TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Báo Cáo Bài Tập Lớn

Thiết Kế Và Quản trị Cơ Sở Dữ Liệu

Đề Bài: *Introduction To Sql Tunning*

Giảng viên hướng dẫn : TS.Trần Việt Trung

*Nhóm sinh viên th*ực hiện *:*

Phùng Văn Dũng:20101292

Lê Đại Hoàn : 20091125

Phạm Quốc Khánh: 20101714

Lê Công Hưng: 20101667

Hà Nội

12/5/2015

MỤC LỤC

1.NHỮNG LÝ DO LÀM GIẢM HIỆU NĂNG CỦA SQL

- Bộ tối ưu hóa xác suất lỗi hoặc đã cũ
- Thiếu cấu trúc truy cập
- Chưa tối ưu lược đồ thực hiện truy vấn
- Xây dựng câu truy vấn sql còn nghèo nàn

1.1 **Bộ tối ưu hóa xác suất cũ hoặc lỗi.** Lược đồ thực thi truy vấn sql được sinh ra bởi chi phí tối ưu hóa cơ bản (CBO).Dựa trên CBO để chọn lược đồ thực hiện truy vấn hiệu quả nhất ,nó cần xác định thông tin về dữ liệu trên mỗi ổ và phân tán trên các bảng và index được tham chiếu trong các câu truy vấn. nếu bộ tối ưu hóa xác xuất không chính xác ,CBO có thể dễ dàng bị sai lệch và sinh ra các lược đồ thực hiện tối ưu hóa truy vấn không hiệu quả

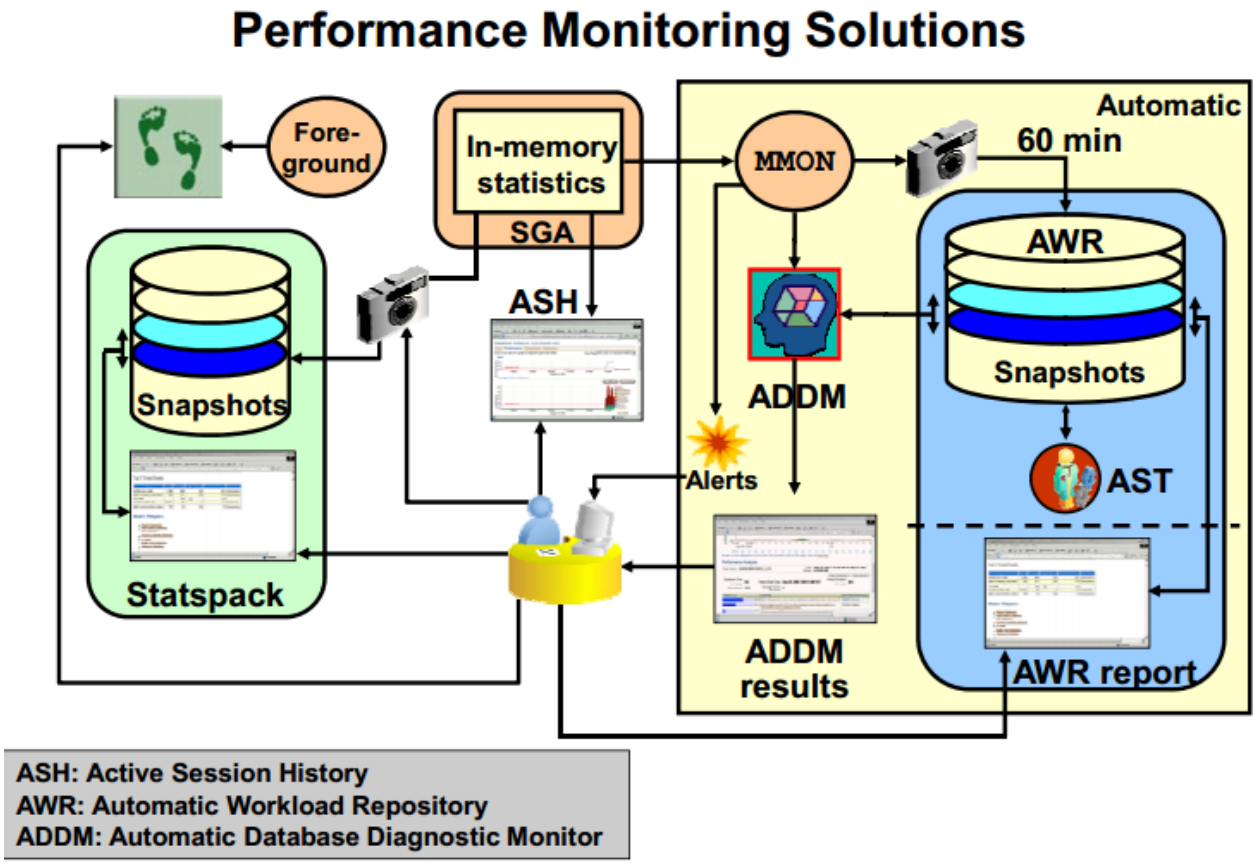
1.2 Thiếu cấu trúc truy cập .Không có các cấu trúc truy cập như index ,khung nhìn (view)..vv là lý do chung dẫn đến hiệu năng thực hiện câu truy vấn giảm .xây dựng các cấu trúc truy cập có thể cải thiện hiệu năng bằng gấp vài lần .

