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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Nội dung công việc** | **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 |  |  |  |  |

### Topic #4 – Tối ưu truy vấn và các lựa chọn

Mục lục

[I. Quá trình truy vấn và tối ưu 4](#_Toc309861610)

[II. Lợi ích của tính năng tối ưu trong hệ thống cơ sở dữ liệu. 4](#_Toc309861611)

[1. Thay đổi ( hoặc viết lại) truy vấn. 4](#_Toc309861612)

[2. Quan sát sơ đồ thực thi truy vấn 4](#_Toc309861613)

[3. Tần số 5](#_Toc309861614)

[4. Những lời gợi ý sơ đồ thực hiện truy vấn 5](#_Toc309861615)

[5. Tối ưu sâu. 5](#_Toc309861616)

[III. Ước lượng chi phí truy vấn – ví dụ 5](#_Toc309861617)

[IV. Phát triển sơ đồ thực thi câu truy vấn. 10](#_Toc309861618)

[1. Nguyên tắc biến đổi sơ đồ truy vấn 11](#_Toc309861619)

[2. Thuật toán sắp xếp lại sơ đồ thực thi truy vấn. 11](#_Toc309861620)

[V. Nhân tố lựa chọn, kích thước bảng, và định lượng chi phí truy vấn. 12](#_Toc309861621)

[1. Việc ước lượng độ chọn lọc cho 1 phép chọn hoặc vị từ. 12](#_Toc309861622)

[2. Biểu đồ tần số. 13](#_Toc309861623)

[3. Ước lượng đồ chọn lọc cho phép kết. 13](#_Toc309861624)

1. **Quá trình truy vấn và tối ưu**

Các bước cơ bản của quá trình truy vấn:

* Duyệt tìm, phân tích ngữ pháp và phân rã của 1 câu truy vấn: bước này kiểm tra cú pháp của truy vấn, nếu cú pháp sai sẽ thông báo lỗi cho người dùng. Đầu ra của bước này là 1 dạng trung gian của truy vấn được biết dưới dạng cây truy vấn hoặc sơ đồ thực thi truy vấn.
* Tối ưu truy vấn:bước này bao gồm tối ưu cục bộ và toàn cục. tối ưu toàn cục xác định loại kết và loại lựa chọn và phép chiếu liên quan với những phép kết. tối ưu cục bộ xác định phương pháp chỉ mục cho phép chọn và phép kết.
* Tạo ra mã truy vẫn và xử lý: bước này dùng ngôn ngữ lập trình và trình biên dịch để tạo mã xử lý.

1. **Lợi ích của tính năng tối ưu trong hệ thống cơ sở dữ liệu.**
2. **Thay đổi ( hoặc viết lại) truy vấn.**

Các quản trị dữ liệu sẽ viết lại các truy vấn để nó trở thành dạng truy vấn có hiệu suất hơn trước khi qua tối ưu. Điều này giúp sự lựa chọn sơ đồ thực thi rất nhiều. Ví dụ các phép biến đổi thường dùng là: thay đổi các thành phần truy vấn thành các phép kết hoặc 1 phần là phép kết, việc đẩy các phép gom nhóm xuống dưới phép kết, việc loại trừ phép kết trên các khoá ngoại khi các bảng bao gồm các kết quả của phép kết mà không được dùng nhiều trong truy vấn, thay đổi kết ngoài thành kết trong khi chúng cho ra các kết quả như nhau, và thay thế view reference thành actual view.

1 phép biến đổi thường dùng nữa là viết lại khung nhìn cụ thể. Nếu 1 vài phần của truy vấn mà tương đương với 1 khung nhìn cụ thể thì truy vấn đó được thay thế bằng khung nhìn đó..

Biến đổi truy vấn rát quen thuộc trong kho dữ liệu sử dụng dạng lược đồ sao.

1. **Quan sát sơ đồ thực thi truy vấn**

Làm thế nào bạn biết được sơ đồ nào được chọn cho những câu truy vấn thường dùng nhất trong cơ sở dữ liệu của bạn? Hầu hết các hệ quản trị đều cung cấp 1 vài công cụ cho người dùng hiểu rõ được sơ đồ truy cập (access plan). Công cụ này miêu tả tất cả các bước trong sơ đồ, thứ tự các bảng nào được truy cập trong câu truy vấn, và có cần dùng chỉ mục cho bảng này hay không. Khi đó sẽ lựa chọn sơ đồ nào tốt nhất.

1. **Tần số**

Nhiều hệ thống cơ sở dữ liệu lưu tần số xuất hiện của dãy các giá trị thuộc tính từ dữ liệu thực để giúp ước lượng độ chọn lọc của phép chọn và kết.

1. **Những lời gợi ý sơ đồ thực hiện truy vấn**

Song song với giải thích, những gợi ý của sơ đồ trở thành có giá trị nhất định trong công nghiệp cơ sở dữ liệu, cho phép người lập trình ứng dụng lựa chọn sơ đồ nào thích hợp, loại bỏ những trường hợp không chăc chắn. những lời gợi ý là cẩm nang chỉ dẫn người lập trình trong câu truy vấn SQL mà có thể làm thay đổi sơ đồ thực thi của truy vấn đó. Chúng được cung cấp bởi tất cả các hệ thống cơ sở dữ liệu.

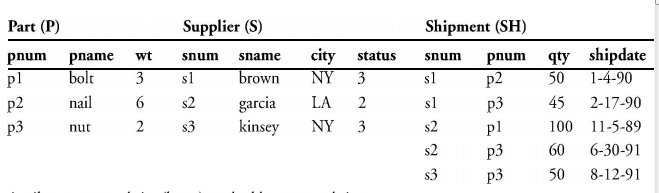
1. **Tối ưu sâu.**

Những quản trị dữ liệu khác nhau có tìm kiếm sâu khác nhau, đơn giản nhất là tìm kiếm kỹ.

1. **Ước lượng chi phí truy vấn – ví dụ**

Giả sử ta có 1 sơ đồ thực thi 1 câu truy vấn đơn giản, và ta tập trung vào định lượng chi phí truy vấn về thời gian truy xuất bé nhất để xử lý câu truy ấn mà sơ đồ của nó được để ra. Nó minh hoạ lợi ích của việc dùng chiến lược tối ưu truy vấn để giải quyết các vấn đề đơn giản ở ngoài đời thực. mặc dù vấn đề này còn cơ bản nhưng nó có lợi ích rất lớn trong thời gian truy vấn mà có thể được xem như là những nguyên tắc, và định nghĩa của những nguyên tắc này được minh hoạ rất rõ.

Ta giả sử có 3 bảng cơ sở dữ liệu được thể hiện cụ thể trong các bảng: part, supplier, intersection .



Tên và kích thước của thuộc tính , các bảng như sau:

* Supplier: snum(5), sname(20), city(10), status(2) 🡪 37 bytes trong 1 record của supplier.
* Part: pnum(8), pname(10), wt(5) --> 23 bytes trong 1 record của part
* Shipment: snum(5), pnum(8), qty(5), shipdate(8) 🡪 26 bytes trong 1 record trong shipment.

Chú ý: giả sử kích thước của block: 15000 bytes.

1. Ví dụ câu truy vấn cho cơ sở dữ liệu trê: những names nào của part được cung cấp bởi nhà cung cấp trong thành phố New York? Ta có truy vấn:

SELECT p.pname

FROM p, sh, s

WHERE P.PNUM = SH.PNUM

AND sh.snum = s.snum

AND s.city = ‘NY’;

* S kết SH kết P (1)
* SH kết S kết P (2)
* P kết SH kết S (3)
* SH kết P kết S (4)
* S X P kết SH (5)
* P X S kết SH (6)

Có 6 cách kết theo thứ tự kết khác nhau của: P, SH, S. phép kết ‘S kết SH kết P’ (1) và phép ‘ SH kết S kết P’ (2) thì tương đương nhau vì phép kết có tính chất giao hoán giữa 2 vế với nhau. Tương tự (3), (4) cũng tương đương nhau, (5) (6) tương đương nhau. Ta nên tránh lệnh (5) (6) nếu có thể vì chúng chứa các bộ cartesian của phép kết khi có sự chồng các cột. khi điều này xảy ra , kích thước của bảng kết quả sẽ tăng lên rất nhiều. và dữ liệu trong bảng kết quả cũng bị nối một cách tuỳ ý.

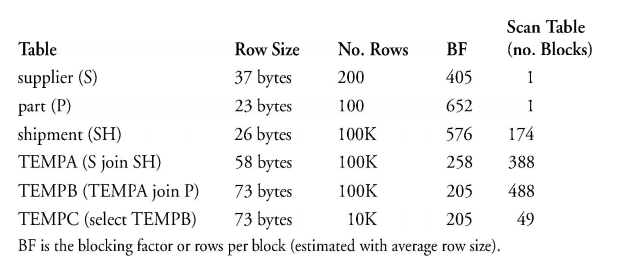
Xem lệnh (1) và (3) ta nên xét nên làm phép kết trước hay là pháp chọn trước. ta cũng xét nên tạo chỉ mục hay không. Giờ ta ước lượng chi phí cho 4 cách sau:

* Cách 1A: dùng lệnh 1 với kết trước, chọn sau và không dùng chỉ mục.
* Cách 1B: dùng lệnh 1 với chọn trước, kết sau, không dùng chỉ mục.
* Cách 3A: dùng lệnh 3 với kết trước, chọn sau, không có chỉ mục.
* Cách 3B: dùng lệnh 3 với chọn trước, kết sau, không có chỉ mục.

Bằng trực giác ta thấy cách 1B cải tiến dựa trên 1A vì phép kết là phép tốn chi phí nhiều nhất, nếu ta giảm kích thước (bằng cách dữ liệu trong bảng trước khi thực hiện phép kết thì sẽ làm tốc độ thức thi nhanh hơn.

Giờ ta sẽ dùng sequential block accesses (SBA) và random block accesses(RBA) như dụng cụ ước lượng thời gian I/O vì có 1 mối quan hệ tuyến tính giữa SBA hoặc RBA với thời gian I/O , và thời gian I/O thì thường được chấp nhận dùng trong việc so sánh chi phí các lệnh trong cơ sở dữ liệu.

Ta tóm tắt kích thước bộ, số lượng dữ liệu, và số lượng block được thống kê dưới bảng sau.



* **Ước lượng chi phí của cách 1A**

Điều này được nói rõ hơn ở hình 3.1: dùng lệnh truy vấn. lệnh truy vấn này chỉ rõ phép kết được thực thi trước tiên, sau đó mới đến phép chọn. ta dùng phép kết merge, vì thế không có chỉ mục được dùng trong truy vấn này. Trong mỗi bước ta ước lượng số lượng record được xét đến trong khoang thời gian được xét, sau đó tính tổng SBA. Ta giả sử mỗi bảng được lưu trên đĩa sau mỗi thao tác lệnh và lệnh kế tiếp phải được đọc từ đĩa lên.

* Bước 1: kết bảng S và SH qua thuộc tính chung là snum tạo ra bảng TEMPA(snum, sname, city, status, qty, shipdate) với kích thước mỗi dòng là 58 byte. Nếu ta sắp xếp bảng trước khi thực thi phép kết thì chi phí cho cách Merge sort là 2 X nb X logM nb [O’Ncil 2001, Silbcrschatz 2006] với nb là số block (trang) trong 1 bảng. ví dụ này M = 3. Tuy nhiên trong trường hợp này, ta không cần sắp xếp SH từ bảng S rất nhỏ, nhỏ hơn cả 1 block, vì thế ta chỉ cần duyệt SH và so sánh trong bộ nhớ nhanh với 1 block của S.

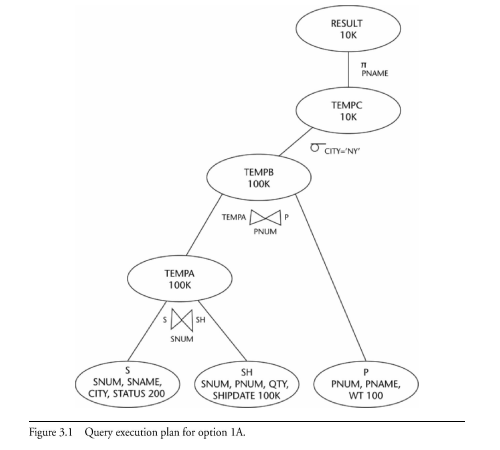
Số lượng block được truy cập:

= read S + read SH + write TEMPA

= ceiling(200/405) + ceiling(100K/576) + ceiling(100K/258).

= 1 + 174 + 388

= 563



* Bước 2: kết P và TEMPA bằng cột pnum thành TEMPB.

Số lượng block truy cập: 5091

* Bước 3: chọn TEMPB ở city = ‘NY’ thành TEMPC ở 73byte mỗi hàng. Giả sử NY có 10% là supplier.

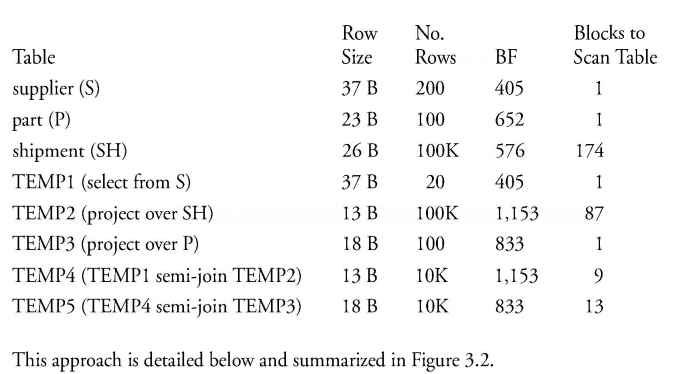
Số lượng block truy cập: 537

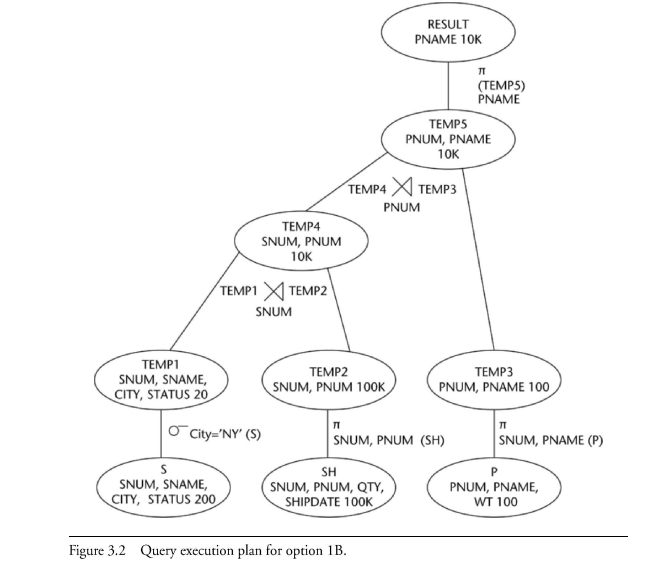
* Bước 4: chiếu TEMPC lấy thuộc tính pname ra RESULT có 10B mỗi hàng.

Số lượng hàng: 10K

Số lượng block: 49.

* **Tổng cộng ta có số lượng block truy cập trong truy vấn theo cách 1A: 6240**
* **Ước lượng chi phí trong cách 1B: làm phép chọn trước( bao gồm cả phép chiếu) phép kết,** **không có chỉ mục.**

****

****

**Cách tính theo hình 3.2 ta được số lượng block truy cập: 398.**

* **So sánh 2 cách ta có:**

**Cách 1A: 6240 xử lý joins trước**

**Cách 1B: 398 xử lý kết sau.**

Rất rõ ràng để thấy giảm số bộ trong bảng thì làm tăng tốc độ rất nhiều.

1. **Phát triển sơ đồ thực thi câu truy vấn.**

Một sơ đồ thực thi truy vấn là 1 cấu trúc dữ liệu mà trình bày các thao tác trên cở sở dữ liệu( phép chọn, phép chiếu, phép kết) như 1 điểm riêng biệt. thứ tự các thao tác trong sơ đồ thực thi truy vấn có thể được thể hiện bằng cách từ trên xuống hoặc từ dưới lên trên. Ta dùng cách tiếp cần từ dưới lên, và trong hình 3.1 và 3.2 những ví dụ kinh điển của việc dùng sơ đồ thực thi truy vấn để thể hiện thứ tự có thể xảy ra của các thao tác cần thực thi trong câu truy vấn SQL. 1 câu truy vấn SQL có thể có nhiều thứ tự thực thi truy vấn khác nhau, phụ thuộc vào đồ phức tạp của câu truy vấn đó, và mỗi thứ tự có thể được trình bày bởi 1 sơ đồ thực thi riêng biệt. mục đích của ta là tìm ra sơ đồ thực thi truy vấn nào mà nó đưa ra câu trả lời chính xác trong khoảng thời gian ngắn nhất.

Một thuật toán phổ biến để tối ưu truy vấn hiện nay đó là thuật toán nói ở phần III: đặt thao tác chọn và chiếu trước khi kết vì kết tốn nhiều chi phí. Chỉ nên làm việc với phép kết với các bảng kích thước càng bé càng tốt.

Để thuận tiện thay đổi 1 sơ đồ thực thi truy vấn từ việc sắp xếp ngẫu nhiên các thứ tự thao tác sang thứ tự có phương pháp mà phép chọn và chiếu được thực thi trước và phép kết sau thì ta tóm tắt ngắn gọn những nguyên tắc sau ( có thể xem chúng như một thuật toán):

1. **Nguyên tắc biến đổi sơ đồ truy vấn**

Dưới đây là những nguyên tắc hiển nhiên trong việc biến đổi sơ đồ thực thi truy vấn để đảo ngược thứ tự nhưng vẫn cho ra kết quả giống nhưa. Cho phép những cây truy vấn khác nhau thì tạo ra kết quả giống nhau.

* + - **nguyên tắc 1:** tính giao hoán của phép kết: R1 kết R2 = R2 kết R1.
    - **Nguyên tắc 2**: tính kết hợp của phép kết: R1 kết (R2 kết R3) = (R1 kết R2) kết R3.
    - **Nguyên tắc 3**: thứ tự các phép chọn trong 1 bảng không làm ảnh hưởng đến kết quả truy vấn.
    - **Nguyên tắc 4**: không cần quan tâm đến thứ tự phép kết và phép chiếu trên cùng 1 bảng. miễn là 1 phép chiếu không loại bỏ 1 thuộc tính được dùng trong phép chọn.
    - **Nguyên tắc 5**: phép chọn trên 1 bảng trước phép kết cho ra kết quả như nhau với cách chọn sau khi kết trên bảng đó.
    - **Nguyên tắc 6**: thứ tự phép chiếu và phép kết chứa các thuộc tính giống nhau thì không cần quan tâm đến thứ tự trong truy vấn miễn là các thuộc tính loại bỏ trong phép chiếu không được chứa trong phép kết.
    - **Nguyện tắc 7**: phép chọn hoặc phép chiếu và phép hội liên quan đến cùng bảng thì không cần phân biệt thứ tự thao tác.

