**I. Thông tin nhóm:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ tên** | **MSSV** | **Ghi chú** |
| 1 | Hà Thị Phương Thảo | 0912430 |  |
| 2 | Nguyễn Thị Thanh Thảo | 0912431 |  |
| 3 | Trương Nguyễn Thủy Tiên | 0912463 |  |
| 4 | Nguyễn Văn Tiến | 0912469 | Nhóm trưởng |

**II. Bảng phân công công việc:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Nội dung công viẹc** | **Thành viên 1**  **0912430** | **Thành viên 2**  **0912431** | **Thành viên 3**  **0912463** | **Thành viên 4**  **0912469** |
| 1 | Dịch tài liệu |  |  |  |  |
| 2 | Tổng hợp kết quả, viết tài liệu |  |  |  |  |

Mục lục

[I. Thiết kế dữ liệu mức vật lý trong cơ sở dữ liệu quan hệ: 3](#_Toc308852041)

[1. Các yếu tố ảnh hưởng đến thiết kế dữ liệu mức vật lý: 3](#_Toc308852042)

[*a.* *Phân tích các câu truy vấn và giao tác cơ sở dữ liệu:* 3](#_Toc308852043)

[*b.* *Phân tích tần suất của các câu truy vấn và giao tác được gọi;* 3](#_Toc308852044)

[*c.* *Phân tích ràng buộc thời gian của truy vấn và giao tác:* 4](#_Toc308852045)

[*d.* *Phân tích tần suất kỳ vọng của thao tác cập nhật:* 4](#_Toc308852046)

[*e.* *Phân tích ràng buộc đơn trị trên các thuộc tính:* 4](#_Toc308852047)

[2. Những hướng giải quyết trong thiết kế cơ sở dữ liệu mức vật lý: 4](#_Toc308852048)

[*a.* *Thiết kế index:* 4](#_Toc308852049)

[*b.* *Denormalization: một cách tăng tốc truy vấn* 5](#_Toc308852050)

[II. Khái quát về tinh chỉnh cơ sở dữ liệu trong hệ thống quan hệ: 5](#_Toc308852051)

[1. Điều chỉnh index: 6](#_Toc308852052)

[2. Điều chỉnh thiết kế cơ sở dữ liệu 7](#_Toc308852053)

[3. Điều chỉnh truy vấn 7](#_Toc308852054)

[4. Các hướng dẫn tinh chỉnh truy vấn: 9](#_Toc308852055)

**THIẾT KẾ DỮ LIỆU VẬT LÝ THỰC TIỄN VÀ TINH CHỈNH**

# Thiết kế dữ liệu mức vật lý trong cơ sở dữ liệu quan hệ:

## Các yếu tố ảnh hưởng đến thiết kế dữ liệu mức vật lý:

Đối với một lược đồ quan niệm cho sẵn, có nhiều lựa chọn thiết kế vật lý trong một DBMS cho trước. Sẽ không thể quyết định một thiết kế có đầy đủ ý nghĩa, và phân tích hiệu suất cho đến khi ta biết các câu truy vấn, giao tác và ứng dụng chạy trên cơ sở dữ liệu đó. Chúng ta phải phân tích các ứng dụng, những truy vấn mong muốn, bất kỳ ràng buộc nào trong thực thi và tần suất của các phép toán cập nhật

### *Phân tích các câu truy vấn và giao tác cơ sở dữ liệu:*

Trước khi đảm trách thiết kế dữ liệu mức vật lý, chúng ta phải có ý tưởng về mục đích sử dụng của cơ sở dữ liệu bằng cách định nghĩa các truy vấn và giao tác chúng ta muốn thực hiện trên cơ sở dữ liệu ở dạng cao hơn. Với mỗi câu truy vấn, chúng ta nên xác định những điều dưới đây:

1. File được câu truy vấn truy xuất
2. Thuộc tính mà điều kiện chọn của câu truy vấn thực hiện trên đó
3. Thuộc tính trên đó các điều kiện kết hay điều kiện để liên kết nhiều bảng hoặc đối tượng của truy vấn được xác định
4. Thuộc tính được truy vấn lấy giá trị