1.3. Chưa tối ưu lược đồ thực hiện truy vấn. Dựa trên chi phí tối ưu hóa cơ bản đôi khi có thể chọn 1 lược đồ thực hiện ko tối ưu.Điều này xảy ra nhiều nhất .bởi vì ước lượng không chính xác một số thuộc tính trên câu sql .cũng như chi phí hoặc các thuộc tính chọn

1.4. Xây dựng các câu truy vấn sql còn nghèo nàn. nếu câu truy vấn sql được thiết kế kém ,điều này có dẫn đến bộ tối ưu hóa cải thiện hiệu năng truy vấn không nhiều.1 lỗi điều kiện join có thể dẫn đến 1 phép tích đề các kết quả .hoặc sử dụng nhiều cấu trúc truy vấn đắt như UNION thay vì sử dụng UNION ALL

- Bốn nguyên nhân ở trên là lý do chính cho dẫn đến việc tối ưu hóa truy vấn kém và ảnh hưởng lớn đến hiệu năng ,cùng một vài lý do khác như vấn đề về kết nối các phần cứng liên quan , cũng như bộ nhớ ,vào ra ,cpu….vv

2. GIẢI PHÁP GIÁM SÁT HIỆU NĂNG



- Automatic Workload Repository(AWR) có nhiệm vụ thu thập ,xử lý ,duy trì việc thống kê hiệu năng cho việc phát hiện các vấn đề và tự tinh chỉnh theo mục đích .Dữ liệu này được lưu ở cả bộ nhớ và database.Tập hợp dữ liệu có thể được hiển thị trên cả reports và views .
- Active Session History (ASH) có nhiệm vụ cung cấp cơ chế lấy mẫu phiên hoạt trong tập thể hiện .Mỗi phiên hoạt động được lấy mẫu mỗi giây và lưu ở trong 1 bộ đệm vòng trong SGA
- Snapshots là một tập các dữ liệu trong lịch sử xác định theo mỗi giai đoạn .Nó được sử dụng cho việc so sánh hiệu năng bằng ADDM .AWR sẽ so sánh sự khác nhau giữa các snapshots để xác định câu truy vấn nào được nạp vào dựa

trên ảnh hưởng đối với hệ thống tải .điều này làm giảm số lượng câu truy vấn phải nạp theo thời gian

- Automatic Database Diagnostic Monior (ADDM) .Ngoài khả năng đã có là tự tinh chỉnh các vấn đề như ở ở những phiên bản trước chẳng hạn như statpack ,sql trace file ,và khung nhìn hiệu năng ,Oracle 10g giới thiệu 2 phương pháp giám sát database của bạn theo 2 loại là giám sát chủ động và giám sát bị động
- Giám sát chủ động .ADDM tự động xác định các nút thắt cổ chai bên trong oracle database ,thêm vào đó nó làm việc với các thành phần có khả năng quản lý .ADDM đưa ra các giải pháp về cho các tùy chọn có sẵn để có thể sửa các nút thắt cổ chai .Oracle 11 có thêm tính tính năng tự động xử lý tinh chỉnh sql bằng cách xác định vấn đề các câu truy vấn ,chạy sql tunning advisor trên chúng và thực thi kết quả sql profile để tự điều chỉnh câu truy vấn mà không cần sự can thiệp của người dùng .Việc này tự động sử dụng AUTOTASK Framework thông qua một tác vụ mới, tự động gọi tác vụ tinh chỉnh câu truy vấn chạy mỗi đêm theo mặc định