1. **Thuật toán sắp xếp lại sơ đồ thực thi truy vấn.**

Dưới đây là 1 khám phá đơn giản đề sắp xếp lại sơ đồ truy vấn để nó tối ưu hoặc gần tối ưu.

* + - Một phép chọn riêng biệt với 1 vài mệnh đề AND thành những phép chọn có thứ tự.
    - Đặt các phép chọn xuống sâu trong sơ đồ nếu có thể, để thao tác chọn được thực thi trước.
    - Nhóm 1 thứ tự các phép chọn nếu có thể.
    - Đẩy phép chiếu xuống sâu trong sơ đồ nếu có thể.
    - Nhóm các phép chiếu trong cùng 1 bảng, loại bỏ phép chiếu dư thừa.

1. **Nhân tố lựa chọn, kích thước bảng, và định lượng chi phí truy vấn.**

1 lần chúng ta có được 1 sơ đồ thực thi truy vấn được đề xuất, ta cần phải ước lượng kích thước những bảng trung gian mà câu truy vấn tối ưu đó sẽ tạo ra trong suốt quá trình thực thi. 1 lần ta ước lượng tổng số kích thước những bảng đó , ta có thể tính toán được thời gian I/O cần dùng để thực thi truy vấn này.

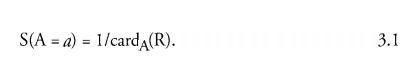
Đề dễ hiểu ta định nghĩa các đo lường sau trong bảng dữ liệu:

* + - Số lượng hàng của bảng R: card(R)
    - Số lượng các giá trị khác nhau của thuộc tính A trong bảng R: cardA(R )
    - Giá trị lớn nhất của A trong bảng R: maxA(R )
    - Giá trị nhỏ nhất của A trong bảng R: minA(R )

1. **Việc ước lượng độ chọn lọc cho 1 phép chọn hoặc vị từ.**

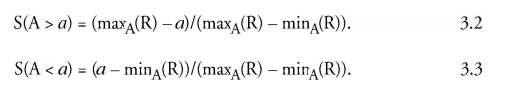
Mối quan hệ dưới đây cho biết làm thế nào tính độ chọn lọc của các phép chọn trong truy vấn SQL.

Độ chọn lọc của 1 thuộc tính A trong bảng R phải có 1 giá trị rõ ràng trong 1 record được chọn ra ở 2 tình huống. một là, nếu thuộc tính A là khoá chính, nơi mà mỗi giá trị chỉ xác định duy nhát, sau đó ta đo độ chọn lọc:



Ví dụ, 1 bảng có 50 record, khi đó độ chọn lọc là 1/50.

Mặt khác, nếu thuộc tính không phải là khoá và mỗi giá trị a xảy ra nhiều trong bảng thì phép tính như sau:



Độ chọn lọc của 2 phép chọn chéo nhau trên cùng 1 bảng thì đực tính:

S(P and Q) = S(P) X S(Q) 3.4

Với P và Q là vị tự.

1. **Biểu đồ tần số.**

Ta tính toán độ chọn lọc bằng cách tính độ trung bình dựa trên tính đúng đắn của dữ liệu, nhưng mỗi dữ liệu khác nhau có thể gây ra số lượng sai khác lớn, nếu tất cả cơ sở dữ liệu chỉ dùng cách tính xấp xỉ này thì ước lượng thời gian truy vấn có thể làm sai lệch nghiêm trọng. nhưng may mắn , nhiều hệ thống quản lý cơ sở dữ liệu hiện nay thường lưu lại các giá trị thuộc tính có thực dưới dạng sơ đồ tần số. trong sơ đồ tần số, các giá trị thuộc tính được chia thành các dãy, và trong mỗi dãy, sẽ thể hiện số lượng các hàng mà thuộc thuộc tính đó có. Sơ đồ tần số rất giúp ích cho việc ược lượng thời gian truy vấn tốt hơn, trung thực, khách quan hơn.

1. **Ước lượng đồ chọn lọc cho phép kết.**

Ước lượng độ chọn lọc cho 1 phép kết rất khó nếu nó dựa vào 1 thuộc tính không khoá, trong trường hợp xấu nhất có thể rơi vào trường hợp tạo ra tích Cartesian. Ta chỉ bàn ước lượng phép kết dựa trên thuộc tính khoá và khoá ngoại. ví dụ, phép kết giữa bảng R1(bảng chứa khoá) với bảng R2( chứa khoá ngoại):



S là độ chọn lọc cho thuộc tính chung dùng trong phép kết, tức là thuộc tính đó là 1 khoá.

### Topic #5 –Lựa chọn các indexes

Mục lục

[I. Khái niệm và thuật ngữ chỉ mục: 15](#_Toc309759931)

[1. Các loại chỉ mục cơ bản: 15](#_Toc309759932)

[2. Các phương pháp truy vấn đối với index 16](#_Toc309759933)

[II. Qui tắc chỉ mục theo kinh nghiệm: 16](#_Toc309759934)

[III. Quyết định lựa chọn chỉ mục: 17](#_Toc309759935)

[IV. Chọn lựa chỉ mục kết hợp: 20](#_Toc309759936)

[1. Phép kết vòng lặp lồng (nested-loop join): 20](#_Toc309759937)

[2. Phép kết vòng lặp lồng block (block nested-loop join) 22](#_Toc309759938)

[3. Phép kết vòng lặp lồng chỉ mục (indexed nested-loop join) 22](#_Toc309759939)

[4. Phép kết kết hợp với sắp xếp: 23](#_Toc309759940)

[5. Phép kết hash: 24](#_Toc309759941)

[THỦ THUẬT VÀ CÁC QUAN NIỆM DÀNH CHO CHUYÊN GIA CƠ SỞ DỮ LIỆU 25](#_Toc309759942)

**SELECTING INDEXES**

# Khái niệm và thuật ngữ chỉ mục:

Chỉ mục là một trong các công cụ cơ bản được sử dụng trong hệ quản trị cơ sở dữ liệu để phân chia hiệu suất cao cho nhiều câu truy vấn thường sử dụng của hệ thống. Chỉ mục có mục đích đa dạng trong phạm vi hiệu suất: tìm kiếm nhanh thông tin cụ thể và phạm vi dữ liệu, sự ép buộc tính duy nhất và v.v…

Các mức độ tinh chỉnh cơ sở dữ liệu:

* **mức đầu tiên** (thấp nhất) trong tinh chỉnh cơ sở dữ liệu là mức phần cứng. Hiện tượng thắt cổ chai có thể được giảm hoặc loại bỏ bằng cách tăng tài nguyên, chẳng hạn làm CPU chạy nhanh hơn, tăng dung lượng bộ nhớ (không gian bộ nhớ đệm), thêm đĩa, sử dụng hệ thống RAID hoặc stripping (phân vùng) dữ liệu dọc các đĩa nhằm tăng khả năng xử lí song song
* **mức thứ hai** là trong chính hệ thống cơ sở dữ liệu. Hiệu suất hệ thống đôi khi có thể được cải thiện bằng cách tăng kích thước buffer hoặc thay đổi khoảng cách giữa các checkpoint.
* **mức thứ ba** (cao nhất) là thực hiện ở mức lược đồ và các giao tác. Giao tác có thể được thiết kế lại để đạt tốc độ cũng như trong tinh chỉnh truy vấn được thực hiện bởi hệ thống. Lập trình viên phải viết cẩn thận các câu truy vấn SQL vì cơ chế tinh chỉnh của DBMS không hoàn hảo. Các bảng có thể không đạt dạng chuẩn để đạt hiệu suất cao nhưng bất lợi trong xóa và cập nhật các bất thường. Cuối cùng quyết định về các khung nhìn, phân vùng, và lựa chọn chỉ mục được thực hiện ở bước này.

## Các loại chỉ mục cơ bản:

* **Chỉ mục cây B+** : chỉ mục bảng đơn giản dựa trên một hoặc nhiều thuộc tính, thường được dùng để ràng buộc tính duy nhất của khóa ngoại
* **Chỉ mục bảng Hash**: Một chỉ mục giúp ánh xạ giá trị khóa chính tương ứng địa chỉ block dữ liệu trong cơ sở dữ liệu cho các dữ liệu chưa được sắp xếp.
* **Chỉ mục kết hợp**: Một chỉ mục dựa trên nhiều hơn một thuộc tính hoặc khóa
* **Chỉ mục phân nhóm (chỉ mục cluster**): Bất kì cấu trúc chỉ mục nào cũng có dữ liệu được truy vấn theo cách index được tổ chức vì thế dữ liệu với các giá trị khóa tương tự hoặc giống nhau sẽ được lưu trữ chung. Phân nhóm dường như giảm thiểu đáng kể thời gian I/O đối với truy vấn và lưu trữ.
* **Chỉ mục bao phủ (chỉ mục duy nhất):** một chỉ mục với đầy đủ thông tin thỏa các truy vấn cụ thể bằng chính nó, hay nói cách khác là truy vấn thỏa yêu cầu khi chỉ cần tìm kiếm trong chỉ mục mà không cần tìm trong cơ sở dữ liệu
* **Chỉ mục bitmap:** một tập hợp các mảng bit để hình thành một chỉ mục có thể được sử dụng để tìm kiếm nhanh chỉ mục thứ cấp hoặc tìm kiếm trong kho dữ liệu. Mỗi mảng bit đại diện cho một giá trị khác nhau của thuộc tính, và độ dài của mỗi mảng là số dòng trong bảng.
* **Chỉ mục dày và thưa:** một chỉ mục dày chứa con trỏ đến mỗi dòng trong bảng, một chỉ mục thưa chứa hầu hết một con trỏ đến một block hoặc page dữ liệu trong bảng

## Các phương pháp truy vấn đối với index

Mỗi hệ cơ sở dữ liệu có các phương pháp truy vấn có sẵn được sử dụng để quét các index và dữ liệu, và mỗi hệ thống được cài đặt để tối ưu tìm kiếm dữ liệu cho mỗi loại truy vấn cụ thể. Các phương pháp cơ bản thường sử dụng là:

* **Quét bảng** (table scanning): khi chỉ mục không cần thiết
* **Quét chỉ mục**(index scanning): đối với chỉ mục phân nhóm và không phân nhóm (clustered and nonclusted index)
* **Quét chỉ mục duy nhất** (index only): quét một chỉ mục bao phủ
* **Chỉ mục block hoặc dòng** (block or row index ANDing): đối với entry chỉ mục kết hợp lại cho các truy vấn đa điểm
* **Nạp trước danh sách:** sự sắp xếp trên các record ID được kèm theo việc nạp trước các dòng đã sắp xếp từ block hoặc trang dữ liệu, cho phép các phép toán tuần tự trong tương lai trên dữ liệu này được nạp lên trước

# Qui tắc chỉ mục theo kinh nghiệm:

1. **Đặt chỉ mục trên mỗi khóa chính và hầu hết các khóa ngoại trên cơ sở dữ liệu:** hầu hết các phép kết xảy ra trên khóa chính và khóa ngoại, do đó hãy đặt chỉ mục trên khóa chính và khóa ngoại bất cứ khi nào có thể.

Chúng ta lưu ý rằng ở hầu hết các hệ quản trị cơ sở dữ liệu, khi khóa chính được khai báo trong SQL, chỉ mục cũng được tạo ra. Khóa ngoại nên được xem xét trong phạm vi sử dụng như một trường tìm kiếm, và trong quá trình xóa khóa chính. Cụ thể là, khi có một số lượng nhỏ giá trị có thể (như giới tính, địa chỉ state), chỉ mục có thể khá lớn và bị ảnh hưởng khi xóa khóa chính.

1. **Thuộc tính thường được dùng trong mệnh đề WHERE là ứng viên tiềm năng cho chỉ mục.** Mệnh đề WHERE được dùng đối với thuộc tính tìm kiếm trong câu truy vấn và là ứng viên cho chỉ mục khi vị từ là một phép bằng. Những thuộc tính vừa được dùng để hạn chế số dòng mục tiêu hoặc phép kết của 2 bảng. Trong cả hai trường hợp, chỉ mục được xem xét để tăng tốc tính toán

Một ngoại lệ đối với qui tác này khi vị từ là phép không bằng như “WHERE cost <> 4”. Chỉ mục ít khi được dùng trong phép không bằng vì có nhiều sự chọn lựa.

1. **Sử dụng chỉ mục cây B+ cho cả truy vấn bằng và truy vấn phạm vi.**
2. **Chọn cẩn thận một cluster index cho mỗi bảng.** Một cluster index cho biết cách dữ liệu được lưu trữ vật lý như thế nào đối với một bảng cụ thể. Việc cài đặt một cluster index trên khó chính rất quan trọng, cho phép tìm kiếm phạm vi và duy trì một thứ tự giúp xử lý nhóm hiệu quả.
3. **Tránh hoặc xóa các chỉ mục dư thừa.** Tránh các chỉ mục được cài đặt trên cùng cột hoặc hầu hết các cột đều giống nhay, nó sẽ làm truy vấn phức tạp hơn.
4. **Chỉ thêm chỉ mục khi thật sự cần thiết.**
5. **Thêm hoặc xóa cột chỉ mục đối với chỉ mục kết hợp để cải thiện hiệu suất. Đừng thay đổi cột khóa chính.**
6. **Sử dụng thuộc tính đối với chỉ mục cần chú ý khi nào chúng thường được cập nhât.**
7. **Duy trì chỉ mục trên nền tảng thông thường, chỉ xóa chỉ mục khi chúng thật sự ảnh hưởng hiệu suất**. Duy trì chỉ mục có thể giảm hiệu suất của các dòng lệnh đáng kể: INSERT, UPDATE, DELETE, IMPORT, LOAD.
8. **Tránh các cực trị trong bản số và phân phối giá trị.**
9. **Chỉ mục bao phủ (covering index – index only) rất có ích nhưng không được lạm dụng**. Chỉ mục bao phủ có thể được dùng để tìm kiếm dữ liệu, tráng I/O với database
10. **Sử dụng chỉ mục bitmap đối với lượng dữ liệu lớn, đặc biệt là trong data warehouse**

# Quyết định lựa chọn chỉ mục:

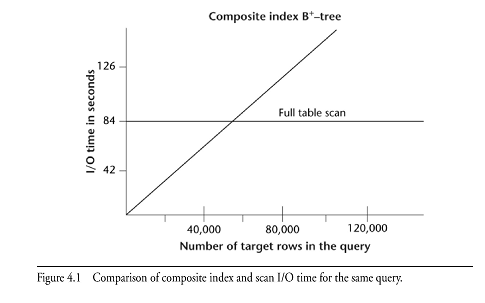
Chúng ta cần xem xét các chỉ mục mở rộng gia tăng và tìm hiểu liệu có cách nào tốt hơn hay không, ví dụ hiệu suất có được cải thiện đáng kể với mỗi index mới hay không. Không có bất kì tiêu chuẩn cụ thể nào ở đây hết, nhưng chúng ta phải cài đặt độc lập cho mỗi hệ thống cơ sở dữ liệu và người dùng.

Chúng ta cũng cần xem xét các thao tác cập nhật. Đôi khi một chỉ mục mới cải thiện hiệu suất truy vấn nhưng lại làm giảm hiệu năng cập nhật. Sự kết hợp giữa hiệu suất truy vấn và cập nhật sẽ quyết định một chỉ mục có ích hay không. Điều này là sự cân bằng thường gặp trong thiết kế chỉ mục, và đòi hỏi chúng ta ghi lại các cập nhật nào là cần thiết, tần số và bảng nào mà chúng tác động đến.

* **Quyết định thiết kế 1: Bảng này cần thiết một chỉ mục hay không, và nếu cần thì nên cài đặt chỉ mục trên từ khóa tìm kiếm nào?** Xuất phát điểm hợp lý cho thiết kế chỉ mục là khai báo kháo chính và chỉ mục trên khóa ngoại, bất chấp hỗn hợp truy vấn bạn đang thực hiện. Đối với các ứng dụng đòi hỏi trong phạm vi tần số, lượng truy cập dữ liệu nhiều, và/hoặc độ ưu tiên, thêm hoặc sửa chỉ mục cần phải được xem xét nếu hiện tượng thắt cổ chai xuất hiện.

Nếu quyết định cần chỉ mục, hãy xem xét mệnh đề WHERE ( và AND) của truy vấn để quyết định thuộc tính nào có thể được dùng cho khóa tìm kiếm chỉ mục. Mỗi truy vấn có thể liên quan đến nhiều thuộc tính khác nhau và chỉ mục nên được chọn cho mỗi truy vấn mà có thể cải thiện đáng kể hiệu suất cơ sở dữ liệu. Các truy vấn đòi hỏi quét toàn bảng có thể không cần bất kì chỉ mục nào. Chỉ mục tăng tốc cho hơn một câu truy vấn sẽ có giá trị nhất.