Những thuộc tính được liệt kê ở mục ii, iii bên trên là ứng viên cho định nghĩa của cấu trúc truy xuất. Với mỗi giao tác, ta nên xác định các điều sau:

1. File sẽ được cập nhật
2. Loại phép toán trên mỗi file (insert, update hay delete)
3. Thuộc tính trên đó các điều kiện delete hoặc update xác định
4. Thuộc tính có giá trị bị thay đổi khi thực hiện cập nhật

Tương tự, những thuộc tính trong mục iii là ứng viên trong cấu trúc truy xuất. Mặt khác những thuộc tính ở mục iv là ứng viên tránh cấu trúc truy xuất, vì khi điều chỉnh chúng đòi hỏi cập nhật lại cấu trúc truy xuất.

### *Phân tích tần suất của các câu truy vấn và giao tác được gọi;*

Bên cạnh việc xác định các đặc điểm của truy vấn và giao tác sẽ xảy ra, chúng ta phải xem xét tỉ lệ sẽ được gọi. Thông tin tần suất này, cùng với thông tin thuộc tính được thu thập ở mỗi truy vấn và giao tác, được dùng để biên soạn một danh sách tích lũy tần suất sử dụng của các truy vấn và giao tác. Điều này được diễn giải như tần suất sử dụng của mỗi thuộc tính trong mỗi file như thuộc tính chọn hoặc thuộc tính kết, trên tất cả truy vấn và giao tác.

### *Phân tích ràng buộc thời gian của truy vấn và giao tác:*

Một vài truy vấn và giao tác có những ràng buộc hiệu suất nghiêm ngặt. Ví dụ một giao tác có ràng buộc thực thi trong vòng 5s trên 95% cơ hội khi nó được gọi lên và không nên lâu hơn 20s. Thuộc tính chọn được sử dụng bởi câu truy vấn và giao tác với ràng buộc thời gian trở thành ứng viên có độ ưu tiên cao với cấu trúc truy xuất chủ yếu

### *Phân tích tần suất kỳ vọng của thao tác cập nhật:*

Số lượng tối thiểu của con đường truy xuất phải được xác định với mỗi file thường được cập nhật vì việc cập nhật con đường truy xuất sẽ làm giảm tốc tộ thao tác cập nhật

### *Phân tích ràng buộc đơn trị trên các thuộc tính:*

Con đường truy xuất cần được xác định trên tất cả thuộc tính khóa ứng viên – hoặc tập các thuộc tính - cả khóa chính và ràng buộc duy nhất.

## Những hướng giải quyết trong thiết kế cơ sở dữ liệu mức vật lý:

### *Thiết kế index:*

Các thuộc tính có giá trị được yêu cầu trong điều kiện bằng hoặc điều kiện miền giá trị (phép chọn) và những thuộc tính là khóa hoặc tham gia vào phép kết đều cần con đường truy xuất.

Hiệu suất truy vấn phần lớn phụ thuộc vào những index hoặc sơ đồ hash tồn tại để xúc tiến xử lí phép chọn và phép kết. Mặt khác, trong khi insert, delete hoặc update, sự tồn tại của index sẽ được tính vào chi phí. Chi phí này phải được chứng minh tăng thêm hiệu quả thực hiện các truy vấn và giao tác

Những quyết định về thiết kế dữ liệu mức vật lý cho index chia theo các mục như sau:

1. ***Nên hay không cài đặt index trên một thuộc tính*** : Thuộc tính phải là khóa, hoặc có các câu query sử dụng thuộc tính trong điều kiện chọn (bằng hoặc miền giá trị) hoặc trong phép kết. Một yếu tố thiên về cài đặt nhiều index là một số truy vấn có thể được xử lý bằng cách quét index mà không cần truy xuất dữ liệu
2. ***Cài đặt index trên thuộc tính nào:*** một index có thể được xây dựng trên một hoặc nhiều thuộc tính. Nếu nhiều thuộc tính từ một quan hệ có liên quan với nhau trong một số truy vấn, một index đa thuộc tính được đảm bảo. Thứ tự của thuộc tính trong một index đa thuộc tính phải tương ứng với câu truy vấn.
3. ***Nên cài đặt clustered index hay không:*** hầu hết index một thuộc tính trên bảng là khóa chính hoặc cluster index, bởi vì điều này nghĩa là file có thể được sắp xếp vật lý trên thuộc tính đó. Trong hầu hết RDBMS, điều này được xác định bằng từ khóa CLUSTER. Nếu một bảng đòi hỏi nhiều index, quyết định cái nào là cluster index tùy thuộc vào có cần thiết giữ thứ tự sắp xếp của bảng trên thuộc tính đó hay không. Truy vấn miền giá trị được lợi nhiều từ cluster. Nếu vài thuộc tính yêu cầu truy vấn theo miền, những lợi ích liên quan phải được đánh giá trước khi quyết định cài cluster trên thuộc tính nào. Nếu môt truy vấn được đáp ứng chỉ bằng tìm kiếm index (không lấy dữ liệu ra), thì index tương ứng không nên là cluster, vì lợi ích chủ yếu của cluster index đạt được khi lấy dữ liệu của chúng ra.
4. ***Nên sử dụng hash index trên cây index hay không:*** Về tổng thể, RDBMS sử dụng cây B+ cho index. Tuy nhiên, ISAM và hash index còn được cung cấp trong một số hệ thống. Cây B+ hỗ trợ truy vấn bằng và miền giá trị trên những thuộc tính được xem là khóa tìm kiếm. Hash index làm việc tốt trong điều kiện bằng, cụ thể là trong phép kết khi tìm một record trùng khớp
5. ***Nên sử dụng một hàm băm động cho file hay không:*** Đối với file khả biến (dễ thay đổi) – nghĩa là dãn nở và co lại liên tục – một trong những sơ đồ băm động sẽ được bàn trong chương 13.9 sẽ thích hợp. Hiện tại thì chungs không được dùng trong hầu hết các RDBMS thương mại

### *Denormalization: một cách tăng tốc truy vấn*

Đôi khi việc chuẩn hóa dữ liệu được bỏ qua để đạt tốc độ nhanh hơn trong thực thi các câu truy vấn và giao tác thường dùng. Quá trình lưu trữ thiết kế logic ở dạng chuẩn yếu hơn như 2NF, 1NF được gọi là ***denormalization***. Điển hình là, người thiết kế thêm vào bảng các thuộc tính cần thiết đáp ứng cho yêu cầu truy vấn hoặc sản sinh báo cáo vì thế có thể tránh việc kết bảng. Điều này sẽ đưa vào bảng các phụ thuộc hàm không đầy đủ hoặc phụ thuộc hàm bắt, cầu, do đó sẽ tạo ra các vấn đề trùng lắp

Những dạng khác của phản chuẩn hóa bao gồm lưu trữ các bảng đặc biệt để duy trì phụ thuộc hàm ban đầu mất đi khi phản chuẩn hóa BCNF.

# Khái quát về tinh chỉnh cơ sở dữ liệu trong hệ thống quan hệ:

Sau khi một cơ sở dữ liệu được triển khai và đang trong thực thi, việc sử dụng thực tế của các ứng dụng, giao tác, truy vấn và view sẽ cho để lộ các yếu tố và vấn đề không được tính đến trong quá trình thiết kế dữ liệu ban đầu. Do đó sẽ cần thiết điều chỉnh hoặc duyệt lại thiết kế dữ liệu vật lý thường xuyên. Mục đích điều chỉnh là để:

* Giúp ứng dụng chạy nhanh hơn
* Giảm thời gian đáp ứng của truy vấn / giao tác
* Cải thiện hiệu suất tổng thể của giao tác

DBMS có thể thu thập các thông tin thống kê sau:

* Kích thước của các bảng độc lập
* Số giá trị phân biệt trong một cột
* Số lần một truy vấn hoặc giao tác được thực thi trong một khoảng thời gian.
* Số thời gian cần thiết cho các pha khác nhau của xử lí truy vấn và giao tác (đối với một tập các câu truy vấn hoặc giao tác)

Những thông tin khác đạt được từ việc điều chỉnh hoạt động và xử lí của hệ thống cơ sở dữ liệu bao gồm:

* Thống kê lưu trữ: dữ liệu về việc cấp phát lưu trữ vào các tablespace, indexspace và các port đệm.
* Thống kê hiệu suất I/O và thiết bị: tổng hoạt động đọc / viết trên phạm vi đĩa và điểm tới hạn của đĩa.
* Thống kê xử lý truy vấn và giao tác: số lần thực thi câu truy vấn và giao tác, số lần tối ưu khi tối ưu hóa câu truy vấn
* Thống kê liên quan lock/log: tỉ lệ phát hiện ra nhiều loại lock, tỉ lệ năng suất giao tác và hoạt động ghi nhật ký record.
* Thống kê index: số level trong index, số trang lá (leaf page) không kế tiếp

Điều chỉnh một cơ sở dữ liệu liên quan đến giải quyết các vấn đề sau:

* Làm thế nào để tránh tranh chấp lock quá mức, từ đó tăng xử lý đồng thời giữa các giao tác
* Làm thế nào để giảm tối thiểu chi phí logging và những kết xuất dữ liệu không cần thiết
* Làm thế nào để tối ưu hóa kích thước buffer và lập lịch cho các xử lý
* Làm thế nào để cấp phát tài nguyên như đĩa, RAM và tiến trình cho việc sử dụng hiệu quả nhất

Hầu hết các vấn đề nói trên đều có thể giải quyết được bằng cách thiết lập các tham số vật lý thích hợp trong DBMS, thay đổi cấu hình của thiết bị, thay đổi tham số hệ điều hành và các hoạt động tương tự.

## Điều chỉnh index:

Các chọn lựa khởi tạo index có thể được xem xét với các lí do sau:

* Câu truy vấn nhất định sẽ mất nhiều thời gian thực thi nếu thiếu index
* Chỉ mục nhất định sẽ không được sử dụng ở tất cả
* Chỉ mục nhất định có thể gây ra chi phí quá mức vì index được cài trên một thuộc tính thay đổi thường xuyên.

Hầu hết các DBMS đều có các dòng lệnh hoặc tiện ích theo dõi, được các DBA sử dụng để yêu cầu hệ thống cho biêt cách một câu truy vấn được thực hiện như thế nào – thao tác nào được thực hiện theo thứ tự nào và cấu trúc truy vấn thứ cấp được sử dụng. Bằng cách phân tích, lược đồ thực thi, có thể xác định nguyên nhân xảy ra các vấn đề trên.

Mục tiêu của điều chỉnh là đánh giá động yêu cầu, mà đôi khi dạo động theo từng thời vụ hoặc suốt các khoảng thời gian khác nhau của tháng hay tuần, và để tổ chức lại index nhằm đạt được hiệu suất tổng thể cao nhất. Loại bỏ và xây dựng index mới là một chi phí có thể điều chỉnh trong phạm vi cải thiện hiệu suất. Cập nhật một bảng về tổng thể được trì hoãn khi một index bị loại bỏ hay tạo mới, vấn đề mất mát của dịch vụ phải được tính đến. Bên cạnh việc loại bỏ hoặc tạo index, và chuyển từ nonclustered index sang clustered index và ngược lạ, thì xây dựng lại index có thể cải thiện được hiệu suất. Hầu hết RDBMS sử dụng cây B+ đối với index. Nếu có nhiều thao tác xóa trên khóa index, các trang index có thể chứa nhiều không gian lãng phí, vốn có thể bù lại khi xây dựng lại thao tác. Một cách tương tự, quá nhiều thao tác insert có thể gây ra quá tải trên index cluster, điều này sẽ làm ảnh hưởng đến hiệu suất. Xây dựng lại cluster index dẫn đến việc tổ chức lại toàn bộ thứ tự của bảng trên khóa đó.