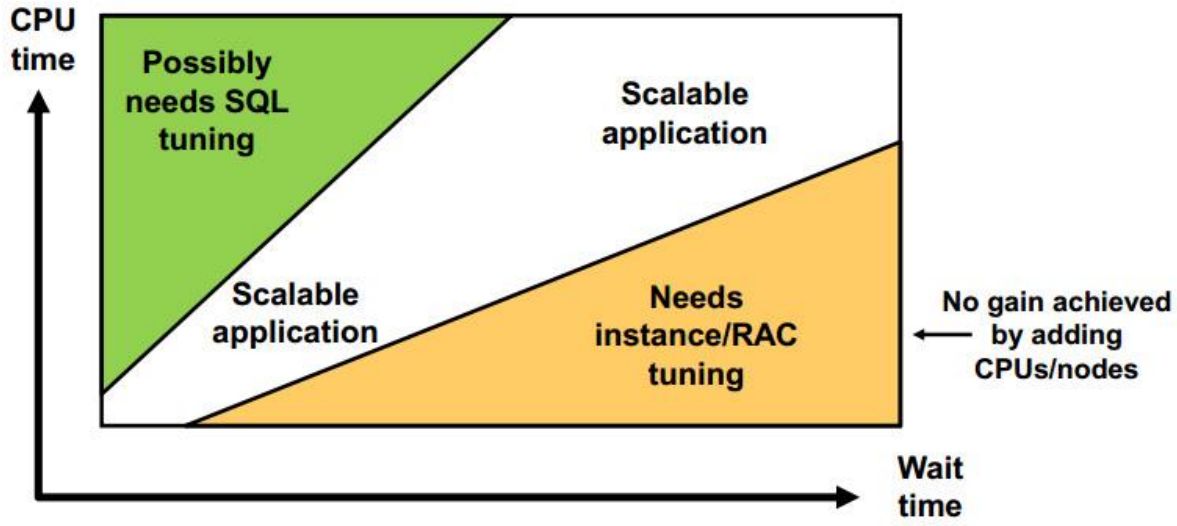
- Giám sát bị động.Server sinh ra các thông báo . Oracle Database có thể tự động phát hiện các trường hợp có vấn đề .để phản ứng lại những vấn đề được phát hiện , Database Oracle gửi thông điệp cho bạn cùng các giải pháp có thể để thực hiện việc tinh chỉnh Oracle Database có sức mạnh của các nguồn dữ liệu mới và khả năng báo cáo lại hiệu năng .Người quản lý doanh nghiệp cung cấp giao diện cho việc tích hợp quản lý hiệu năng ,nó sử dụng các nguồn dữ liệu liên quan.Sử dụng một phương pháp drill-down ,bạn có thể xác định các nút cổ chai chỉ với 1 vài click Các nguồn dữ liệu mới được đưa vào để nắm bắt các thông tin của database của bạn

3. CÁC THAO TÁC TINH CHỈNH SQL

Nhiều thao tác tinh chỉnh SQL có vai trò nền tảng cần được thực hiện thường xuyên. Bạn có thể thấy cách viết lại mệnh đề WHERE nhưng nó còn phụ thuộc vào index được tạo dựng. Danh sách các task sau đây cho bạn một cái nhìn tổng quan về một số tác vụ cần được thực hiện và cho bạn thấy được những thành phần cần quan tâm khi thực hiện tinh chỉnh SQL:

- Nhận diện mệnh đề high-load là một trong những bước quan trọng nhất cần được thực hiện. ADDM là công cụ điển hình cho tác vụ này.
- Mặc định, Oracle Database tự động tập hợp các chỉ số tối ưu. Công việc này được thực hiện trong chương trình bảo trì.
- Các chỉ số của hệ điều hành cung cấp thông tin về mức độ sử dụng và hiệu năng của các thành phần phần cứng cũng như hiệu năng của chính hệ điều hành.
- Việc tái xây dựng index thường có lợi ích giúp tang hiệu năng làm việc. Ví dụ như loại bỏ những index không được chọn để tăng tốc ngôn ngữ truy vấn dữ liệu, hoặc thêm cột vào index để cải thiện thao tác chọn.
- Có thể lưu trữ quá trình thực thi của những câu lệnh SQL bằng những chỉ số hay các baseline thực thi SQL.

ĐỘ PHỨC TẠP THỜI GIAN CPU VÀ THỜI GIAN TINH CHỈNH



Khả năng mở rộng là hệ thống có thể xử lý nhiều công việc hơn với tài nguyên được cân đối.