* **Quyết định thiết kế 2: Khi nào cần từ khóa tìm kiếm đa thuộc tính (kết hợp) và nên chọn cái nào?** Một truy vấn đa điểm liên quan mệnh đề WHERE với nhiều thuộc tính. Khi loại truy vấn này thực hiện, chỉ mục cây B+ kết hợp nên được sử dụng để truy vấn hiệu quả tập hợp các dòng thỏa điều kiện truy vấn. Phân tích sự cân bằng giữa quét toàn bảng và chỉ mục B+ đa thuộc tính được thể hiện trong hình bên dưới



* **Quyết định thiết kế 3: Nên sử dụng chỉ mục dày hay thưa?** Khi số dòng nhỏ tương ứng với kích thước page, chỉ mục thưa tốt hơn. Một chỉ mục dày sẽ có nhiều entry hơn và sẽ thường yêu cần thêm mức chỉ mục nữa để có thể truy xuất dữ liệu hơn chỉ mục thưa. Mặt khác, chỉ mục dày có thể kết hợp một cách hiệu quả để tạo ra chỉ mục kết hợp cho truy vấn đa điểm. Khi số dòng lớn tương ứng kích thước trang lớn, chỉ mục thưa ít hiệu quả hơn. Một ưu thế khác của chỉ mục thưa là nó có thể trở thành chỉ mục bao phủ cho một số truy vấn nhất định
* **Quyết định thiết kế 4: Khi nào có thể dùng chỉ mục bao phủ?** Đôi khi một chỉ mục kết hợp có thể được sử dụng như chỉ mục bao phủ đối với một số truy vấn nhất định có thể đạt yêu cầu khi chỉ tìm kiếm trên index. Ví dụ, mọt chỉ mục trên attribute, make, và model có thể thỏa câu truy vấn sau:

**SELECT make, model, vin, year**

**FROM carOnLot**

**WHERE feature = ‘catalytic converter’;**

* **Quyết định thiết kế 5: Nên sử dụng chỉ mục phân nhóm (clustered index) hay không?** Các qui tắc cơ bản của cluster index:
* Chỉ có một cluster index được cài trên một bảng, hay nói cách khác sẽ có nhiều xung đột giữa các dòng thỏa nhiều truy vấn sử dụng các chỉ mục khác nhau
* Nếu một chỉ mục có thể được dùng như một chỉ mục bao phủ, thì không cần phân nhóm nữa, vì câu truy vấn có thể đạt yêu cầu chỉ cần truy vấn chỉ mục
* Truy vấn phạm vi, truy vấn đa điểm và truy vấn đơn điểm trên các giá trị không phải khóa chính có thể được lợi từ phân nhóm. Mỗi loại truy vấn liên quan đến truy xuất nhiều dòng, và bất kì sự phân nhóm nào cũng sẽ cải thiện hiệu suất. Nếu ta có vài lựa chọn, phân tích sự cân bằng có thể cần để quyết định chọn lựa nào sẽ tối ưu hiệu suất.
* **Quyết định thiết kế 6: Một chỉ mục có còn thích hợp khi cập nhật được thực hiện? Đâu là sự cân bằng giữa truy vấn và cập nhật đối với mỗi lựa chọn chỉ mục?** Xem xét sự cân bằng từ viễn cảnh chi phí / lợi ích, xem xét thời gian I/O với tất cả giao tác liên quan đến truy vấn và các cập nhật của mỗi bảng mục tiêu đối với chỉ mục đã cho, tính đến tần số mỗi truy vấn và cập nhật trong một khoảng thơi gian cố định:

***Lợi ích của chỉ mục = thời gian I/O (tất cả truy vấn không có index) – thời gian I/O (tất cả truy vấn có index)***

***Chi phí của chỉ mục = thời gian I/O (tất cả cập nhật có index) – thời gian I/O (tất cả cập nhật không có index)***

Chú ý mỗi cập nhật có 2 bước: truy vấn để truy xuất đến dòng được cập nhật và thao tác cập nhật dòng đó. Clustered index dường như có chi phí cập nhật cao hơn so với nonclustered index, và nên được xem xét cẩn thận

**Qui tắc cơ bản**: tạo một chỉ mục nếu lợi nhiều hơn hại (lợi ích nhiều hơn chi phí). Qui tắc này có một ngoại lệ phải xem xét, bởi vì nó chỉ dựa trên thời gian I/O. Độ ưu tiên truy vấn có thể cao hơn độ ưu tiên cập nhật, đặc biệt nếu cập nhật được xử lý đợt trong giờ không cao điểm trong ngày, nghĩa là bạn có thể sống với việc hoãn cập nhật lại vài giờ. Trong trường hợp này, lợi ích nhất định thấp hơn chi phí, bởi vì được đo bằng thời gian I/O, vì vậy mà độ ưu tiên là một yếu tố quan trọng hơn.

Mặt khác, nếu cập nhật phải thực hiện ngay lập tức, truy vấn bị hoãn lại thì lợi ích đương nhiên lơn hơn chi phí khi xác định sử dụng chỉ mục

* **Quyết định thiết kế 7: Làm sao biết đã chọn lựa đúng chỉ mục?** Chúng ta có thể thực hiện một vài phân tích cân bằng hiệu suất như đo thời gian I/O cần thiết để đáp ứng một tập các truy vấn trên bảng hoặc một tập các bảng và chỉ mục. Để so sánh được thì dữ liệu cần được lấy trước và sau cài đặt chỉ mục. Nếu không có hiệu quả nào, hoặc có tình trạng giảm hiệu năng, hãy xem xét các phương án tốt hơn

# Chọn lựa chỉ mục kết hợp:

Phân tích của chúng ta được dựa trên giả sử nếu có một phép kết giữa một bảng mà khóa ngoại của nó khớp với khóa chính của bảng khác, m đại diện cho số dòng trong bảng có khóa chính, n đại diện cho số dòng trong bảng có khóa ngoại. Nếu phép kết không giữa 2 bảng như trên, thì thiết kế của bảng nào có m dòng và bảng nào có n dòng là tuỳ ý.

## Phép kết vòng lặp lồng (nested-loop join):

Chiến lược lồng vòng lặp là một phương pháp kết cơ bản. Vòng lặp ngoài là một phép quét tuần tự bảng đầu tiên R, và với mỗi dòng trong bảng R, vòng lặp trong được thực hiện, một phép quét tuần tự bảng thứ hai S. Tham số cơ bản là số dòng, m và n, trong hai bảng của phép kết và thứ tự vật lý các dòng trong mỗi bảng. Độ phức tạp là O(mn) vì lặp 2 lần. Mã giả của phép kết:

**For each tuple r in R do**

**For each tuple s in S do**

**If r and s satisfy the join condition**

**Then output the tuple <r,s>**

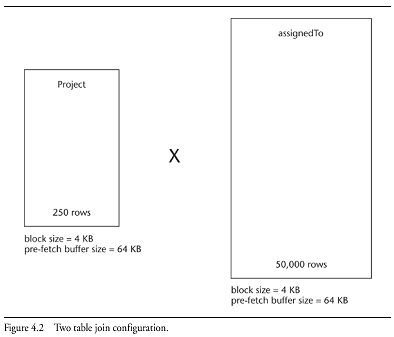
Chúng ta giả sử mỗi bảng được lưu trong không gian đĩa vật lý kề nhau để có lợi cho quét toàn bảng. Chi phí thời gian thực hiện chiến lược này còn tùy thuộc vào bảng nào được chọn cho vòng lặp ngoài và vòng lặp trong

***Ví dụ: giả sử có bảng Employee với 10,000 dòng, bảng Project với 250 dòng, và bảng AssignedTo với 50,000 dòng. Ta có câu truy vấn sau (không tính thời gian hiển thị kết quả của phép kết)***

***SELECT p.projectName, p.projectLeader, a.empId***

***FROM project AS p, assignedTo AS a***

***WHERE p.projectName = a.projectName;***

****

***Biết kích thước block: 4KB***

***Kích thước bộ đệm nạp trước: 64KB***

***Thời gian truy xuất block (giả sử xét ổ cứng IBM U320): 3,6ms (seek time), thời gian xoay trung bình là 2ms, và tốc độ chuyển tải là 320 MB/sec.***

*Số buffer cần nạp trước (64KB) cho bảng assignedTo*

*= ceiling (( 50,000 dòng x 100bytes/dòng) /64KB)*

*= ceiling (5,000,000/65,536) = 77*

*Số buffer cần nạp trước cho bảng project*

*= ceiling((250 dòng x 200 bytes/dòng) / 64KB)*

*= ceiling (50,000 /65,536)*

*= 1*

Trong đó ceiling là hàm lấy mức tối đa của số thực

**Vòng lặp lồng trường hợp 1:** assignedTo là bảng vòng lặp ngoài

*Thời gian I/O phép kết = quét assignedTo 1 lần, quét bảng project n lần*

*= ( 77 buffers + 50,000 lần quét x 1 buffer) x 5.8ms*

*= 290.4 sec*

**Vòng lặp lồng trường hợp 2:** project là bảng lặp ngoài

*Thời gian I/O phép kết = quét project 1 lần, quét bảng assignedTo m lần*

*= (1 buffer + 250 lần quét x 77 buffer) x 5.8ms*

*= 19,251 buffer x 5.8ms*

*= 111.7 sec*

Một cách tổng quát bảng nhỏ hơn thường được chọn là bảng ngoài sẽ có lợi hơn trong thực thi truy vấn. Chú ý chiến lược này không tận dụng bất kì sự sắp xếp dòng hoặc phương pháp chỉ mục nào. Chúng ta cũng nên tránh các phép kết dựa trên các thuộc tính không khóa

## Phép kết vòng lặp lồng block (block nested-loop join)

Phép kết lồng vòng lặp block chỉ là sự đa dạng của phép kết lồng vòng lặp trong đó bảng vòng lặp trong được quét mỗi lần một block của bảng vòng lặp ngoài thay vì mỗi lần một dòng. Khi một block của bảng vòng lặp được lưu trong buffer, thuật toán có thể quét tất cả dòng trong block mà không có thao tác I/O. Xử lý này có thể giảm đáng kể số I/O cần thiết cho phép kết và thích hợp hơn khi tiếp cận vòng lặp lồng cơ bản. Hãy xem xét lại ví dụ bên trên

**Trường hợp 1:** assignedTo là bảng vòng lặp ngoài

*Thời gian I/O phép kết = quét assignedTo 1 lần, quét project 77 lần nạp trước vào buffer*

*= ( 77 buffer + 77 lần quét x 1 buffer) x 5.8 ms*

*= 0.89 sec*

**Trường hợp 2:** project là bảng vòng lặp ngoài

*Thời gian I/O phép kết = quét project 1 lần, quét assigned m lần*

*= ( 1 buffer + 1 lần quét x 77 buffer) x 5.8 ms*

*= 45 sec*

Cấu hình vòng lặp lồng block nhanh hơn so với cấu hình vòng lặp lồng cơ bản. Chúng ta lưu ý rằng tạo bảng project là bảng vòng lặp ngoài là chiến lược tốt hơn cho cả 2 cấu hình

## Phép kết vòng lặp lồng chỉ mục (indexed nested-loop join)

Phương pháp này là một thay thế có lợi cho phép kết vòng lặp lồng block khi sự chọn lựa trong phép kết thấp vì thế chỉ có một tập hợp nhỏ các dòng của bảng kết cần được truy xuất.

**Ví dụ 4.2;**

**SELECT p.projectName, p.projectLeader, a.empId**

**FROM project AS p, assignedTo AS a**

**WHERE p.projectName = a.projectName**

**AND p.projectName = ‘financial analysis’;**

**Trường hợp 1:** quét bảng khóa ngoại 1 lần và chỉ mục đến bảng khóa chính một lần

Chiến lược cơ bản là quét toàn bộ bảng chứa khóa ngoại, và với mỗi giá trị thuộc tính kết thỏa yêu cầu, xác định dòng duy nhất tương ứng trong bảng khóa chính thông qua chỉ mục duy nhất. Sẽ có thao tác quét bảng khóa ngoại và (h +1) lần truy xuất block của khóa chính thông qua chỉ mục duy nhất đối với mỗi dòng. Giả sử số dòng mục tiêu ntr = 100 – dòng thỏa bảng khóa ngoại khớp với một dòng trong bảng khóa chings. Giả sử h = 2

*Thời gian I/O phép kết= quét toàn bảng khóa ngoại (assignedTo) + chỉ mục đến những dòng thỏa của bảng khóa chính (project)*

*= (77 buffer + (h+1) truy xuất block (buffer)) x 5.8ms*

*= ( 77 + 3) x 5.8 ms*

*= 0.46 sec*

**Trường hợp 2**: chỉ mục đến bảng khóa chính, và sau đó chỉ mục đến bảng khóa ngoại

Một phương pháp khác là sử dụng cây B+ hoặc chỉ mục hash đến bảng khóa chính để tìm các giá trị khóa chính thỏa điều kiện, sau đó sử dụng chỉ mục B+ kết hợp đến bảng khóa ngoại tìm các dòng chứa giá trị khóa ngoại trùng với giá trị khóa chính. Đối với trường hợp này, giả định h = 3, một block đòi hỏi con trỏ ntr = 100 dòng khóa ngoại mục tiêu được cho như trong trường hợp 1

*Thời gian I/O phép kết = chỉ mục đến bảng khóa chính + chỉ mục đến bảng khóa ngoại*

*= ( (h +1) + [h + 1 + ntr]) x 5.8ms*

*= (4 + 104 ) 5.8 ms*

*= 0.63 sec*

## Phép kết kết hợp với sắp xếp:

Chiến lược phép kết kết hợp sắp xếp lợi dụng các dòng theo thứ tự tương tự cách xử lý theo đợt thực hiện. Nếu bảng được sắp xếp trên các cột kết, thì chỉ cần một phép quét tuần tự từng bảng để hoàn thành phép kết. Nếu một hoặc cả hai bảng chưa được sắp xếp trên cột kết, thì mỗi bảng chưa sắp xếp sẽ được sắp xếp trước khi phép trộn thực thi. Thậm chí khi tính cả chi phí sắp xếp, thuật toán này vẫn nhanh hơn vòng lặp lồng. Nếu truy vấn không đòi hỏi sự sắp xếp(tức trong cây truy vấn có order by) thì phép kết kết hợp sắp xếp sẽ hiệu quả hơn phép kết vòng lặp lồng block và ngược lại

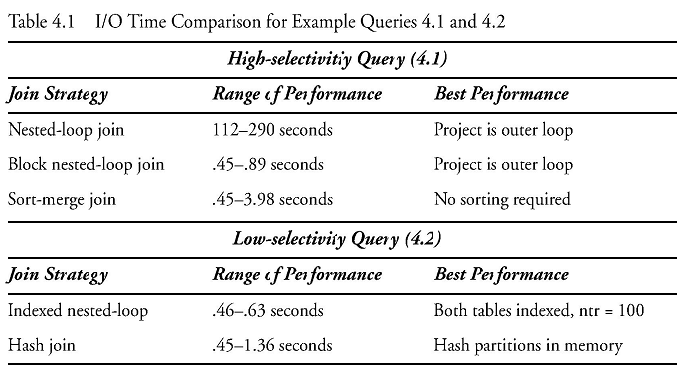
## Phép kết hash:

Chiến lược kết hash cũng hiệu quả cho phép kết ít sự chọn lọc. Chiến lược cơ bản là xây dựng một bảng hash trong bộ nhớ, nếu có thể, hoặc trên đĩa nếu không đủ bộ nhớ, bằng cách quét mỗi bảng kết và băm giá trị thuộc tính các cột thỏa yêu cầu để phân vùng. Thuật toán thay thế còn được sử dụng để băm input nhỏ hơn trong hai input của phép kết vào bộ nhớ và sau đó đọc input lớn hơn khớp với bảng hash trong bộ nhớ. Chúng ta giả sử thuật toán đầu tiên ở đây, nơi mà hai phân vùng được tìm kiếm để xác định các thuộc tính bẳng (lưu ý các thuộc tính khác nhau có thể băm(hash) vào cùng nơi, vì vậy cần thiết phải tìm kiếm để kiểm tra chúng, nhưng các giá trị thuộc tính tương tự sẽ không băm vào các vị trí khác nhau)

Khi một bảng băm được xây dựng, pha thứ hai của thuật toán là truy vấn đến các dòng thật sự thỏa điều kiện từ mỗi bảng (chẳng hạn những dòng từ hai bảng với các giá trị thuộc tính giống nhau ở các phân vùng file hash). Độ phức tạp là O (3m + 2n) vì có 3 pha thực hiện quét: một pha đọc 2 bảng, một pha ghi lại 2 bảng vào các phân vùng bằng giá trị băm, và một để đọc các phân vùng cho phép kết thật sự.

Trong thực thi phép kết hash, thao tác quét bảng cho các phân vùng có thể thực hiện không thường xuyên miễn là phân vùng hash file trong bộ nhớ vẫn không bị thay đổi đối với chuỗi các truy vấn

**Tóm lại, đối với truy vấn có nhiều sự lựa chọn, cả phép kết vòng lặp lồng block và phép kết trộn với sắp xếp đều thực hiện tốt. Đối với phép kết có ít sự lựa chọn, phép kết vòng lặp lồng index và phép kết hash làm việc tốt, với phép kết hash có ít giá trị hơn cho cận trên đối với các truy vấn cho trước.**



# THỦ THUẬT VÀ CÁC QUAN NIỆM DÀNH CHO CHUYÊN GIA CƠ SỞ DỮ LIỆU

* **Thủ thuật 1: Số lượng và kích thước chỉ mục thay đổi.** Tuy nhiên theo tổng quát, chỉ mục thường chiếm khoảng 10% - 20% không giân đĩa cơ sở dữ liệu. Hơn 25% đĩa được sử dụng cho chỉ mục nên bị phản đối và nên tiến hành xem xét lần hai một cách nghiêm túc.
* **Thủ thuật 2: Chỉ mục giúp truy vấn nhưng làm hại thao tác ghi. Vì vậy cơ sở dữ liệu truy vấn** nhiều có thể phạm sai lầm trên chỉ mục, và cơ sở dữ liệu có cả nhiều hoạt động insert/delete/update, hoặc đối với cơ sở dữ liệu nào mà hiệu suất của các hoạt động ghi được chú trọng, nên bảo toàn số lượng chỉ mục chúng định nghĩa cho hệ thống.
* **Thủ thuật 3: Ôn lại toàn bộ qui tắc chỉ mục dựa trên kinh nghiệm đã liệt kê ở trên.**

### Topic #6 – Lựa chọn các khung nhìn

Mục lục

[Cụ thể hóa khung nhìn đơn giản 2](#_Toc309759931)

[I. Khai thác phổ biến: 31](#_Toc309759934)

[II. Khai thác theo nhóm và tổng quát: 37](#_Toc309759934)

[III. Xem xét tài nguyên : 39](#_Toc309759935)

[IV. Các ví dụ tốt xấu 43](#_Toc309759936)

[V. Cách sử dụng cú pháp và các ví dụ: 46](#_Toc309759936)

[VI. Tổng kết 48](#_Toc309759936)

**CỤ THỂ HÓA KHUNG NHÌN ĐƠN GIẢN**

Kiểu khung nhìn cụ thể hóa đơn giản nhất chứa dữ liệu cùng loại với kết quả của một câu truy vấn xác định. Đọc từ khung nhìn cụ thể hóa cho kết quả giống với kết quả được tính toán từ các bảng cơ sở. Chúng tôi sẽ minh họa với một ví dụ cụ thể và giải thích sự tiết kiệm truy vấn I/O.