## Điều chỉnh thiết kế cơ sở dữ liệu

Chúng ta chỉ ra rằng thiết kế toàn bộ dữ liệu phải được xác định bằng yêu cầu xử lý cũng như yêu cầu dữ liệu. Nếu yêu cầu xử lý thay đổi thường xuyên, thiết kế cần phải đáp ứng bằng cách thay đổi lược đồ quan niệm nếu cần và phản ánh những thay đổi này trong lược đồ logic và thiết kế vật lý. Những thay đổi đó có thể gồm:

Tồn tại các bảng có thể kết (phản chuẩn hóa) vì các thuộc tính nhất định từ 2 hoặc nhiều bảng thường được sử dụng cùng nhau. Điều này làm giảm mức độ chuẩn hóa từ BCNF về 3NF, 2NF, 1NF

Đối với các tập hợp các bảng cho trước, có thể có nhiều thiết kế thay thế nhau, tất cả đều đạt dạng chuẩn BCNF hoặc 3NF.

* Một quan hệ R (K, A, B, C, D, …) – với K là một tập các thuộc tính khóa – nghĩa là trong BCNF có thể lưu nhiều bảng cũng có chuẩn BCNF, ví dụ R1(K, A, B), R2 ( K, C, D)… bằng cách lặp lại khóa K trong mỗi bảng. Mỗi tập hợp các bảng của thuộc tính có thể truy xuất cùng nhau. Đây được gọi là chẻ dọc (phân chia dọc)
* Thuộc tính từ một bảng có thể được lặp lại trong bảng khác thậm chí gây ra dư thừa dữ liệu và những vấn đề tiềm ẩn
* Chỉ có phân chia dọc sẽ phân bảng thành nhiều bảng nhỏ, phân chia ngang sẽ cắt ngang bảng và lưu trữ chúng trong những bảng riêng biệt.

Những loại điều chỉnh trên thiết kế để đạt khối lượng truy vấn và giao tác cao mà không hi sinh dạng chuẩn, là thông thường trong thực tế

## Điều chỉnh truy vấn

Có 2 dấu hiệu cho biết cần thiết điều chỉnh truy vấn:

* Một truy vấn thực hiện quá nhiều truy xuất đĩa (ví dụ một truy vấn trùng khớp chính xác quét toàn bộ bảng)
* Lược đồ truy vấn thể hiện các index thích hợp không được sử dụng

Một vài biểu hiện của tình huống thúc đẩy điều chỉnh truy vấn:

1. Nhiều bộ tối ưu truy vấn không sử dụng index khi xuất hiện của biểu thức số học, so sánh số của thuộc tính của các kích thước và tính chính xác khác nhau, so sánh **NULL** và so sánh chuỗi con
2. Index thường không được dùng cho truy vấn lồng
3. Một vài từ khóa **DISTINCTS** có thể dư thừa và tránh sử dụng mà không làm thay đổi kết quả. **DISTINCT** thường gây ra thao tác sắp xếp và phải tránh càng nhiều càng tốt
4. Việc sử dụng không cần thiết các bảng kết quả tạm thời có thể tránh được bằng cách rút gọn các câu truy vấn thành các truy vấn đơn lé trừ khi quan hệ tạm thời cần thiết cho xử lý ngay lập tức
5. Trong một vài trường hợp liên quan đến việc sử dụng truy vấn tương quan, tính tạm thời rất có ích. Hãy xem xét câu truy vấn sau:

**SELECT SSN**

**FROM EMPLOYEE E**

**WHERE SALARY = SELECT MAX (SALARY)**

**FROM EMPLOYEE AS M**

**WHERE M.DNO = E.DNO;**

Điều này có một nguy hiểm tiềm tàng trong việc tìm kiếm tất cả EMPLOYEE bảng M đối với mỗi bộ từ EMPLOYEE bảng E. Để giúp nó hiệu quả hơn, chúng ta có thể tách ra thành 2 câu truy vấn như sau:

**SELECT MAX (SALARY) AS HIGHSALARY, DNO INTO TEMP**

**FROM EMPLOYEE**

**GROUP BY DNO;**

**SELECT SSN FRO EMPLOYEE, TEMP WHERE SALARY = HIGHSALARY AND EMPLOYEE.DNO = TEMP.DNO;**

1. Nếu có nhiều lựa chọn cho điều kiện kết hãy chọn một điều kiện sử dụng cluster index và tránh điều kiện sử dụng so sánh chuỗi.
2. Một đặc tính với tối ưu hóa truy vấn là thứ tự bảng trong mệnh đề **FROM** có thể ảnh hưởng xử lí phép kết. Nếu đó là vấn đề, thì nên chuyển đổi thứ tự sao cho quan hệ nhỏ hơn được xem xét và quan hệ lớn hơn được sử dụng với index thích hợp
3. Một vài bộ tối ưu truy vấn hoạt động tệ hơn trên truy vấn khi so sánh với phần còn lại không có truy vấn tương ứng. Có 4 loại truy vấn lồng:

* Truy vấn phụ không tương quan với tổ hợptrong câu truy vấn nội bộ
* Truy vấn phụ không tương quan không tổ hợp
* Truy vấn phụ tương quan với tổ hơp trong truy vấn nội bộ
* Truy vấn phụ tương quan không tổ hợp

Khác với các loại còn lại, loại 1 đại diện cho không có vấn đề, vì hầu hết các bộ tối ưu chỉ đánh giá truy vấn nội bộ 1 lần. Tuy nhiên đối với loại 2, hầu hết các bộ truy vấn có thể không sử dụng index

1. Cuối cùng nhiều ứng dụng dựa trên view định nghĩa dữ liệu liên quan tới chính ứng dụng. Đôi khi, những khung nhìn này trở thành tàn phá, vì nhiều truy vấn có thể đặt trực tiếp trên các bảng cơ bản, hơn là đi vòng qua view được định nghĩa bằng phép kết

## Các hướng dẫn tinh chỉnh truy vấn:

* Một truy vấn với nhiều điều kiện chọn lọc được liên kết thông qua **OR** có thể ngăn cản bộ tối ưu truy vấn sử dụng bất kỳ index nào. Do đó một câu truy vấn nên tách ra và biểu diễn bằng phép hội, mỗi điều kiện trên thuộc tính sẽ sử dụng index.
* Để giúp xúc tiến một truy vấn, những chuyển đổi dưới đây có thể thử:
* Điều kiện **NOT** có thể được chuyển thành biểu thức dương
* Nhúng khối **SELECT** sử dụng IN, =ALL, =ANY và = SOME có thể được thay thế bằng phép kết
* Nếu phép kết bằng được thiết lập giữa hai bảng, xác nhận miền (range predicate) (điều kiện chọn) trên thuộc tính kết được cài đặt trên một bảng có thể lặp lại nhiều lần với các bảng khác
* Điều kiện **WHERE** có thể được viết lại sử dụng index trên nhiều cột.

Ví dụ:

**SELECT REGION#, PROD\_TYPE, MONTH, SALES**

**FROM SALES\_STATISTICS**

**WHERE REGION# = 3 AND ((PRODUCT\_TYPE BETWEEN 1 AND 30) OR (PRODUCT\_TYPE BETWEEN 8 AND 10));**

Có thể sử dụng index chỉ trên REGION# và tìm kiếm tất cả các page lá của index cho PRODUCT\_TYPE trùng khớp. Thay vào đó sử dụng:

**SELECT REGION#, PROD\_TYPE, MONTH, SALES**

**FROM SALES\_STATISTICS**

**WHERE (REGION# = 3 AND ( PRODUCT\_TYPE BETWEEN 1 AND 3)) OR (REGION# = 3 AND (PRODUCT\_TYPE BETWEEN 8 AND 10));**

Có thể sử dụng index kết hợp trên (REGION#, PRODUCT\_TYPE) để hoạt động hiệu quả hơn