Khi tinh chỉnh hệ thống, so sánh giữa thời gian CPU và thời gian đợi của hệ thống là việc quan trọng bởi điều này xác định thời gian phản hồi có ích và thời gian chờ tài nguyên đang được trưng dụng bởi các tiến trình khác.

Một quy luật thường thấy là hệ thống có thời gian CPU ưu thế thường ít cần tinh chỉnh hơn sơ với hệ thống có thời gian đợi chiếm ưu thế. Mặt khác, mức độ chiếm dụng CPU cao có thể xảy ra do những câu lệnh SQL tồi.

Mặc dù tỉ lệ của thời gian CPU so với thời gian chờ có xu hướng giảm khi mà khả năng tải của hệ thống được cải thiện, sự leo dốc về thời gian đợi là vấn đề cần được quan tâm để hướng đến khả năng mở rộng tốt hơn.

Thêm nhiều CPU hơn vào một nút, hoặc thêm nhiều nút vào một cụm không giúp ích nhiều cho việc cạnh tranh. Trong khi một hệ thống mà tỉ lệ thời gian CPU giảm không đáng kể khi mà tăng tải có một khả năng mở rộng tốt hơn, và tỏ ra rất hữu ích nếu có nhu cầu thêm CPU hay thêm nút vào cụm Real Application Clusters (RAC) nếu cần thiết.

4. KHẢ NĂNG MỞ RỘNG CỦA HỆ THỐNG VỚI THIẾT KẾ, THỰC THI VÀ CẤU HÌNH CỦA ỨNG DỤNG

Thiết kế, triển khai và cấu hình ứng dụng tồi ảnh hưởng rất lớn đến khả năng mở rộng. Điều này dẫn tới:

- Thiết kế SQL và index tồi, làm cho cần nhiều đầu vào ra logic (I/O) cho cùng số kết quả (dòng) trả về.
- Giảm khả năng sẵn dụng vì các đối tượng dữ liệu mất nhiều thời gian để bảo trì.

Tuy nhiên, vấn đề không chỉ ở thiết kế, điểm yếu cũng có thể là ở việc triển khai mức vật lý của ứng dụng, ví dụ như sau:

- Hệ thống hoạt động với những câu lệnh SQL tồi do người dùng viết làm tăng chiếm dụng I/O.
- Không thường xuyên thực thi COMMITS hay ROLLBACKS có thể làm cho tài nguyên bị gián đoạn khi sử dụng.
- Kế hoạch thực thi thực tế khác với kế hoạch dùng trong kiểm thử.
- Các ứng dụng chiếm dụng nhiều bộ nhớ mà chưa giải phóng bộ nhớ có thể gây ra phân mảnh xấu đến bộ nhớ.
- Sử dụng bộ nhớ không hiệu quả gây tải cao lên các hệ thống bộ nhớ logic, làm giảm hiệu năng và khả năng sẵn dùng.

5. NHỮNG LỖI PHỔ BIẾN TRÊN CÁC HỆ THỐNG CỦA KHÁCH HÀNG

- **Quản trị kết nối tồi:** Ứng dụng kết nối và ngắt kết nối mỗi khi tương tác database. Vấn đề này thường xảy ra với middleware tồi trong các máy chủ ứng dụng. Nó ảnh hưởng đến 2 vấn đề lớn của hiệu năng hệ thống, điều không thể chấp nhận được.
- **Sử dụng không tốt những con trỏ và tài nguyên chia sẻ:** Không sử dụng con trỏ dẫn tới những bước đọc lặp lại. Nếu những biến gán không được dùng, có thể xảy ra việc đọc liên tục những mệnh đề SQL giống nhau. Điều này ảnh hướng lớn tới hiệu năng và không thể chấp nhận được. Dùng con trỏ với biến gán mà điều đó là cần thiết thực thi nhiều lần. Hãy cân nhắc về ứng dụng tạo SQL động.
- **Câu SQL tồi:** là SQL dùng nhiều tài nguyên vượt mức cho phép của ứng dụng. Có thể là truy vấn của hệ trợ giúp quyết định chạy quá 24h hay truy vấn của ứng dụng online chạy quá 1 phút. SQL tiêu tống tài nguyên cần được chú ý để có thể cải thiện khả năng xử lý. ADDM nhận diện high-load SQL và SQL Tuning Advisor có thể cung cấp những gợi ý để cải thiện.
- **Dùng những tham số khởi tạo không chính tắc:** Những biến này có thể được thực thi dựa vào gợi ý tồi hoặc giả định sai. Hầu hết các hệ thống cung cấp hiệu năng chấp nhận được mà chỉ sử dụng tập các tham số cơ bản. Thông thường những tính năng tối ưu tạo nên cách ứng phó tốt với vấn đề cần sự quan tâm đầu tư nhất định. Ví dụ schemas, chỉ số schema và cài đặt tối ưu có thể quản lý thành một nhóm để tạo sự ổn định hiệu năng.
- **Sai đầu vào database I/O:** nhiều site tổ chức cơ sở dữ liệu của họ một cách sơ sài qua những ổ đĩa khả dụng. Những site khác định nghĩa sai số đĩa do họ định nghĩa bằng vùng trống của đĩa mà không phải băng thông I/O.