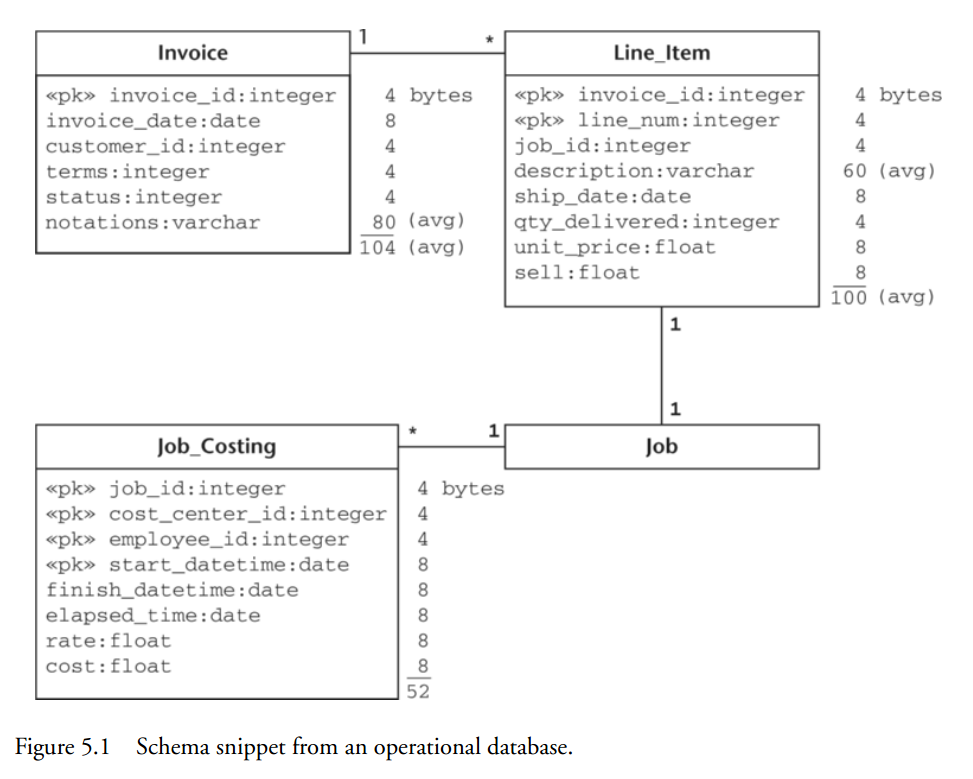
Hình 5.6 là một phần của một lược đồ từ một CSDL được chuẩn hóa. Chúng tôi chỉ tham khảo các bảng nằm trong phạm vi của chương này. Các bảng thích hợp trong CSDL này chứa các dữ liệu có ích cho chu kỳ tính toán, chi phí công việc tại một nhà máy sản xuất.

Mỗi hóa đơn (invoice) có một ngày hóa đơn (invoice date) và một khách hàng liên quan (customer). Các điều khoản (terms) của hóa đơn cho biết trong bao nhiêu ngày khách hàng phải trả hóa đơn. Tình trạng (status) cho biết giai đoạn hiện tại của chu kì thanh toán cho hóa đơn. Các ghi chú (notations) có thể chứa bất kì ý kiến chung nào, chẳng hạn như ghi chú cuộc thoại với khách hàng về hóa đơn.

Một hóa đơn có thể có nhiều mặt hàng, mỗi loại một dòng riêng. Ví dụ, bạn có thể đã mua các đồ dùng màu xanh, đã được vận chuyển ngày 23 tháng 10 năm 2006 (ship\_date), số lượng (qty\_delivered) là 100 với đơn giá (unit\_price) là 2.50$ mỗi cái, tổng giá trị (sell) là 250$. Bạn cũng có thể mua các đồ dùng màu đỏ, nằm trên một dòng khác trên cùng một hóa đơn. Dữ liệu này được lưu trữ trong bảng Line\_Item.

Mỗi chi tiết hóa đơn có liên quan với một công việc được sản xuất tại nhà máy. Bảng Job có chứa các chi tiết kỹ thuật của công việc. Tuy nhiên các chi tiết này không được thể hiện ở đây vì các chi tiết này không nằm trong nội dung thảo luận của chúng tôi.

Bảng Job\_Costing chứa dữ liệu theo dõi thời gian và chi phí sản xuất cho từng công việc. Khi công việc đi qua một máy, các công nhân tại máy đó quét mã ID của công việc trên thẻ CV. Công nhân cũng có thể quét mã chi phí trung tâm đại diện cho hoạt động đã được thực hiện, và mã này chứa mã của nhân viên. Thời gian bắt đầu và thời gian kết thúc cho hoạt động được tự động ghi lại bởi hệ thống máy tính. Dữ liệu được lưu trong bảng Job\_Costing, cùng với mức lương giờ cho chi phí trung tâm và chi phí tính từ thời gian trôi qua và tỉ lệ tính theo giờ. Những thông tin này có ích trong việc so sánh chi phí thực tế và giá trị bán ra của mỗi công việc, dẫn tới quá trình cải tiến và sinh nhiều lợi nhuận hơn.



Với các tính toán tiếp theo, chúng tôi giả sử các số liệu thống kê trong CSDL như sau:

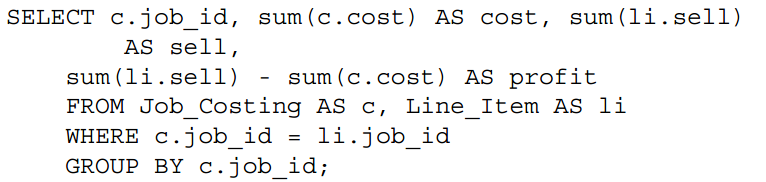
* Average jobs per day = 10,000.
* Average invoices per day = 5,000.
* Average Job\_Costing rows per job = 50.
* Day on record = 200.
* Customers = 5,000.
* The rows in all tables are clustered in primary key order.
* Anything 200MB (i.e., 50,000 blocks) or less can be sorted in memory.

Chúng tôi tiếp tục sử dụng các công thức I/O. Các tính toán của chúng tôi trong chương này đều dựa trên ổ cứng IBM U320 146GB.

* I/O time (4KB block in a dedicated disk) = 2.0 ms.
* I/O time (64KB buffer in a dedicated disk) = 2.2 ms.
* I/O time (4KB block in a shared disk) = 5.6 ms.
* I/O time (64KB block in a shared disk) = 5.8 ms.

Quét bảng cần giả định sử dụng các bộ đệm nạp trước. Mặc dù thời gian I/O truyền 64KB dài hơn truyền I/O 4KB, truyền 64KB chỉ thực hiện 1 lần so với 16 lần, vì thế hiệu quả tổng thể tốt hơn 16 lần.

Bộ phận quản lí của công ty muốn theo dõi lợi nhuận của công việc. Dịch vụ thông tin đã cung cấp một ứng dụng phân tích lợi nhuận tuyệt vời. Thông thường ứng dụng được chạy bởi 10 nhà quản lý mỗi ngày. Ứng dụng đòi hỏi thông tin về chi phí, giá bán và lợi nhuận cho từng công việc. Các thông tin này có thể lấy từ CSDL hiện có bằng câu truy vấn sau đây:



Hãy tính thời gian cho câu truy vấn này, sau đó chúng tôi sẽ cải thiện hiệu suất bằng cách sử dụng khung nhìn cụ thể hóa. Câu truy vấn liên quan đến 2 bảng Line\_Item và bảng Job\_Costing. Chúng tôi cần tính số lượng hàng, hệ số block và số block cho mỗi bảng. Có 10,00 công việc mỗi ngày, với 200 ngày trong bản ghi, cho tổng cộng 2,000,000 dòng trong bảng Line\_Item. Vì chúng tôi sẽ thực hiện quét bảng, chúng tôi sẽ giả định bộ đệm 64KB cho các hoạt động đọc và ghi I/O. Các phép tính cho bảng Line\_Item như sau:

Average rows per prefetch buffer

= floor ((65,536 bytes/block) / (100 bytes/row)) = 655.

Number of buffers

= ceiling (2,000,000 rows / (655 rows/buffer)) = 3,054.

Bảng Job\_Costing chứa trung bình 50 dòng cho mỗi công việc, 10,000 công việc mỗi ngày, với 200 ngày trong bản ghi, cho tổng cộng 100,000,000 dòng. Các phép tính của bảng Job\_Costing như sau:

Average rows per prefetch buffer

= floor ((65,536 bytes/block) / (52 bytes/row)) = 1,260.

Number of buffers = ceiling (100,000,000 rows / (1,260 rows/buffer)) = 79,366.

Bởi vì chúng tôi kết hai bảng hoàn toàn nên kiểu merge-join là hiệu quả. Dữ liệu trong bảng Job\_Costing được sắp xếp bởi job\_id. Bảng Line\_Item cần phải được sắp xếp bởi job\_id để thực hiện được phép kết merge-join. Bảng Line\_Item đủ nhỏ để sắp xếp trong bộ nhớ, vì thế chúng tôi chỉ cần một lần quét tuần tự để tải bảng vào bộ nhớ.

Join cost (shared disk) = scan time (Line\_Item and Job\_Costing tables using 64KB prefetch buffers)

= (3,054 + 79,366 buffers) x 5.8 ms

= 82,420 x 5.8 ms

= 478 second 8 minutes.

Đây là một khoảng thời gian khá dài cho một câu truy vấn, đặc biệt khi bạn phải làm chúng nhiều lần và 10 nhà quản lý quyết định chạy chương trình phân tích lợi nhuận tuyệt vời của bạn để lấy thông tin cho cuộc họp lớn của họ sẽ bắt đầu trong 5 phút nữa. Cuối cùng cũng đã có người gõ cánh cửa của bạn. Chúng tôi có thể làm gì? Chúng tôi có thể tính toán các kết quả cần thiết trong giờ nghỉ, và lưu trữ dữ liệu đã được xử lý trong khung nhìn cụ thể hóa. Sau đó khi các nhà quản lý đến, họ có thể có kết quả ngay lập tức. Hình 5.2 cho thấy lược đồ của khung nhìn cụ thể hóa.

Đầu tiên, hãy tính thời gian tạo khung nhìn, sau đó chúng tôi sẽ tính thời gian để lấy kết quả từ bảng Profit\_by\_Job. Khung nhìn được tạo bằng cách chạy các câu truy vấn ban đầu, sau đó viết kết quả vào bảng Profit\_by\_Job. Số block (buffer) truy cập tham gia là như nhau, nhưng thời gian I/O bây giờ ngắn hơn bởi vì môi trường đã được dành sẵn trong giờ nghỉ.

Join cost (dedicated disk) = 82,420 x 2.2 ms 181 seconds 3 minutes.

Chúng tôi cần tính hệ số blocking để tính thời gian ghi kết quả vào bảng Profit\_by\_Job. Chúng tôi có 10,000 công việc mỗi ngày, với 200 ngày trong bản ghi, vì thế chúng tôi có 2,000,000 dòng trong bảng Profit\_by\_Job. Các phép tính của bảng Profit\_by\_Job:

Average rows per prefetch buffer

= floor ((65,536 bytes/block) / (28 bytes/row)) = 2,340.

Number of buffers = ceiling (200,000,000 rows / (2,340 rows/buffer)) = 855.

Write cost (dedicated disk) = 855 x 2.2 ms 2 seconds.

Profit\_by\_Job creation cost = join cost + write cost = 181 + 2 = 183 seconds.

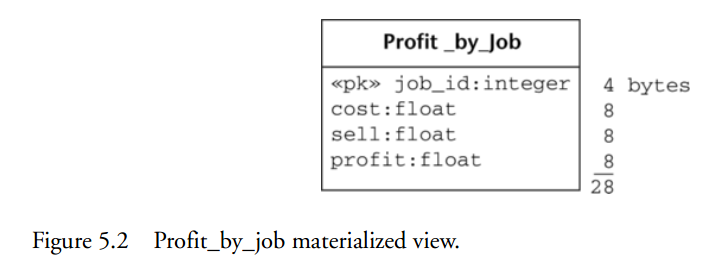
Câu truy vấn lợi nhuận theo công việc sẽ đơn giản hơn khi dựa trên khung nhìn mới này:



Chúng tôi đã tính được có 855 buffer cho bảng Profit\_by\_Job. Câu truy vấn được chạy trên đĩa chia sẻ (5 phút trước khi họp).

Query I/O time (table scan, shared disk) = 855 × 5.8 ms ≈ 5 seconds.

Những người quản lý có bản báo cáo của họ trong tay, hơn thế nữa còn có nhiều thời gian để nói chuyện trước cuộc họp.



Cùng tóm tắt lại:

Disk I/O time before Profit\_by\_Job

= query frequency × I/O time per query

= (10 queries/day) × (478 sec/query)

= 4,780 seconds (or about 1 hour, 20 min)

Disk I/O time with Profit\_by\_Job

= creation cost + query frequency × I/O time per query

= 183 sec creation cost + (10 queries/day) × (5 sec/query)

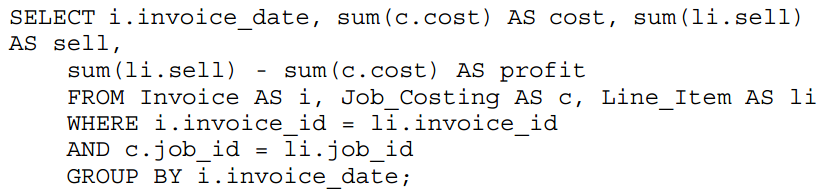
= 233 seconds.

* Sử dụng khung nhìn cụ thể hóa có thể mang lại cải thiện đáng kể cho cả đĩa I/O và tốc độ truy vấn. Tuy một số lợi ích có được từ việc tận dụng ưu điểm của đĩa chuyên dùng để tạo ra khung nhìn, nhưng phần lớn đến từ 2 yếu tố đầu tiên. Thứ nhất, khung nhìn cụ thể hóa có thể nhỏ hơn so với các bảng cơ sở, dẫn tới đạt hiệu quả lớn trong hiệu suất truy vấn đĩa I/O. Thứ 2 là các câu truy vấn thường xuyên làng tăng hiệu quả này.

1. **KHAI THÁC PHỔ BIẾN**

Có một số câu truy vấn thường xuyên xem lợi nhuận từ các quan điểm khác nhau. Một sự phân tích hợp lý để theo dõi lợi nhuận bởi khách hàng. Ví dụ, nếu cúng tôi biết ai là khách hàng lợi nhuận cao nhất, chúng tôi có thể thu hút thêm nhiều công việc bằng cách giảm giá số lượng lớn nếu họ đồng ý mang lại số công việc ít nhất trong một khung thời gian nhất định. Chúng tôi kiểm tra khả năng của hai khung nhìn để trả lời nhanh chóng các câu truy vấn này. Sau đó chúng tôi sẽ xem xét một khung nhìn thay thế khác tận dụng lợi thế của sự tương đồng của ba câu truy vấn lợi nhuận của chúng tôi.

Xu hướng lợi nhuận theo ngày được xem tóm tắt mỗi ngày một lần bởi người quản lý cấp trên và cũng có thể quản lý tính toán. Chúng tôi xác định rằng câu truy vấn thường được chạy mỗi ngày 5 lần. Lợi nhuận tính bằng cách gom nhóm ngày của hóa đơn được trình bày dưới đây:



Câu truy vấn này đọc từ các bảng Invoice, Job\_Costing và Line\_Item. Chúng tôi đã tính số block cho bảng Job\_Costing và bảng Line\_Item. Chúng tôi vẫn phải tính số block cho bảng Invoice. Trung bình có 5,000 hóa đơn mỗi ngày, với 200 ngày trong bản ghi, cho 1,000,000 dòng trong bảng Invoice.

Phép tính của bảng Invoice như sau:

Average rows per prefetch buffer

= floor ((65,536 bytes/block) / (104 bytes/row)) = 630.

Number of buffers = ceiling (1,000,000 rows / (630 rows/buffer)) = 1,588.

Có hai phép kết có thể thực hiện được, dẫn tới hai kế hoạch truy vấn thay thế sau đây.

PLAN A:

* B1: thực hiện merge-join giữa Line\_Item và Invoice (không yêu cần sắp xếp)
* B2: merger-join kết quả bước 1 với Job\_Costing (yêu cần sắp xếp KQ bước 1 the job\_id)

PLAN B

* B1: merger-join Line\_Item và Job\_Costing (yêu cầu sắp xếp theo job\_id)
* B2: merge-join kết quả bước 1 với Invoice (yêu cần sắp xếp KQ bước 1 the invoice\_id)

Cả hai kế hoạch trên yêu cầu quét tất cả ba bảng cơ sở. Kế hoạch A là tốt hơn bởi vì nó yêu cầu ít phép sắp xếp hơn kế hoạch B. Sauk hi merge-join bảng Line\_Item với Inovoice, chỉ cần giữ cột invoice\_date, job\_id và sell. Kết quả là 2,000,000 dòng, mỗi dòng 20 byte cho khoảng 10,000 block. Kết quả đủ nhỏ để sắp xếp trong bộ nhớ, vì thế chúng tôi chỉ cần đến thời gian quét các bảng một lần.

Query I/O time (shared disk)

= scan Line\_Item + scan Invoice + scan Job\_Costing

= (3,054 blocks + 1,588 blocks + 79,366 buffers scanned) × 5.8 ms

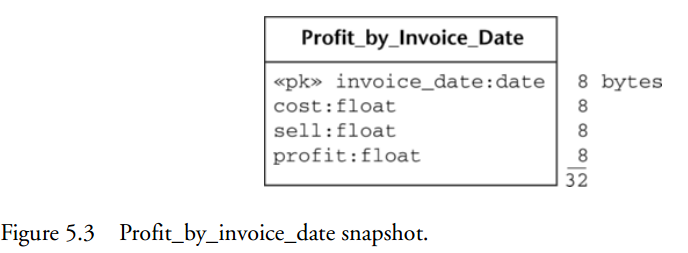
= 84,008 × 5.8 ms

= 487 seconds.

Kết quả tương tụ cũng có được từ khung nhìn cụ thể hóa trong hình 5.3.

Chúng tôi có thể phát sinh khung nhìn cụ thể hóa từ đầu trong giờ nghỉ bằng cách chạy câu truy vấn tương tự, và sau đó lưu trữ kết quả.

Query I/O time (dedicated disk) = 84,008 × 2.2 ms ≈ 185 seconds.



Chúng tôi có 200 ngày trong bản ghi, vì thế bảng Profit\_by\_Invioce\_Date có 200 dòng. Các phép tính của bảng Profit\_by\_Invioce\_Date:

Average rows per block (buffer)

= floor((65,536 bytes/buffer)/(32 bytes/row)) = 2,048.

Number of blocks (buffers) = ceiling(200 rows/(2,048 rows/block))

= 1.

Write cost (dedicated disk) = 1 × 2.2 ms = 2.2 ms.

Profit\_by\_Invoice\_Date creation cost = query cost + write cost

= 185 sec + .0022 sec

≈ 185 seconds.

Câu truy vấn được biến đổi như sau:



Thời gian tính toán I/O cho câu truy vấn trên đĩa chia sẻ là:

Query I/O time (shared disk) = 1 × 5.8 ms = 5.8 ms.

