một bảng nào.
- Dùng SET NOCOUNT ON
- Giữ transaction nhỏ nhất có thể vì transaction khóa việc xử lý dữ liệu bảng và có thể dẫn đến kết quả bị deadlocks.

3. Execution Plan

- Table Scan
- Clustered Index Scan
- Clustered Index Seek
- Non-clustered Index Seek
- RID Lookup
- Key Lookup
- Sort
- Aggregate - Stream Aggregate
- Compute Scalar

https://www.sqlshack.com/sql-server-execution-plan-operators-part-1/
https://www.sqlshack.com/sql-server-execution-plan-operators-part-2/
https://www.sqlshack.com/sql-server-execution-plan-operators-part-3/
https://www.sqlshack.com/sql-server-execution-plan-operators-part-4/

THANK YOU !

COLE.VN
Connecting knowledge

www.cole.vn