- **Vấn đề Redo log setup:** Nhiều site chạy với những file redo log quá nhỏ. Điều này dẫn tới những checkpoints hệ thống liên tục gây tải cao lên bộ nhớ đệm và hệ thống I/O. Nếu chỉ có ít redo log, thông tin archive không được lưu trữ tốt, và database phải đợi quá trình xử lý archive để tiếp tục khả dụng.
- **Lạm dụng đánh nhãn:** đánh nhãn những data block vào bộ nhớ đệm do sự giản lược của phân đoạn undo nên thường được dùng trong ứng dụng với số lượng lớn user hoạt động và ít phần đoạn hoàn tác. Dùng Automatic Segment Space Management (ASSM) và công cụ quản trị hoàn tác tự động để khắc phục vấn đề này
- **Duyệt toàn bảng và quá lâu:** thường do thiết kế tồi của giao dịch, thiếu index hay SQL chưa tối ưu. Điều này tiêu tốn I/O và không thể chấp nhận được.
- **Số lượng lớn SQL đệ quy:** số lượng lớn SQL đệ quy thực thi bởi SYS chỉ tới những hoạt động quản lý vùng trống như mức độ phân bổ. Điều này không chấp nhận được và ảnh hưởng thời gian phản hồi tới người dùng. Dùng những bảng trống cục bộ để giảm SQL đệ quy do mức độ phân bổ. SQL đệ quy thực thi dưới user ID khác chỉ là SQL và PL/SQL, và không phải vấn đề khó khăn.
- **Lỗi triển khai và di chuyện dữ liệu:** Trong nhiều tình huống, một ứng dụng dùng quá nhiều tài nguyên bởi vì schema lưu giữ những bảng chưa được migrated từ môi trường phát triển hoặc từ sự triển khai trước đó. Ví dụ như việc thiếu index hay những chỉ số sai. Những lỗi này có thể dẫn tới hiệu năng tương tác người dùng rất tồi. Khi mà migrate ứng dụng đã biết hiệu năng, export các chỉ số schema để bảo trì tính ổn định của kế hoạch bằng cách dùng DBMS_STATS package.Tuy rằng những sự cố này không trực tiếp được ADDM phát hiện, nhưng ADDM highlight trong những high-load SQL

6. PHÁC ĐỒ TINH CHỈNH CHỦ ĐỘNG

- Simple design
- Data modeling
- Tables and indexes
- Using views
- Writing efficient SQL
- Cursor sharing
- Using bind variables

Tinh chỉnh thường liên quan tới khắc phục một vấn đề về hiệu năng. Tuy nhiên, tinh chỉnh nên là một phần của quy trình phát triển một ứng dụng, trong quá trình phân tích, thiết kế, lập trình, bàn giao và bảo trì. Giai đoạn tinh chỉnh thường kết thúc khi hệ thống được bàn giao. Tại thời điểm ấy, nó trở thành công việc bị động, mà trọng tâm là nhận diện và khắc phục.Dưới đây là một số vấn đề ảnh hưởng tới hiệu năng mà cần được chủ động tinh chỉnh.

7. TÍNH ĐƠN GIẢN TRONG THIẾT KẾ ỨNG DỤNG

Ứng dụng không khác gì các sản phẩm cần thiết kế khác trong đời sống. Nếu nó trông có vẻ ổn, thì nó ổn. Điều này cần được lưu tâm khi xây dựng các ứng dụng, tập trung vào một vài vấn đề sau đây:

- Nếu thiết kế bảng quá phức tạp để không ai có thể hiể đầy đủ về nó, đó là thiết kế tồi.
- Nếu những câu lệnh SQL quá dài và dường như không hể tối ưu trong thời gian thực thì đó là vấn đề tồi do thiết kế transaction hoặc thiết kế bảng.
- Nếu có nhiều index trong một bảng mà cùng một cột có sự lặp lại của index, đó chắc chắn là thiết kề tồi.