Disk I/O time before Profit\_by\_Invoice\_Date

= query frequency × I/O time per query

= (5 queries/day) × (487 sec/query)

= 2,435 seconds (just over 40 min)

Disk I/O time with Profit\_by\_Invoice\_Date

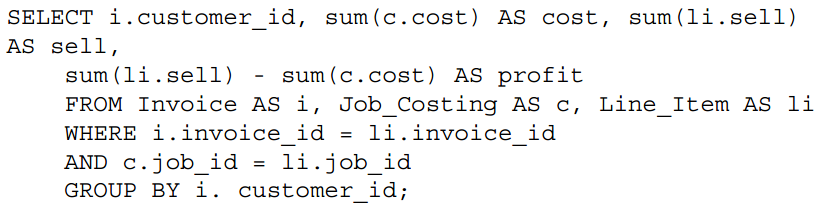
= creation cost + query frequency × I/O time per query

= 185 sec creation cost + (5 queries/day)

× (5.8 ms/query) ≈ 185 seconds.

Có một cách tiếp cận khác để duy trì khung nhìn cụ thể hóa có thể cải thiện lợi ích trong nhiều trường hợp. Thay vì tính khung nhìn từ đầu mỗi đêm, chúng có thể được duy trì thêm. Ví dụ, trong trường hợp của bảng Profit\_by\_Invoice\_Date, chúng tôi có thể giữ dữ liệu hiện có, và thêm dòng mới bất cứ khi nào một chu trình hóa đơn mới hoàn thành.

Câu truy vấn tính lợi nhuận theo khách hàng tương tự như lợi nhuận tính theo ngày hóa đơn. Câu truy vấn này chủ yếu được chạy bởi bộ phận bán hàng. Chúng tôi xác định rằng câu truy vấn này chạy 3 lần mỗi ngày. Đây là câu truy vấn dựa trên bảng cơ sở:

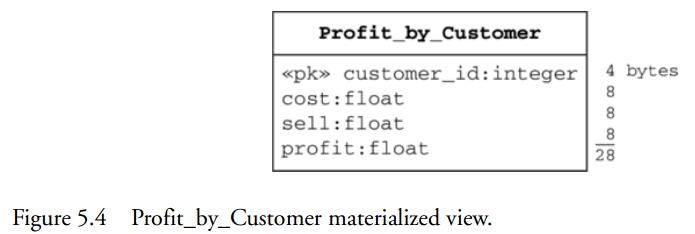


Chú ý rằng các bảng và các phép kết tương tự như khi thực hiện truy vấn theo ngày hóa đơn. Bởi vì cùng một bảng được đọc, và cùng một phép kết được thực hiện, vì thế thời gian truy vấn I/O cũng không thay đổi.

Query I/O time (shared disk)

= scan Line\_Item + scan Invoice + scan Job\_Costing

= 84,008 × 5.8 ms = 487 seconds.



Hình 5.4 là hình của khung nhìn tương ứng. Chi phí I/O để tạo bảng Profit\_by\_Customer tương đương với chi phí truy vấn trên bảng cơ sở cộng với chi phí viết kết quả vào đĩa.

Query I/O time (dedicated disk) = 84,008 × 2.2 ms ≈ 185 seconds.

Chúng tôi có 5,000 khách hàng, vì thế bảng Profit\_by\_Customer có 5,000 dòng. Các phép tính của bảng Profit\_by\_Customer:

Average rows per block (buffer)

= floor((65,536 bytes/block)/(28 bytes/row)) = 2,340.

Number of blocks (buffers)

= ceiling(5,000 rows/(2,340 rows/block)) = 3.

Write cost (dedicated disk) = 3 × 2.2 ms = 6.6 ms.

Profit\_by\_Customer creation cost = query cost + write cost

≈ 185 sec + .0066 sec ≈ 185 seconds.

Câu truy vấn được biến đổi thành:



Phép tính thời gian truy vấn I/O cho đĩa chia sẻ là:

Query I/O time (sequential scan of entire table, shared disk)

= 3 × 5.8 ms = 17.4 ms.

Disk I/O time before Profit\_by\_Customer

= query frequency × I/O time per query

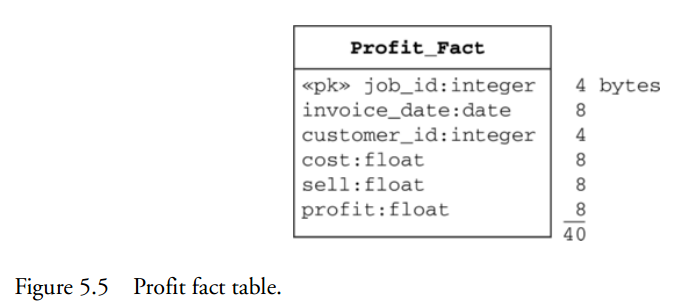
= (3 queries/day) × (487 sec/query) = 1,461 seconds.

Disk I/O time with Profit\_by\_Customer

= creation cost + query frequency × I/O time per query

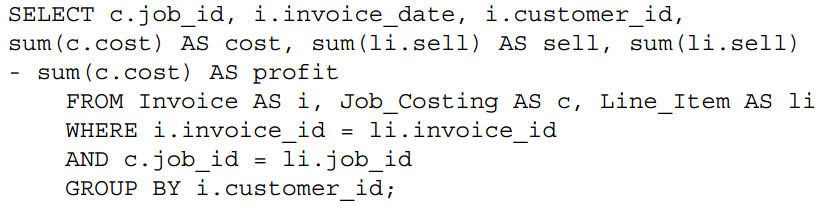
= 185 sec creation cost + (3 queries/day)

× (0.017 sec/query) ≈ 185 seconds.



Nếu chúng bạn nhìn lại hình 5.2, 5.3 và 5.4, bạn sẽ thấy rằng chúng đều có cost, sell và profit. Lý do là chúng là yếu tố để kết hợp các bảng này. Hình 5.5 là lược đồ bảng kết hợp. job\_id là khóa chính, bởi vì chúng tôi có phụ thuộc hàm: job\_id → invoice\_date, customer\_id, cost, sell, profit .

Câu truy vấn phát sinh dữ liệu từ bảng Profit\_Fact là:



Chi phí tạo bảng Profit\_Fact bằng với chi phí truy vấn từ bảng cơ sở và cho phí viết kết quả lên đĩa. Câu truy vấn không sử dụng bảng cơ sở dẫn tới phép kết giống nhau mà chúng tôi tính được cho bảng Profit\_by\_Customer.

Query I/O time (dedicated disk) = 84,008 × 2.2 ms ≈ 185 seconds.

Số dòng bằng với số công việc, bởi vì job\_id là khóa chính. Vì thế, có 2,000,000 dòng trogn bảng Profit\_Fact. Các phép tính của bảng Profit\_Fact:

Average rows per block (buffer)

= floor((65,536 bytes/buffer)/(40 bytes/row)) = 1,638.

Number of buffers = ceiling(2,000,000 rows/(1,638 rows/buffer))

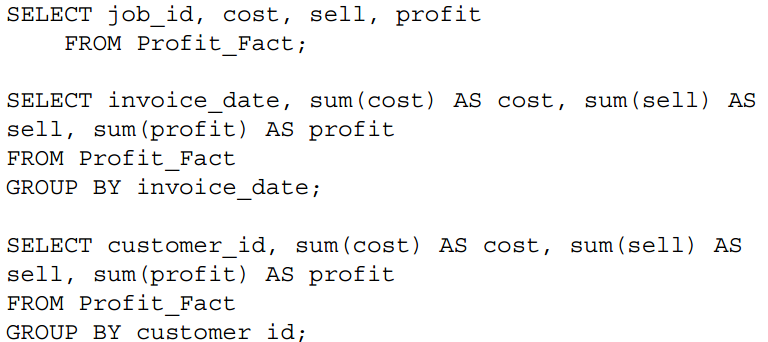
= 1,222.

Write cost (dedicated disk) = 1,222 × 2.2 ms ≈ 3 seconds.

Profit\_Fact creation cost = query cost + write cost

≈ 185 sec + 3 sec ≈ 188 seconds.

Sử dụng khung nhìn cụ thể hóa mô tả trong hình 5.5, 3 câu truy vấn của chúng tôi là:



Tất cả các câu truy vấn đều quét qua bảng Profit\_Fact\_Table.

Query I/O time (sequential scan, shared disk)

= 1,222 × 5.8 ms ≈ 7 seconds.

Tóm lại, chúng tôi sẽ tính tổng thời gian cho ba câu truy vấn: lợi nhuận theo công việc, theo ngày, theo khách hàng. Chúng tôi sẽ tính số trường hợp khác nhau cho ba câu truy vấn: sử dụng bảng cơ sở, khung nhìn dành riêng cho mỗi câu và sử dụng bảng Profit\_Fact\_Table.

Disk I/O times using base tables

= (4,780 sec + 2,435 sec + 1,461 sec)/day = 8,676 seconds.

Disk I/O times using Profit\_by\_Job, Profit\_by\_Invoice\_Date,

and Profit\_by\_Customer = (233 sec + 185 sec + 185 sec)/day

= 603 sec/day.

Disk I/O time using Profit\_Fact table

= creation cost + query frequency × I/O time per query

= 188 sec creation cost + (10 profit by job queries/day)

× (7 sec/query)

+ (5 profit by date queries/day) × (7 sec/query)

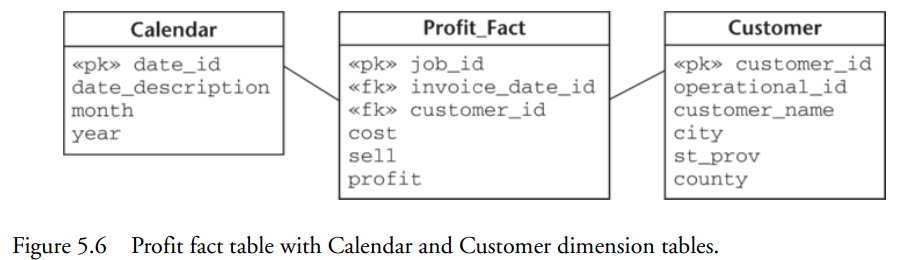
+ (3 profit by customer queries/day) × (7 sec/query)

≈ 314 seconds per day.

Sử dụng khung nhìn chung tiết kiệm nhiều thời gian hơn so với sử dụng ba khung nhìn cho mỗi câu truy vấn. Nếu như cách này không được chấp nhận trong một số trường hợp, phương án thay thế là tạo bảng Profit\_Fact, sau đó tạo các khung nhìn khác từ bảng Profit\_Fact hơn là tạo từ các bảng cơ sở. Bởi vì bảng Profit\_Fact nhỏ hơn các bảng cơ sở, thời gian tạo các khung nhìn sẽ được giảm đáng kể.

1. **KHAI THÁC THEO NHÓM VÀ TỔNG QUÁT**

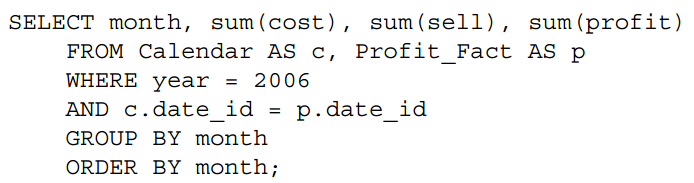
Tình hình của công ty phụ thuộc vào các xu hướng ngắn hạn và dài hạn. Theo dõi mỗi ngày là cần thiết, nhưng chúng ta cũng cần các báo cáo tháng và năm. Tương tự như vậy, theo dõi lợi nhuận theo khách hàng cũng rất cần thiết. Chúng ta có thể mở rộng bảng trong hình 5.5 để hỗ trợ cho các câu truy vấn khác. Hình 5.6 cho thấy việc bổ sung hai bảng để tạo điều kiện thuận lợi cho các câu truy vấn khác. Bảng Calendar chứa cột tháng và năm có thể được gom nhóm để tính xu hướng từ bảng Profit\_Fact. Cột state\_province và country của bảng Customer có thể được gom nhóm để tính chiến lược về địa lý.



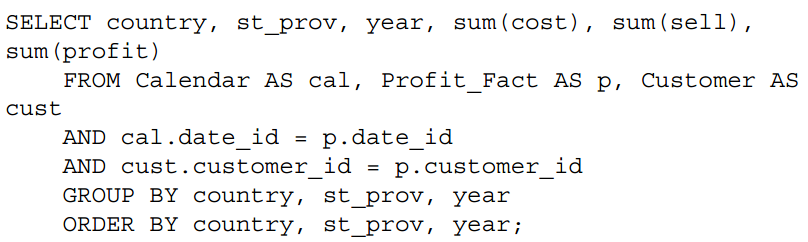
Nếu bạn đã quen thuộc với kho dữ liệu, bạn sẽ nhận ra hình 5.6 là một lược đồ sao đơn giản với các bảng hai chiều, cụ thể là bảng Calendar và Customer.

Chúng tôi đề nghị hai câu truy vấn ví dụ minh họa làm thế nào các bảng này có thể được sử dụng để nhóm dữ liệu trong các kết quả chung. Sau đó chúng tôi sẽ tổng hợp mô hình liệt kê và tổng hợp các mối kết hợp trong mỗi chiều.

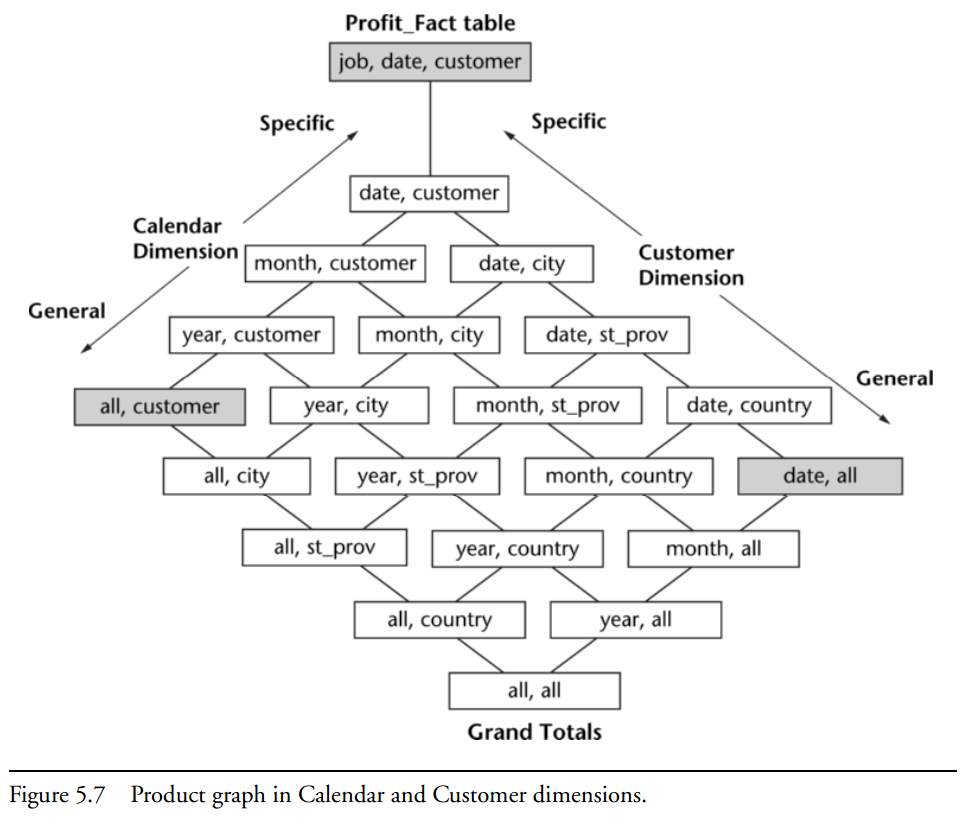
Đây là câu truy vấn lấy thông tin lợi nhuận theo tháng năm 2006:



Đây là câu truy vấn lấy thông tin theo năm:



Có nhiều phép kết của các cấp có thể. Chúng tôi có thể liệt kê các phép kết và minh họa mối quan hệ trong mô hình. Hình 5.7 là mô hình cho lược đồ sao trong hình 5.6. Trên thực tế, mô hình sẽ có ba chiều, bởi vì có chiều Job. Tuy nhiên. Để giữ cho đồ thị đơn giản, chúng tôi đã bỏ bậc gồm job\_id, ngoại trừ bảng Profit\_Fact quan trọng trong dữ liệu nguồn. Chiều Calendar và Customer là trực giao. Mỗi nút là một khung nhìn.



1. **XEM XÉT TÀI NGUYÊN**

Có một số hạn chế về tài nguyên khi bạn thiết kế khung nhìn. Bao gồm số lượng khung nhìn, dung lượng đĩa yêu cầu, chiều dài của cửa sổ cập nhật có sẵn. Chúng tôi sẽ thảo luận từng yếu tố.

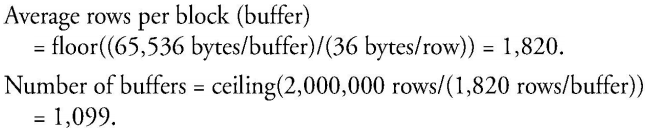
Số lượng các khung nhìn có thể có trong một cơ sở dữ liệu thông thường rất lớn. Cho một tập hợp các kích thước, số lượng các khung nhìn là sản phẩm của số lượng bậc trong mỗi kích thước. Số lượng các khung nhìn có thể có trong ví dụ hình 5.7 là 4 bậc Calendat X 5 bậc Customer X 2 bậc Job = 40 khung nhìn có thể có. Hãy nhớ rằng, chúng tôi bỏ bậc của bảng Job để mô hình dễ đọc hơn. Chúng tôi chỉ kiểm tra quá trình kinh doanh của phân tích lợi nhuận. Thông thường có nhiều tiến trình kinh doanh cần phải xem xét khi thiết kế CSDL. Rõ ràng là chúng ta không thể cụ thể hóa mọi khung nhìn, vì thế chúng tôi phải chọn ra một tập hợp các chiến lược của các khung nhìn cho sự cụ thể hóa.

Nghiên cứu trước đây trong vấn đề chọn khung nhìn cụ thể hóa đã sử dụng hạn chế về số lượng khung nhìn được cụ thể hóa [Harinarayan, Rajaraman, and Ullman 1996]. Ngày nay, nó vẫn còn hữu dụng để ép buộc số lượng khung nhìn cụ thể hóa vì ít nhất hai lý do: thời gian có thể để thiết kế CSDL là hạn chế và cụ thể hóa nhiều khung nhìn đòi hỏi nhiều thời gian thiết kế hơn. Cụ thể hóa nhiều khung nhìn còn đòi hỏi nhiều tiến trình cập nhật. Không những làm tăng số lượng các khung nhìn cụ thể hóa sẽ sử dụng nhiều hơn các tài nguyên máy tính, nó còn yêu cần nhiều CSDL quản lý. Khi các bảng cơ sở phát triển với nhu cầu của công ty, các khung nhìn cụ thể hóa và các tiến trình cập nhật phải được điều chỉnh. Thiết lập một giới hạn về số lượng các khung nhìn, bạn đang sẵn sang để thiết kế và duy trì nó.