- Nếu truy vấn được xác nhận mà không có ràng buộc phù hợp (mệnh đề WHERE) để phàn hồi nhanh cho người dùng trực tuyến thì đó là thiết kế tồi trong giao diện người dùng hoặc transaction.

8. MÔ HÌNH HÓA DỮ LIỆU

Mô hình hóa dữ liệu có vai trò quan trọng liên quan tới thành công của thiết kế ứng dụng. Công việc này cần thực hiện nhanh chóng và sát thực với yêu cầu đặt ra. Nỗ lực tốt nhất để mô hình hóa những thực thể được tác động bởi hầu hết các giao dịch. Sử dụng các công cụ mô hình hóa có thể nhanh chóng tạo ra các định nghĩa schema và hữu dụng khi cần chế thử

Khái quát hóa dữ liệu tránh được sự dư thừa. Khi dữ liệu được khái quát, bạn có cái nhìn tường minh về các khóa và ràng buộc. Nhờ đó triển khai bước tiếp theo dễ dàng hơn: tạo bảng, ràng buộc và các index. Một mô hình hóa dữ liệu đạt tối ưu có nghĩa là các truy vấn được viết dễ dàng và hiệu quả hơn.

9. TABLE DESIGN – THIẾT KẾ BẢNG

Thiết kế bảng là sự cân bằng quan trọng giữa sự mềm dẻo và hiệu năng của các giao dịch chính. Để giữ cho database mềm dẻo và có thể đáp ứng những yêu cầu làm việc ngoài dự tính thì thiết kế bảng cần bám sát vào mô hình dữ liệu, và nó cần được khái quát hóa từ 3 dạng mô hình gần nhất. Đôi khi, những giao dịch quan trọng có thể yêu cầu tập không chính tắc phục vụ yêu cầu hiệu năng.

Sử dụng các tính năng cung cấp bởi Oracke Database để đơn giản hóa thiết kế bảng nhằm tăng hiệu năng: lưu trữ các bảng cần kết nối vào cụm cluster, thêm cột chung và những giá trị hợp nhất, và dùng góc nhìn nguyên tố hay phân đoạn bảng. Hơn nữa, tạo ràng uộc kiểm tra và các cột với các giá trị mặc định để lọc các dữ liệu nhiễu.

Thiết kế cần tập trung vào các bảng có vai trò quan trọng trong giao dịch để đạt được hiệu năng tốt vì đây là vùng được sử dụng nhiều nhất. Với những bảng còn lại, tóm lược thiết kế có thể triển khai để giúp cho phát triển ứng dụng nhanh hơn, tuy nhiên nếu đó trở thành vấn đề hiệu năng trong quá trình kiểm thử thì cần khắc phục và thiết kế lại.

10. THIẾT KẾ INDEX

Đây cũng là công việc có vai trò quan trọng hướng tiến trình. Tuy nhiên có thể bắt đầu bằng việc xây dựng index chứa khóa ngoài để giảm thời gian phản hồi của việc kết nối khóa chính và khóa ngoài, tạo index dựa trên dữ liệu thường xuyên cần truy cập như là tên người. Khóa chính và unique key thường được tự động gán index trừ những ràng buộc DISABLE VALIDATE và DISABLE NOVALIDATE RELY. Bởi ứng dụng và kiểm thử thực hiện lên quan tới kích thước thực của dữ liệu, nhiều câu truy vấn cần được cải thiện hiệu năng nên việc xây dựng index tốt là cần thiết.

Phụ thêm cột vào index hay sử dụng bảng tổ chức index

Một trong những cách dễ nhất để tăng tốc 1 truy vấn là làm giảm số đầu vào ra I/O logic bằng cách loại bỏ việc duyệt bảng trong thực thi. Điều này có nghĩa là tạo một index qua tất cả các cột của bảng được duyệt bởi truy vấn. Những cột này trong danh sách cột được SELECT, mệnh đề WHERE và bất cứ yêu cầu kết nối hay sắp xếp cột. Kỹ huật này thường hữu dụng trong việc tăng tốc thời gian phản hồi của ứng dụng trực tuyến khi sự tiêu tốn I/O được giảm xuống. Đây là cách áp dụng hữu hiệu khi kiểm thử ứng dụng với dữ liệu có size cố định trong lần đầu. Cách chủ động nhất của kỹ thuật này là xây dựng bảng quản lý index (IOT).

11. USING VIEWS

View (giao diện) giúp nhanh chóng và đơn giản hóa thiết kế ứng dụng. Một định nghĩa view đơn giản có thể đặc tả độ phức tạp mô hình dữ liệu từ người lập trình – người mà ưu tiên hơn vấn đề truy vấn, hiển thị, thu thập và lưu trữ dữ liệu. View thường được dùng cung cấp sự đơn giản hóa ràng buộc truy cập mức hàng và cột.

Tuy nhiên, do view cung cấp giao diện lập trình thuần túy, chúng có thể dẫn tới những câu truy vấn con rất ngốn tài nguyên khi can thiệp sâu. Điều tệ hại dùng view là tạo ra sự kết nối các view chỉ tới view khác và lại trỏ tới các view khác nữa. Trong nhiều trường hợp, developer có thể hài long với câu truy vấn trực tiếp không dùng view. Bởi với thuộc tính vốn có của chúng, view thường gây khó khăn cho việc tối ưu thực thi.

12. HIỆU QUẢ THỰC THI SQL

Ứng dụng thiết kế thực thi SQL hiệu quả cần đáp ứng các yêu cầu sau

- **Quản trị tốt kết nối database**: Kết nối database là một thao tác không thể mở rộng. Vì vậy số kết nối tương tranh cần giảm thiểu nhất có thể. Một hệ thống đơn giản khi mà một người dùng kết nối tại thời điểm khởi chạy ứng dụng là lý tưởng. Tuy nhiên, trong ứng dụng web base hay ứng dụng đa luồng, máy chủ cần phải phân kênh kết nối dữ liệu tới các người dùng, điều này trở nên khó khăn. Với những dạng ứng dụng như vậy, hiệu quả thiết kế phải đáp ứng được các kết nối được sắp xếp và không tái lập cho mỗi yêu cầu người dùng
- **Sử dụng và quản lý tốt con trỏ:** Duy trì kết nối người dùng quan trọng tương đương với việc giảm thiểu hành động phân tích trên hệ thống. Parsing là tiến trình thông dịch SQL và tạo ra thực thi. Tiến trình này có nhiều giai đoạn bao

gồm kiểm tra cú pháp, kiểm tra bảo mật, sinh kế hoạch thực thi và tải cấu trúc chia sẻ vào bộ nhớ

- **Hard parsing**(parsing cứng): Một câu SQL được gửi tới lần đầu và không tìm thấy trong bộ nhớ chia sẻ. Hard parse tiêu tốn tài nguyên và rất khó mở rộng bởi nó làm tất cả các tác vụ của một parse
- **Soft parsing (parsing mềm):** Một câu SQL được gửi tới trong lần đầu tiên và có bản ghi trùng hợp trong bộ nhớ chia sẻ. Có thể là kết quả của lần thực thi trước bởi người dùng khác. Câu SQL được chia sẻ, và hữu ích cho hiệu năng. Tuy nhiên soft parses không thực suự tốt vì nó vẫn kiểm tra cú pháp và bảo mật và tốn tài nguyên hệ thống.

Bởi vì cần giảm thiểu nhất có thể parsing nên developer cần thiết kế ứng dụng của họ để parse SQL 1 lần và thực thi nó nhiều lần. Điều này có được nhờ cursors.

Developer cũng cần đảm bảo rằng các câu SQL được chia sẻ trong bộ nhớ cho phép. Để làm điều này cần các biến gán đại diện cho các phần của truy vấn có thay đổi qua các lần thực thi. Nếu không câu SQL có vẻ được parse 1 lần và không được dùng lại lần nào.