Tập trung vào các khung nhìn được dùng để trả lời thường xuyên các câu truy vấn là một ý kiến hay để giảm sự phức tạp của quá trình thiết kế để dễ quản lý hơn. Xây dựng một sản phẩm đồ thị hoàn chỉnh là một vấn đề có độ phức tạp tăng theo cấp số nhân của số lượng kích thước (dimension). Tuy nhiên, một mạng đơn giản có thể được xây dựng dựa trên các câu truy vấn thường xuyên. Mạng lưới trong hình 5.8 minh họa một cấu trúc mạng đơn giản. Các bảng cơ sở thích hợp nằm ở trên cùng. Khung nhìn Profit\_Fact có thể đượ tạo từ các bảng cơ sở. Ba khung nhìn khác có thể được tính từ khung nhìn Profit\_Fact.

Về cơ bản, mạng lưới trong hình 5.8 là một phiên bản giảng lược của đồ thị trong hình 5.7. Các khung nhìn không được truy vấn thường xuyên không được thêm vào, làm cho đồ thị đơn giản hơn. Chúng tôi đã thêm vào các bảng cơ sở và các lượn đồ khung nhìn vào mạng, một cách ngắn gọn nắm bắt được tình hình. Mỗi nút được gắn với kích thước trong các block, và tần số truy vấn liên quan khi thích hợp. Siêu dữ liệu này rất quan trọng khi tính toán các lợi ích có liên quan của các khung nhìn cụ thể hóa khác nhau.

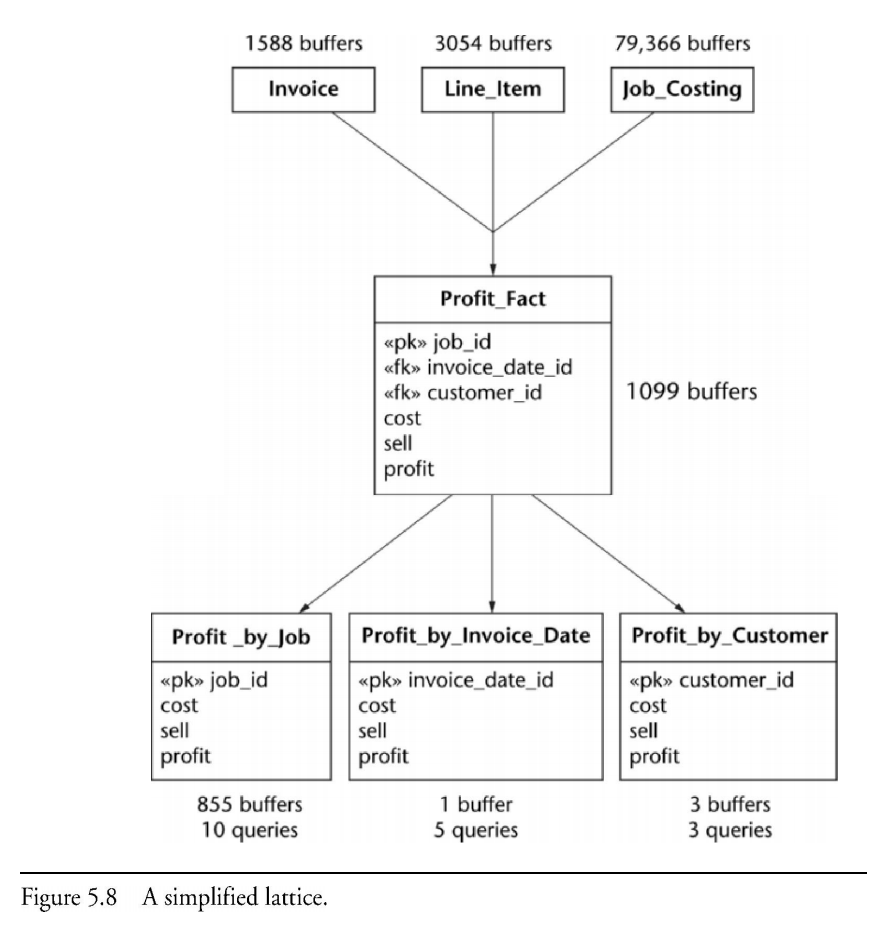
Hãy chú ý trường invoice\_date\_id 4 byte thay cho trường invoice\_date 8 byte trong hình 5.5. Số lượng các block được tính lại cho phù hợp. Bảng Profit\_Fact được tính như sau:



Lượng không gian đĩa yêu cầu để lưu trữ các khung nhìn cụ thể hóa là một hạn chế hiển nhiên. Kimball [1998, p544] có đề cập đến quy luật ngón tay cái trong ngữ cảnh của kho dữ liệu, nhiều hệ quản trị CSDL hạn chế tăng không gian cần cho khung nhìn tổng hợp. Điều này có thể khác nhau tùy vào mục đích của CSDL và tài nguyên của tổ chức. Nếu bạn đang hy vọng cải thiện hiệu suất mà không cần kho dữ liệu chính thức, bạn có thể thiết lập một giới hạn không gian đĩa vừa phải hơn. Nếu các khung nhìn mang lại lợi ích lớn nhất được chọn để khai thác đầu tiên, sau đó, cam kết các mức độ vừa phải của không gian đĩa có thể mang lại các cải tiến cho sự trả lời câu truy vấn. Cam kết lượng không gian đĩa lớn hơn sẽ mang lại sự trả lời truy vấn nhanh hơn, nhưng sự trả về thì chậm hơn. Hãy quyết định giới hạn cho số lượng của không gian đĩa.

Nếu bạn nhận ra rằng quá trình phát triển đang thúc đẩy giới hạn không gian đĩa, có nhiều lựa chọn để mở rộng tính hữu dụng trong khi bạn vẫn duy trì trong khả năng tổ chức của bạn. Ví dụ, ban quản trị quyết định rằng họ muốn theo dõi các dữ liệu tổng kết trong năm năm. Bạn có thể tiếp tục thanh lọc dữ liệu từ các bảng cơ sở sau khi lấy nửa năm, và cùng lúc đó duy trì dữ liệu của 5 năm trong các khung nhìn cụ thể hóa. Tương tự như vậy, nếu khung nhìn cụ thể hóa chứa dữ liệu trở nên quá lớn, các chi tiết có thể được thanh lọc sau một thời gian thích hợp, trong khi khung nhìn tổng hợp có thể tiếp tục giữa các lịch sử mở rộng. Ví dụ, nếu bảng Profit\_Fact trở nên lớn hơn, các dữ liệu cũ hơn 2 năm có thể được thanh lọc, trong khi bảng Profit\_by\_Invoice\_Date có thể giữ các dữ liệu có giá trị 5 năm và vẫn rất nhỏ. Sự thanh lọc của các dữ liệu chi tiết trong khi vẫn giữ trong dữ liệu tổng hợp đòi hỏi một chiến lược cập nhật phát triển.

Thời gian cần cho việc cập nhật là một hạn chế khác để ta xem xét. Thông thường, một cửa sổ cập nhật được dự trữ trong giờ nghỉ, khi các khung nhìn cụ thể hóa được cập nhật trong đĩa chuyên dụng. Hãy xác định lượng thời gian có thể cho việc cập nhật cửa sổ nếu đã áp dụng. Bạn có thể xác định một cửa sổ cập nhật là không cần thiết. Có thể tổ chức của bạn hoạt động 24 giờ 1 ngày, và hệ thống không thể bị làm cho không dùng được cho bất kì cửa sổ cập nhật nào. Có thể người dùng cần dữ liệu tổng hợp tại thời điểm hiện tại, và không thể sử dụng dữ liệu đã được tính tối qua. Cập nhật thường xuyên có thể giải quyết những vấn đề này.



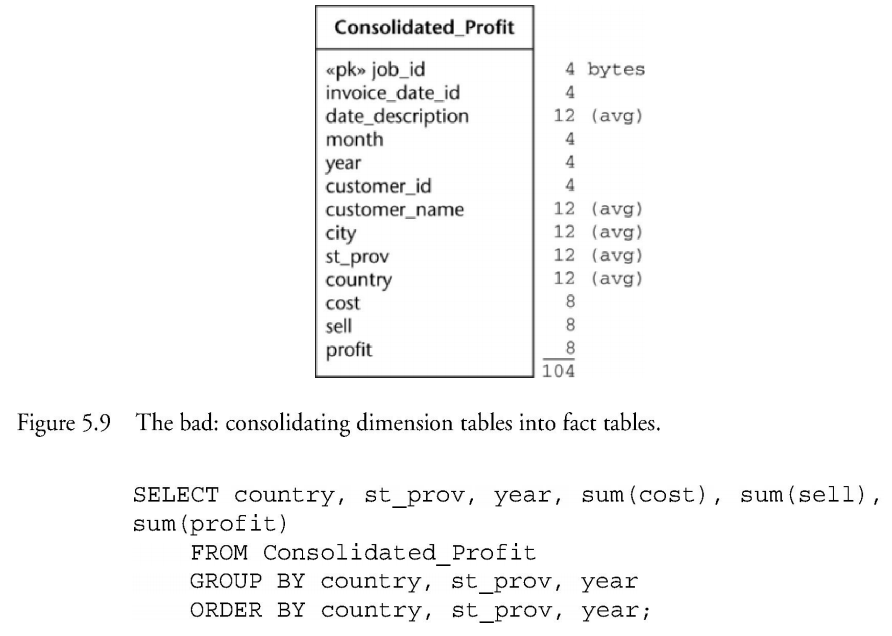
Thay vì phải tính lại khung nhìn, cập nhật tiếp tục giữ đa số các dòng của khung nhìn cụ thể hóa nguyên vẹn, và thêm các dòng mới nếu thích hợp. Ví dụ, hãy xem bảng Profit\_by\_Invoice\_Date. Hãy xem các hóa đơn được xử lú theo tiến trình hàng ngày. Thông tin được tập hợp cho các công việc đã được vận chuyển từ chu trình hóa đơn trước đó. Mỗi khách hàng có các công việc trong chu trình hiện tại có một hóa đơn giao. Các công việc của khách hàng được liệt kê trong hóa đơn. Sau khi các hóa đơn được xác minh, các hóa đơn được in và gửi cho các khách hàng, và trạng thái của mỗi hóa đơn được cập nhật trong CSDL. Khi chu trình hoàn thành, giá trị bán cho từng công việc đều được biết. Giá của mỗi công việc cũng được biết, bởi vì bảng Job\_Costing đã được cập nhật trong thời gian đó và công việc hoàn thành mỗi công việc trước khi hóa đơn được gửi. chúng tôi có thể tính tổng giá trị mỗi ngày và thêm một dòng vào bảng Profit\_by\_Invoice\_Date. Cập nhật trên khung nhìn cụ thể hóa trong thời gian thực thỉnh thoảng tương tự như phương pháp *trickle feed*.

1. **CÁC VÍ DỤ: TỐT, XẤU**

Các khung nhìn và lược đồ là cách tiếp cận tốt để giảm đĩa I/O, kết quả là câu truy vấn chạy nhanh hơn. Hình 5.6 là một ví dụ của một lược đồ sao, minh họa cách tiếp cận thiết kế không gian. Thông thường, có nhiều chiều trong một lược đồ sao, nhưng ví dụ đã đủ để thảo luận. Bảng thực tế là một khung nhìn lấy được từ các bản cơ sở. Khung nhìn chung quy là để lưu trữ các phép tính để giảm lượng đĩa I/O. Các bảng này cung cấp một phương tiện để gom nhóm dữ liệu ở các cấp độ khác nhau của cùng một mức độ. Các bảng này còn thêm tính uyển chuyển, tăng lợi ích của các bảng thực tế trong một gia đình lớn hơn của các câu truy vấn. Lược đồ sao và cách tiếp cận thiết kế không gian có thể cách tốt để cải thiện tốc độ câu truy vấn.

Một câu hỏi tự nhiên phát sinh. Nếu tính lại tiết kiệm thời gian, vậy tại sao không thực hiện lại phép kết giữa các bản thực và các bảng ảo? Liệu làm như vậy có tiết kiệm thời gian hơn không? Lược đồ trong hình 5.9 có thể là sự hợp nhất giữa bảng thực và bảng ảo từ hình 5.6. Bây giờ chúng tôi có thể truy vấn trên một bản, và chúng tôi vẫn có khả năng gom nhóm tương tự như khi chúng tôi thực hiện trên những bảng ảo.

Cùng viết lại câu truy vấn thứ 2 trong phần 5.3. Câu truy vấn bây giờ đã đơn giản hơn:



Tất cả những gì chúng ta cần làm là duyệt trên một bảng. Hãy tính xem có bao nhiêu khối được đưa ra. Có 2.000.000 dòng bởi vì có 2.000.000 công việc. Phép tính của bảng Consolidated\_Profit là:

Average rows per block (buffer)

= floor((65,536 bytes/buffer)/(104 bytes/row)) = 630.

Number of buffers = ceiling(2,000,000 rows/(630 rows/buffer)) = 3,175.

Hãy tính số dòng của một block được quét nếu lược đồ hình 5.6 được thay thế. Bảng Calendar có 200 dòng, mỗi dòng trung bình 24 bytes. Bảng Profit\_Fact có 2,000,000 dòng, mỗi dòng 36 bytes. Bảng Customer có 5,000 dòng, trung bình 64 bytes một dòng. Bảng Calendar được tính như sau:

Average rows per block (buffer)

= floor((65,536 bytes/buffer)/(24 bytes/row)) = 2,730.

Number of buffers = ceiling(200 rows/(2730 rows/buffer)) = 1.

Bảng Profit: 1,099 buffer.

Bảng Customer:

Average rows per block (buffer)

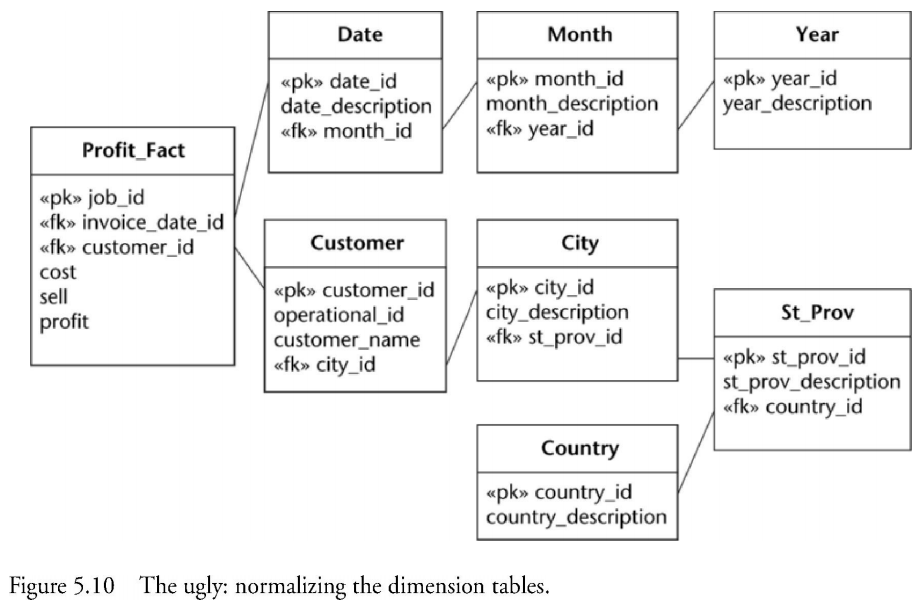
= floor((65,536 bytes/buffer)/(64 bytes/row)) = 1,024.

Number of buffers = ceiling(5,000 rows/(1024 rows/buffer)) = 5.

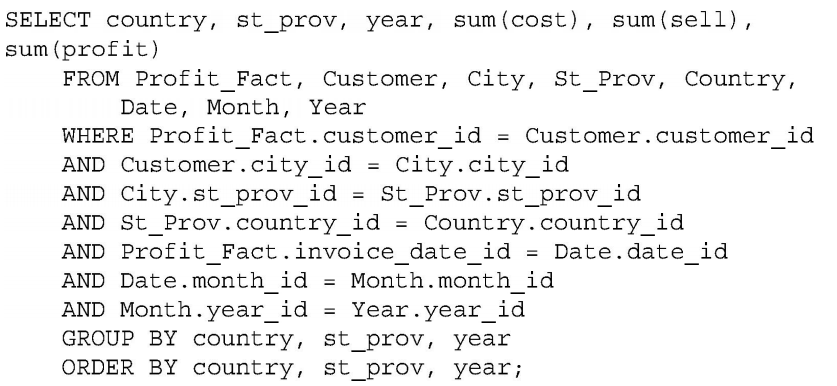
Tổng số block được quét sử dụng lược đồ sao = 1 + 1,099 + 5 = 1,105 block.

Hợp nhất bảng ảo và bảng thực thực chất làm chậm đi câu truy vấn gấp 3 lần. Sự hợp nhất bảng ảo và bảng thực là một ý kiến tồi.

Tại sao lại là bảng ảo không bình thường trong hình 5.6? Nếu hợp nhất bảng ảo với bảng thực là một ý kiến tồi, có lẽ nếu chúng tôi bình thường hóa bảng ảo thành một bảng nhỏ hơn thì chúng tôi có thể đạt được nhiều thứ hơn. Hình 5.10 cho thấy bảng ảo được bình thường hóa. Trong các tài liệu gọi nó là lược đồ hoa tuyết.



Đây là cây truy vấn của chúng tôi, được viết lại từ lược đồ trong 5.10:

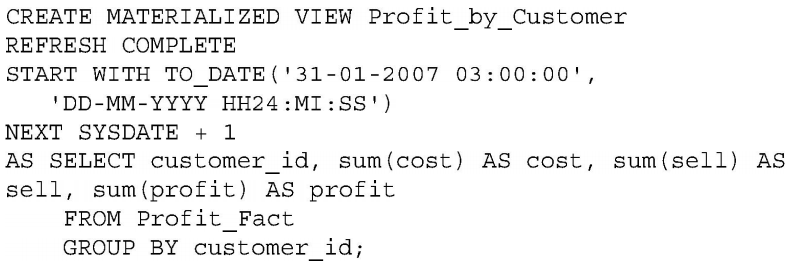


Câu truy vấn bây giờ phức tạp hơn nhiều, có tới 7 phép kết. Kimball và Ross [2002] xác định phép kết phức tạp là một lý do để tránh lược đồ hoa tuyết này. Các câu truy vấn khó viết hơn, và tối ưu hóa câu truy vấn có thể phải tìm một phương án tốt một cách khó khăn. Hơn nữa, lược đồ sao trực quan hơn lược đồ tuyết. Một vài lược đồ có thể tìm thấy các mẫu bình thường hóa của các bảng ảo trong lược đồ hoa tuyết, nhưng thường là các giải pháp rắc rối chứ không phải ngắn gọn.