13.VIẾT SQL ĐỂ CHIA SẺ CON TRỎ

- Tạo mã chung sử dụng cho những các vấn đề sau:
  + Các stored procedure và các gói
  + Các trigger
  + Một vài thói quen và thủ tục khác

- Viết theo đúng định dạng chuẩn (giúp việc đọc dễ dàng hơn) trong:
  + Trường hợp (case)
  + Khoảng trắng
  + Comment
  + Đối tượng tham chiếu
  + Các biến ràng buộc
- Viết SQL đề chia sẻ con trỏ

  Ứng dụng có thể được chia sẻ con trỏ khi các mã được viết bằng những cách giống nhau (cái mà cho phép hệ thống nhận ra rằng 2 câu giống nhau vì vậy có thể được chia sẻ), ngay cả khi bạn sử dụng một số tham số khởi tạo đặc biệt, chẳng hạn như CURSOR_SHARING (sẽ thảo luận trong bài "Các biến ràng buộc"). Bạn nên viết coding convention theo các đoạn sql mẫu, các giao diện gọi Oracle.
- Sử dụng các mã chia sẻ chung
  + Viết và lưu trữ các thủ tục có thể được chia sẻ giữa các ứng dụng.
  + Sử dụng trigger.
  + Viết các thư viện, các thủ tục trong các môi trường khác nhau.
- Viết đúng định dạng chuẩn
  + Phát triển định dạng chuẩn cho toàn bộ các báo cáo, bao gồm mã PL/SQL.
  + Xây dựng quy tắc cho các kí tự hoa, thường.
  + Xây dựng quy tắc cho việc sử dụng các khoảng trống trắng (dấy cách, tabs, return).
  + Xây dựng quy tắc cho việc sử dụng comment.
  + Sử dụng tên tương tự cho các đối tượng csdl giống nhau. Nếu có thể, sử dụng tiền tố cho từng đối tượng.

Checklist để đảm bảo hiệu suất

- Thiết lập số lượng tối thiểu của các tham số khởi tạo, lý tưởng nhất là các tham số khởi tạo nên để ở mặc định, mọi điều chỉnh cho việc tối ưu nên thực hiện khi hệ thống đang được load. Thiết lập kiểu lưu trữ, đánh chỉ mục thích hợp cho từng bảng.
- Xác nhận rằng mọi câu sql là tối ưu, và nắm rõ được cách sử dụng tài nguyên của chúng.
- Xác nhận rằng các middleware có hiệu quả trong việc kết nối csdl, không nên đăng nhập, đăng xuất liên tục.
- Xác nhận rằng câu lệnh sql sử dụng con trỏ một cách hiệu quả, mỗi câu lệnh sql nên được phân tích một lần, và sau đó thực hiện nhiều lần. Khi các biến ràng buộc không được sử dụng đúng cách, và mệnh đề WHERE được gửi dưới dạng xâu.
- Đảm bảo rằng các đối tượng như bảng, chỉ mục, trình tự, gói, thủ tục, đối tượng, từ đồng nghĩa đã được chuyển đổi từ môi trường đến database.

Oracle SQL là gì ?

- Là một tool có giao diện đồ họa miễn phí được thiết kế để cải thiện năng suất của bạn và đơn giản hóa việc phát triển csdl. Chỉ với một vài cú nhấp chuột, bạn có thể dễ dàng tạo ra và duy trì các thủ tục lưu trữ, báo cáo, kiểm tra SQL. Chúng gồm tập các công cụ trực quan cho phát triển csdl, đơn giản hóa các nhiệm vụ sau:
  + Quản lý các đối tượng csdl
  + Thực hiện các câu lệnh sql và các kịch bản.
  + Chỉnh sửa và gỡ lỗi các câu lệnh SQL
  + Tạo báo cáo.

- Oracle SQL *Plus: là một giao diện dòng lệnh cho phép bạn gửi các câu lệnh SQL và khối PL/SQL để thực thi và nhận kết quả trong một ứng dụng và một cửa sổ lệnh. SQL *Plus:

  + đi cùng với các sơ sở dữ liệu

  + truy cập từ một biểu tượng hoặc một dòng lệnh
- Khi code chương trình con PL/SQL sử dụng SQL *Plus cần nhớ những điều sau:

  + tạo chương trình con, sử dụng: CREATE SQL

  + chạy chương trình con, sử dụng: EXCUTE

  + nếu bạn sử dụng gói thủ tục DBMS_OUTPUT để in đoạn text ra màn hình, trước tiên bạn cần sử dụng lệnh SET SERVEROUTPUT trong phiên làm việc của bạn.
- Để khởi động SQL *Plus trong môi trường Linux, mở một cửa sổ terminal và nhập lệnh sqlplus.

TÀI LIỆU THAM KHẢO

1. Oracle Database 11g: SQLTuning Workshop
2. Slide thiết kế và quản trị hệ csdl –Trần Việt Trung

Oracle tunning steps- http://www.dba-oracle.com/art_sql_tune.htm