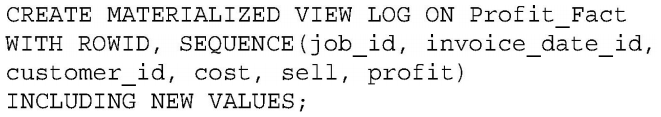
1. **CÁCH SỬ DỤNG CÚ PHÁP VÀ CÁC VÍ DỤ**

Chúng tôi minh họa một số chiến lược cập nhật có sẵn, sử dụng các ví dụ được viết trên SQL. Các ví dụ SQL trong phần này cũng sử dụng được cho Oracle 10g. Nếu bạn đang sử dụng một phiên bản khác của Oracle, kiểm tra tài liệu của bạn để có thể thay đổi.

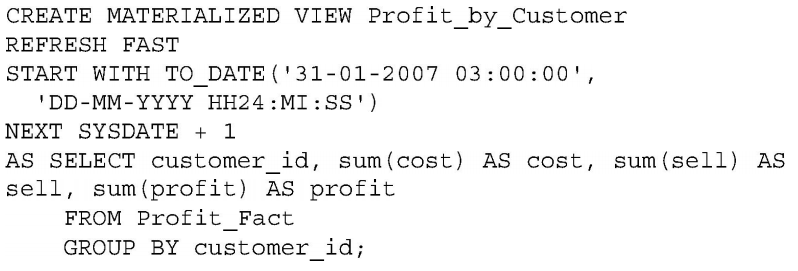
SQL trong Oracle bao gồm các khái niệm để tạo các khung nhìn và chỉ rõ chiến lược cập nhật. Nếu bạn muốn tạo một khung nhìn để lưu trữ lợi nhuận bởi các dữ liệu khách hàng, hãy dựa trên lược đồ trong hình 5.6. Nếu bạn muốn khung nhìn được nạp lại từ đầu mỗi sáng vào lúc 3 giờ bắt đầu từ ngày 31.7.2007, câu SQL sẽ được viết như sau:



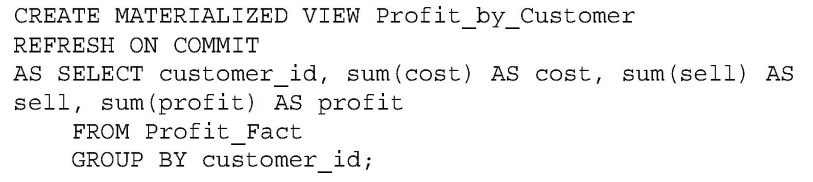
Dĩ nhiên, nạp lại hoàn toàn sẽ sử dụng nhiều tài nguyên nếu bảng Profit\_Fact có nhiều dòng. Khi bảng nguồn lớn, thường trong trường hợp là một kho dữ liệu, một lựa chọn khác khả thi hơn là sử dụng các lựa chọn nạp lại nhanh. Nạp lại nhanh là một chiến lược cập nhật lớn. Về cơ bản các dữ liệu hiện có trong khung nhìn vẫn còn nguyên vẹn, những thay đổi trên bảng nguồn luôn được theo dõi, và quá trình cập nhật tổng kết các dữ liệu mới và sau đó điều chỉnh hay tạo mới các dòng thích hợp trên khung nhìn. Nếu chúng tôi muốn sử dụng phương pháp nạp lại nhanh, sau đó chúng tôi cần ra lệnh cho Oracle theo dõi sự thay đổi trên bảng dữ liệu nguồn.



Dưới đây là sự định nghĩa cho lợi nhận bằng cách tính tổng theo khách hàng, tận dụng lợi thế của chiến lược nạp lại nhanh:



Những chiến lược khác cũng khả thi. Ví dụ như sự định nghĩa dưới đây giữ khung nhìn được cập nhật bất cứ khi nào có sự commit trên bảng Profit\_Fact.



Thông thường, các kho dữ liệu chứa nhiều bảng lớn, và các tiến trình nạp lại chạy trong lúc các cửa sổ được cập nhật. Sự lựa chọn “nhanh” hỗ trợ chiến lược này. Nếu bạn có một khung nhìn nơi một phần đáng kể của các dòng thay đổi mỗi ngày, thì giải pháp nạp lại hoàn toàn có thể sẽ chạy nhanh hơn, vì các phép tính lớn sẽ hoạt động hiệu quả hơn so với một số lượng lớn những thay đổi gia tăng. Sự lựa chọn “on commit” hỗ trợ việc cập nhật hiện thời (real-time). Nếu bạn cần dữ liệu hiện thời, hãy luôn nhớ rằng việc cập nhật khung nhìn bất cứ khi nào có một sự commit trên bảng nguồn có thể là nguồn tài nguyên lớn. Bạn có thể sẽ không được hỗ trợ nhiều loại khung nhìn như những chiến lược mà bạn đã lập trình sẵn.

Oracle có một thiết lập QUERY\_REWRITE\_ENABLE nên được cài trong tập tin thông số cấu hình để làm đầy đủ hơn mục đích của khung nhìn mà bạn tạo. Nếu câu truy vấn viết lại được bật, Oracle sau đó sẽ tự động nhận ra khi có một khung nhìn nhỏ có thể dùng để trả lời câu truy vấn, thay vì đọc nhiều bảng nguồn cơ sở lớn hơn.

1. **TỔNG KẾT**

Khung nhìn là kết quả của câu truy vấn được lưu trữ trong đĩa. Khung nhìn lưu trữ có hiệu quả kết quả tính toán, cho phép tái sử dụng kết quả, làm cho câu truy vấn ra kết quả nhanh hơn. Số lượng các khung nhìn có thể có thường rất lớn, yêu cầu người thiết kế phải liên tục lựa chọn các chiến lược của khung nhìn. Sự tìm hiểu tốt sẽ giúp tập trung vào những câu truy vấn thường xuyên và chiếm nhiều dung lượng đĩa I/O. Mỗi câu truy vấn sẽ tự ánh xạ lên một khung nhìn. Các đồ thị kết quả và các cấu trúc mạng có thể chứa các mối quan hệ giữa các khung nhìn, tạo điều kiện thuận lợi cho sự lựa chọn các khung nhìn để sử dụng. Những khung nhìn tốt là khung nhìn tự nhiên của những câu truy vấn thường xuyên, và cũng là hình thức ban đầu chung của những khung nhìn đó. Các lược đồ thường là mô hình tốt để làm theo và tăng khả năng sử dụng của các bảng trong một tập lớn các câu truy vấn. Việc lựa chọn các khung nhìn thông thường theo cách tiếp cận tham lam, chọn khung nhìn thuận lợi nhất ở mỗi bước, cho đến khi đạt đến sự hạn chế tài nguyên. Sự hạn chế tài nguyên có thể bao gồm số lượng khung nhìn, sự sử dụng không gian đĩa và thời gian yêu cầu để duy trì khung nhìn.

### Topic #7 – Các kết hoạch thực thi truy vấn và thiết kế vật lý

Mục lục

[Dẫn nhập 50](#_Toc309759931)

[I. Tập kết quả từ chuỗi câu truy vấn 50](#_Toc309759934)

[II. Kế hoạch thưc thi câu truy vấn 54](#_Toc309759934)

[III. Giải thích không trực quan: 55](#_Toc309759935)

[IV. Tìm hiểu kế hoạch thực thi câu truy vấn và cải tiến thiết kế vật lý 60](#_Toc309759936)

[V. Chỉ số kế hoạch thực thi và cải tiến thiết kế vậy lý 67](#_Toc309759936)

[VI. Tìm hiểu cách tối ưu mà không làm thay đổi thiết kế 70](#_Toc309759936)

[VII. Các vấn đề bất buộc khi tối ưu hóa truy vấn 71](#_Toc309759936)

VIII. Tóm tắt

**DẪN NHẬP**

Trong suốt cuốn sách này, chúng tôi đã trình bày chủ yếu về phần dữ liệu mức vật lý, cách sử dụng và tại sao lại có nó. Làm thế nào bạn có thể xác định được một thuộc tính thiết kế một cơ sở dữ liệu riêng biệt thực sự giúp ích hay đang phá hoại cơ sở dữ liệu của bạn? Về nguyên tắc, làm thế nào bạn có thể xác định nếu một tính năng thiết kế được sử dụng vào cơ sở dữ liệu có là phương pháp thuật toán của khóa học: Chạy các thí nghiệm với khối lượng công việc ứng dụng của bạn để đánh giá sự đóng góp của tất cả các thay đổi thiết kế vật lý đơn lẻ để xác định tác động Loại chiến lược thuật toán có lẽ là không hiệu quả vì nó có nghĩa là thực sự tạo ra và thử nghiệm mỗi khả năng thiết kế với một cơ sở dữ liệu thực sự đóng gói với dữ liệu lớn. Có một cách khác để đánh giá lợi ích của việc lựa chọn thiết kế, đó là cùng một phương pháp mà bản thân cơ sở dữ liệu sử dụng, cụ thể để kiểm tra tác động của sự lựa chọn thiết kế các kế hoạch thực hiện truy vấn của khối lượng công việc của bạn. Trong thực tế ký thuật này là kỹ thuật chủ đạo được sử dụng bởi các quản trị cơ sở dữ liệu (DBA) và thiết kế cơ sở dữ liệu cho mỗi sản phẩm cơ sở dữ liệu lớn, và kết quả là mỗi sản phẩm cơ sở dữ liệu lớn cung cấp dụng cụ cho phép người dùng kiểm tra kế hoạch thực hiện truy vấn được lựa chọn bởi cơ sở dữ liệu cho mỗi truy vấn. Xem kế hoạch thực hiện truy vấn, người ta có thể quan sát xem cơ sở dữ liệu để lựa chọn việc thiết kế, chẳng hạn như quan điểm về khung nhìn và chỉ mục, có vẻ có ích Làm thế nào vậy?Bởi vì nếu cơ sở dữ liệu cho rằng tính năng thiết kế là hữu ích, nó sẽ khai thác nó trong kế hoạch thực hiện truy vấn . Ví dụ, nếu một chỉ mục A mang lại lợi ích cho một truy vấn trong thời gian chạy, chúng ta sẽ thấy truy cập chỉ mục trong kế hoạch thực hiện truy vấn cho truy vấn tốt hơn. Trong chương này, chúng tôi sẽ cho thấy các ví dụ về các cách khác nhau về kế hoạch một câu truy vấn, sẽ được trình bày trong văn bản và hình trực quan bởi các nhà cung cấp cơ sở dữ liệu chính, và minh họa làm thế nào họ có thể giúp cải thiện thiết kế cơ sở dữ liệu vật lý.

1. **Tập kết quả từ chuỗi câu truy vấn**

Làm thế nào để truy vấn đi vào một cơ sở dữ liệu có được chuyển đổi thành một bộ câu trả lời? Hình 11.1 cho thấy trình tự cơ bản của hoạt động xảy ra cho đến các câu lệnh SQL. Khi một truy vấn được gửi đến một cơ sở dữ liệu, điều đầu tiên cơ sở dữ liệu sẽ làm là phân tích các dòng lệnh để bắt lỗi và sau đó xây dựng một đại diện nội bộ của các truy vấn thường được lưu trữ trong một mô hình đồ thị mà người dùng không thể nhìn thấy .

Sau đó, câu truy vấn này được kiểm tra về tính chính xác ngữ nghĩa để đảm bảo các truy vấn có ý nghĩa. Ví dụ, tất cả các tài liệu tham khảo của đối tượng trong truy vấn tồn tại, và có thể truy vấn ngay cả khái niệm được thực hiện (thậm chí với bộ dữ liệu sau khi truy vấn được là NULL). Trong giai đoạn này CSDL cũng có thể mở rộng đồ thị truy vấn bằng cách thêm các điều kiện logic để kiểm tra ràng buộc toàn vẹn hay trigger. Sau khi kiểm tra ngữ nghĩa, truy vấn sau đó được viết lại bởi cơ sở dữ liệu để nâng cao có tính hiệu quả hơn. Dưới đây là một ví dụ sử dụng truy vấn với tiêu chuẩn ngành công nghiệp TPC-R.

select l\_returnflag, l\_linestatus, sum(l\_quantity) as

sum\_qty,

sum(l\_extendedprice) as sum\_base\_price,

sum(l\_extendedprice \* (1 - l\_discount)) as

sum\_disc\_price, sum(l\_extendedprice \* (1 - l\_discount) \*

(1 + l\_tax)) as sum\_charge, avg(l\_quantity)as avg\_qty,

avg(l\_extendedprice) as avg\_price,

avg(l\_discount) as avg\_disc,

count(\*) as count\_order

from tpcd.lineitem

where l\_shipdate <= date ('1998-12-01') - 90 day

group by l\_returnflag, l\_linestatus

order by l\_returnflag, l\_linestatus

Dưới đây cũng là một tập hợp các câu lệnh được viết lại bởi một cơ sở dữ liệu mới:

SELECT Q3.$C7 AS “L\_RETURNFLAG”,

Q3.$C6 AS “L\_LINESTATUS”,

Q3.$C5 AS “SUM\_QTY”,

Q3.$C4 AS “SUM\_BASE\_PRICE”,

Q3.$C3 AS “SUM\_DISC\_PRICE”,

Q3.$C2 AS “SUM\_CHARGE”,

(Q3.$C5 / Q3.$C0) AS “AVG\_QTY”,

(Q3.$C4 /Q3.$C0) AS “AVG\_PRICE”,

(Q3.$C1 / Q3.$C0) AS “AVG\_DISC”,

INTEGER(Q3.$C0) AS “COUNT\_ORDER”

FROM

(SELECT SUM(Q2.$C2), SUM(Q2.$C3), SUM(Q2.$C4),

SUM(Q2.$C5),SUM(Q2.$C6),SUM(Q2.$C7), Q2.$C0, Q2.$C1

FROM

(SELECT Q1.L\_LINESTATUS, Q1.L\_RETURNFLAG, Q1.COUNT,

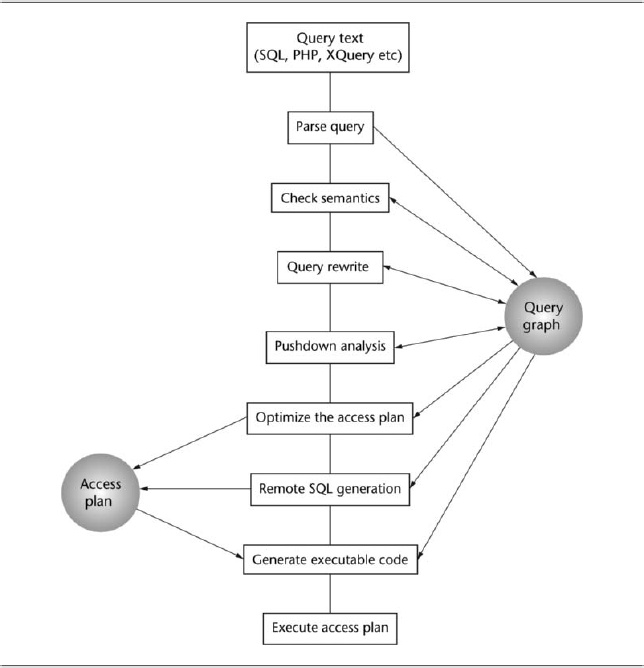
Q1.S5, Q1.S4, Q1.S3, Q1.S2, Q1.S1

FROM TPCD.L\_SUMMARY AS Q1

WHERE (Q1.L\_SHIPDATE <= '09/02/1998')) AS Q2

GROUP BY Q2.$C1, Q2.$C0) AS Q3

ORDER BY Q3.$C7, Q3.$C6



Hình 11.1 Việc lập quá trình thực hiện truy vấn.

Các truy vấn viết lại là một chút khó khăn để đọc, bởi vì nó sử dụng các đại diện bên trong của các bảng và cột thay vì người dùng xác định tên bình thường mà các nhà thiết kế cơ sở dữ liệu quy định tại DDL.Không sao, bởi vì các phiên bản viết lại của câu truy vấn chỉ là một bước đệm trong quá trình xử lý nội bộ của các truy vấn, trước khi lựa chọn kế hoạch thực hiện truy vấn. Nó không được thiết kế chủ yếu cho con người để đọc. Tuy nhiên, nhiều nhà cung cấp cơ sở dữ liệu cho phép người dùng xem các truy vấn viết lại trực quan thông qua các công cụ thực hiện kế hoạch truy vấn.Điều này có thể hữu ích trong việc xác định nếu một vấn đề hiệu suất đã được gây ra bởi cơ sở dữ liệu trong giai đoạn này viết lại.

Có thể trong quá trình viết lại truy vấn các trình biên dịch truy vấn cơ sở dữ liệu có thể tìm thấy như một trận đấu tốt cho một quan điểm vật chất là một trong là lựa chọn và thay thế vào các truy vấn viết lại, mặc dù hầu hết các nhà sản xuất rời khỏi các lựa chọn cuối cùng quan điểm vật chất cho các bước tối ưu hóa truy vấn đa số các thời gian.

Cuối cùng, một kế hoạch thực hiện truy vấn có thể được lựa chọn dựa trên các dữ liệu kết hợp trong đồ thị truy vấn . Điều này sẽ bao gồm không chỉ các truy vấn văn bản viết lại, danh sách các đối tượng, cột truy cập, các chỉ mục có sẵn, và quan điểm vật chất và các thuộc tính thiết kế vật lý khác, mà còn thống kê hiện có trên đối tượng (bảng, xem vật chất, và thống kê chỉ mục) và thông tin về các nguồn tài nguyên vật lý của máy chủ cơ sở dữ liệu, chẳng hạn như tốc độ CPU, tốc độ truy cập thiết bị lưu trữ, tốc độ và độ trễ mạng và. Như đã thảo luận trong Chương 3, quá trình lựa chọn kế hoạch thực hiện truy vấn phức tạp và có thể liên quan đến việc thay đổi mức độ phức tạp tìm kiếm. Hầu như luôn luôn cách nhiều kết quả thiết lập cho một truy vấn có thể được xác định, và cơ sở dữ liệu sẽ cố gắng trong quá trình tối ưu hóa truy vấn để lựa chọn kế hoạch thực hiện truy vấn có thể lấy kết quả thiết lập một cách hiệu quả nhất. Cơ sở dữ liệu khác nhau có những mục tiêu khác nhau để xác định "hiệu quả nhất", mặc dù lựa chọn kế hoạch thực hiện truy vấn với tiêu thụ tài nguyên thấp nhất dường như là một chiến lược chung.

Tuy nhiên, kế hoạch thực hiện truy vấn chính nó vẫn là một khái niệm trừu tượng . Bước cuối cùng là yêu cầu trước khi lấy các bộ dữ liệu, được gọi là thế hệ mã . Quá trình này tạo ra các mã thực thi cần thiết để thực hiện các nhiệm vụ được đề nghị bởi các kế hoạch thực hiện truy vấn được xác định bởi quá trình tối ưu hóa truy vấn. Mã thế hệ sẽ chuyển đổi các ký hiệu trừu tượng của kế hoạch thực hiện truy vấn vào mã đối tượng được biên soạn cơ sở dữ liệu có thể hiểu và thực hiện.

Cuối cùng, các mã đối tượng sản xuất bởi các giai đoạn thế hệ mã được thực thi, bao gồm cả các hoạt động để xử lý chỉ mục, gia nhập, các loại, vv, và các kết quả truy vấn được viết ra và truyền đạt lại cho các ứng dụng thông qua một quá trình giao tiếp, chẳng hạn như TCP / IP. Không thể tin nổi, toàn bộ quá trình xử lý giao dịch trực tuyến(OLTP) có thể thực hiện trong các phần của một giây.

1. **Kế hoạch thực thi câu truy vấn**

Từng nhà cung cấp có định dạng độc quyền của nó cho việc hiển thị các kế hoạch thực hiện truy vấn và các thông tin cụ thể này bao gồm . Từng nhà cung cấp có sở dữ liệu có ịnh đạng bản quyền cho việc thiết kế kế hoạch truy vấn và chứa một số thông tin cụ thể bao gồm. Tuy nhiên, mặc dù các chi tiết khác nhau nhưng chủ đề cơ bản là như nhau. Kế hoạch thực hiện truy vấn cho một truy vấn được hiển thị như một chỉ dẫn biểu đồ, với cả các đối tượng cơ sở dữ liệu và điều hành kế hoạch thực hiện truy vấn được hiển thị như các nút đồ thị. Đối tượng sẽ bao gồm các bảng biểu, chỉ mục, quan điểm vật chất, danh mục, bảng tạm thời, vv… Các loại phép kết (và kiểu của chúng, kết theo bảng bam,kết theo vòng lặp, kết nữa, kết tự nhiên), sắp xếp , vv ... Tiện ích Oracle được gọi là kế hoạch giải thích (Hình 11.2) và giải thích bằng hình trực quan. Các tiện ích microsoft SQL Server được gọi là phân tích truy vấn, và các tiện ích DB2 được gọi là giải thích trực quan (Hình 11.3) .

1. **Giải thích không trực quan**

Oracle cũng cung cấp văn bản giải thích rằng cho phép các quản trị viên để xem đường dẫn truy cập cho các truy vấn mà không cần sử dụng giao diện người dùng đồ họa (GUI). Sử dụng của nó khá đơn giản, bằng cách sử dụng các câu lệnh SQL để giải thích kế hoạch văn bản như sau:

EXPLAIN PLAN FOR

<SQL statement text>

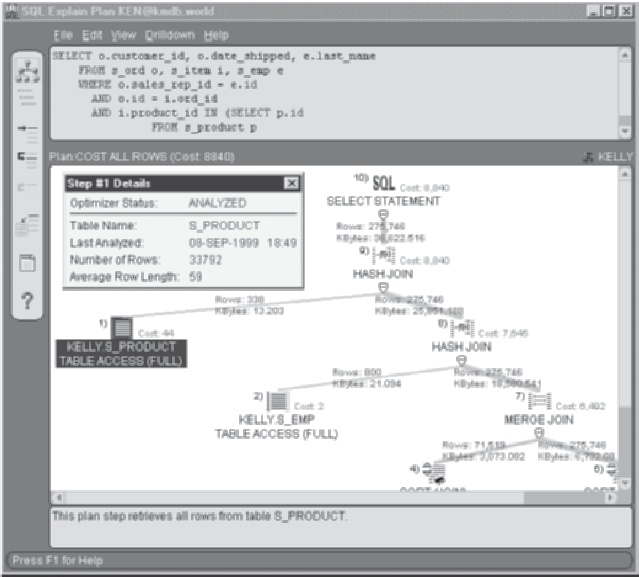
Điều này tạo ra sản lượng Giải thích và lưu trữ nó trong bảng quản trị viên PLAN\_TABLE sau đó có thể chọn kế hoạch thực hiện từ PLAN\_TABLE để xem đường dẫn truy cập.  
Nếu bạn muốn giải thích nhiều câu lệnh SQL và lưu trữ các đường dẫn truy cập cho tất cả chúng (hoặc một số đường dẫn trong đó), bạn có thể chỉ định một định danh tuyên bố về kế hoạch giải thích CHO cú pháp để xác định kế hoạch thực hiện cụ thể cho mỗi câu. Ví dụ:

EXPLAIN PLAN

SET STATEMENT\_ID = <identifier> FOR

<SQL Statement>

Ví dụ sau đây minh họa dựa trên văn bản giải thích của Oracle, và cho thấy Oracle có định dạng đầu ra.

  
Hình 11.2 Oracle Giải thích cho một kế hoạch thực hiện truy vấn. (Hình ảnh lịch sự của Oracle Corp. )

EXPLAIN PLAN SET statement\_id = 'Oracle-example1' FOR

SELECT arrival\_date, name, description, inventory\_id

FROM product\_stock

WHERE arrival\_date = :b1 AND name LIKE '%-COM'

AND NVL(end\_date\_active,sysdate+1) > SYSDATE ;

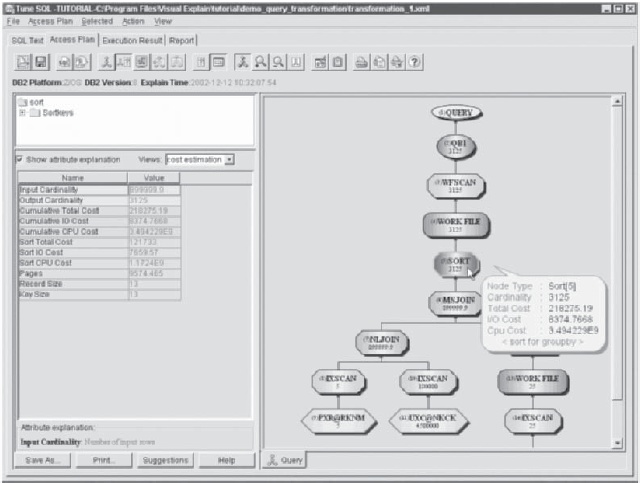
Kế hoạch

--------------------------------------------------

SELECT STATEMENT

TABLE ACCESS BY INDEX ROWID PRODUCT\_STOCK

INDEX RANGE SCAN STOCK\_ITEMS



Hình 11.3 DB2 Visual Giải thích, zOS phiên bản. (Hình ảnh lịch sự của IBM Corp. )

EXPLAIN PLAN SELECTION SET QUERYTAG = 'DB2-example1'

FOR

SELECT arrival\_date, name, description, inventory\_id

FROM product\_stock

WHERE arrival\_date = :b1AND name LIKE '%-COM'

AND NVL(end\_date\_active,sysdate+1) > SYSDATE ;

Cú pháp gần như là giống nhau, nơi mà Oracle sử dụng statement\_id DB2 sử dụng QUERYTAG, và DB2 bao gồm các lựa chọn từ khóa sau đây giải thích kế hoạch.

Ngoài ra, cả Oracle và DB2 có thể được cấu hình để giải thích tất cả các truy vấn mới đến cho phiên hiện tại. Đối với DB2 này được thực hiện bằng cách đặt các phiên vào. Như:

SET CURRENT EXPLAIN MODE YES

or

SET CURRENT EXPLAIN MODE EXPLAIN

Oracle SET AUTOTRACE Về tương tự như DB2’s SET CURRENT EXPLAIN MODE YES

Sau nắm bắt kế hoạch trong DB2 lệnh có thể được sử dụng để định dạng giải thích dữ liệu văn bản, tạo ra sản lượng văn bản với một cấu trúc cây tương tự như cấu trúc thể hiện qua các giao diện đồ họa của hầu hết các sản phẩm. ? Dưới đây là một ví dụ cho truy vấn sau đây:

SELECT year(dateofsale), avg(sales), color

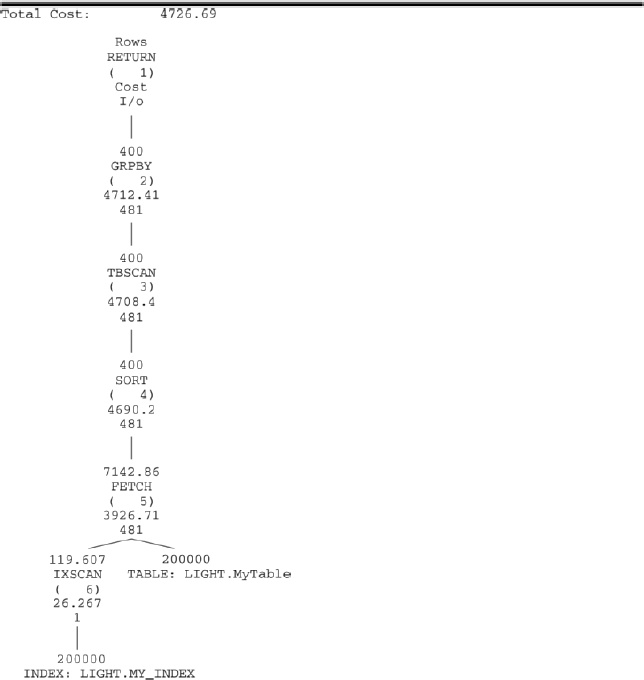
FROM LIGHT.MyTable

WHERE REGION = 'NY'

GROUP BY (COLOR, YEAR(DATEOFSALE))

Giải thích đầu ra cho truy vấn này được thể hiện trong Hình 11.4. Bạn sẽ nhận thấy việc sử dụng các từ khóa để mô tả các nhà khai thác trong kế hoạch . Họ, hầu hết các phần, khá intuitive. ví dụ, GROUPPBY được sử dụng để chỉ một nhóm, TBSCAN được sử dụng cho bảng quét, etc.Chỉ số xuất hiện trong ngoặc đơn là một định danh cho mỗi nhà khai thác, và chi tiết báo cáo sản xuất Giải thích cung cấp thông tin nhiều hơn cho mỗi nhà khai thác, đó là danh mục phù hợp. Hai con số xuất hiện phía bên dưới mỗi nhà khai thác chi phí tổng và chi phí đầu vào / đầu ra (I / O ) tương ứng.

Có một số sự khác biệt rõ ràng giữa đầu ra dựa trên văn bản của hai sản phẩm. Hệ thống DB2 mang lại một cấu trúc cây rất giống với cấu trúc cây được sử dụng giao diện đồ họa trong hầu hết các sản phẩm. Oracle của đầu ra là hơn bảng, bằng cách sử dụng một cấu trúc nestnext để hiển thị các mối quan hệ thứ bậc. DB2 rất dễ dàng để làm theo, nó cung cấp các đường dẫn truy cập hẹp đủ để thực thi trong một trình soạn thảo văn bản (thường là 80-120 kí tự). Tuy nhiên, đối với kế hoạch rất rộng các định dạng Oracle vẫn nhỏ gọn trong khi định dạng DB2 có kết quả đẹp và có thể khó khăn hơn để xem trong văn bản.Rõ ràng, mỗi định dạng có lợi ích của nó.



Hình 11.4 Giải thích đầu ra trong văn bản.

1. **Tìm hiểu kế hoạch thực thi câu truy vấn và cải tiến thiết kế vật lý**

Ví dụ sau đây nêu bật một số các thuộc tính điển hình của một thực hiện kế hoạch truy vấn , và mặc dù ví dụ này sử dụng DB2 tương tự như một kế hoạch thực hiện truy vấn, hiểu biết sâu sắc điển hình của các nhà cung cấp chủ yếu với các công cụ tương ứng của họ . Trong ví dụ này, chúng tôi sẽ xem xét tại một đoạn của một kế hoạch thực hiện truy vấn cho một truy vấn phức tạp . Các truy vấn là truy vấn # 20from TPC-H tiêu chuẩn công nghiệp tiêu chuẩn (xem www.tpc.org). Các truy vấn đầy đủ như sau:

 select s\_name, s\_address

from tpcd.supplier, tpcd.nation

where

s \_suppkey in (

selectp s\_suppkey

from tpcd.partsupp

wherep s\_partkey in

(select p\_partkey

From tpcd.part

Where p\_name like ':1%')

and

ps\_availqty > (

select 0.5 \* sum(l\_quantity)

from tpcd.lineitem

where l\_partkey = ps\_partkey

and l\_suppkey = ps\_suppkey

and l\_shipdate >= date (':2')

and l\_shipdate < date (':2') + 1 year

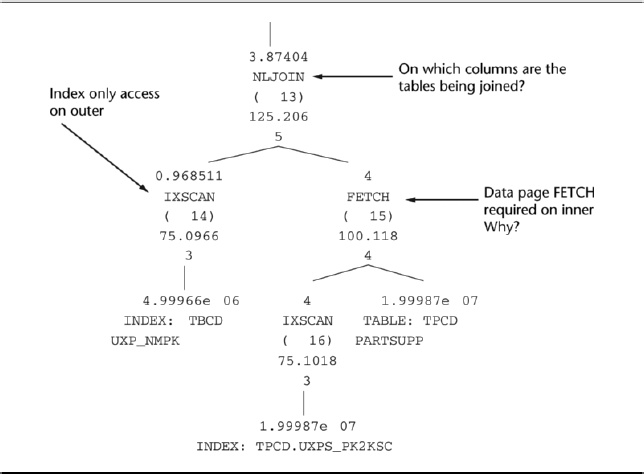
))

and s\_nationkey = n\_nationkey

and n\_name = ':3'

order by s\_name;

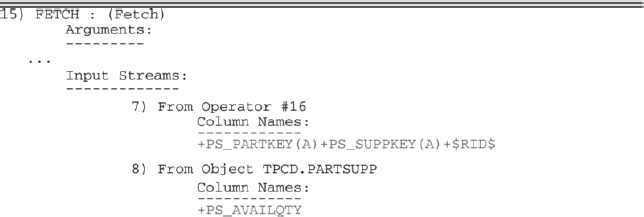
Thực hiện kế hoạch truy vấn đầy đủ cho câu lệnh này liên quan đến nhau, nhưng chúng ta có thể sử dụng một đoạn của kế hoạch thực hiện truy vấn để làm nổi bật những ý tưởng chính . Hình 11.5 cho thấy một đoạn của kế hoạch thực hiện truy vấn thực hiện một tham gia vòng lặp lồng nhau giữa các bảng các bộ phận và bảng PARTSUPP (một phần nhà cung cấp ). Kế hoạch thực hiện truy vấn



Hình 11,5 kế hoạch thực hiện trong định dạng văn bản cho các bộ phận bảng với một bảng nhà cung cấp (Hình ảnh lịch sự của IBM .)

đồ thị nên được đọc từ dưới lên và từ trái sang phải.Các lá của đồ thị cho thấy các đối tượng đang bị truy cập.Điều bất thường đầu tiên bạn sẽ nhận thấy là bảng PARTS không xuất hiện bất cứ nơi nào trong kế hoạch này. Thay vào đó, một chỉ mục trên bảng PARTS được gọi là TPCD.UXP\_NMPK được sử dụng! Tiết kiệm cơ sở dữ liệu cần truy cập vào bảng PARTS !Tiếp theo, bạn sẽ thấy từ khóa IXSCAN mà là biểu tượng DB2 cho chỉ số quét. Điều này cho chúng ta biết rằng TPCD.UXP\_NMPK chỉ số đang được quét cho các phím đủ điều kiện được thông qua NLJN (13). Bạn có thể đoán mà NLJN là viết tắt của phép kết vòng lặp.Số (13) trong ngoặc đơn là một định danh. Như chúng ta sẽ thấy, ở những nơi khác trong đầu ra Giải thích có nhiều thông tin về mỗi nút số trong đồ thị.

Kế hoạch thực hiện truy vấn rõ ràng cho thấy rằng những thông tin cho PARTSUPP  
yêu cầu truy cập TPCD.UXPS\_PK2KSC và chỉ số bảng các PARTSUPP  
chính nó. Chúng ta có thể xác định lý do tại sao? Nhìn vào các thông tin bổ sung cho nhà điều hành (15) xuất hiện trong đầu ra Giải thích, và được sao chép ở đây trong hình 11,6, chúng ta có thể nhìn thấy FETCH đòi hỏi phải truy cập vào bảng PARTSUPP bởi vì chỉ số bao gồm PS\_PARTKEY và PS\_SUPPKEY cột, nhưng không bao gồm cột PS\_AVAILQTY. Điều này cho thấy mạnh mẽ rằng bằng cách thêm cột PS\_AVAILQTY chỉ số này chúng ta có thể tránh truy cập vào các bảng PARTSUPP trong subplan, qua đó cải thiện hiệu suất.



Hình 11,6 Thông tin chi tiết Giải thích về toán tử # 15.(Hình ảnh lịch sự của IBM).

Tiếp theo, chúng tôi sẽ kiểm tra việc sử dụng một cái nhìn hiện thực (xem Chương 5) để chứng minh rằng những thay đổi đường dẫn truy cập và làm thế nào chúng ta có thể sử dụng đường dẫn truy cập để xác nhận xem vật chất mới đều được sử dụng bởi cơ sở dữ liệu trong thời gian chạy và cung cấp giá trị.

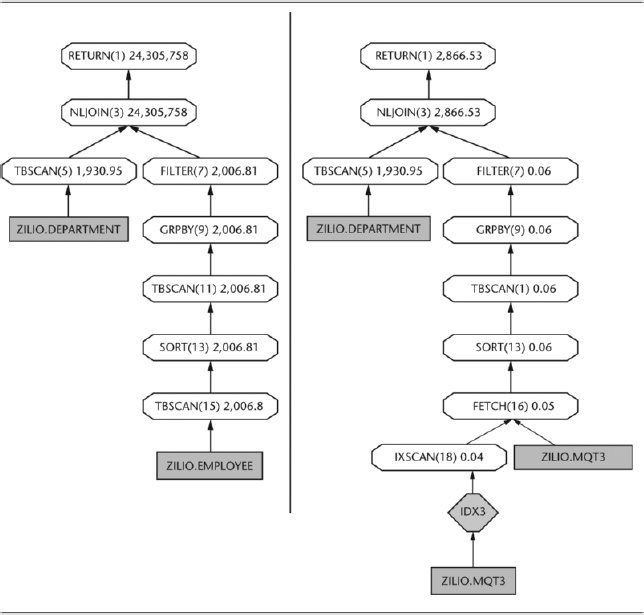
select \* from department where 10 in

(select count(\*) from employee where admrdept = workdept and salary > 1000 group by job);

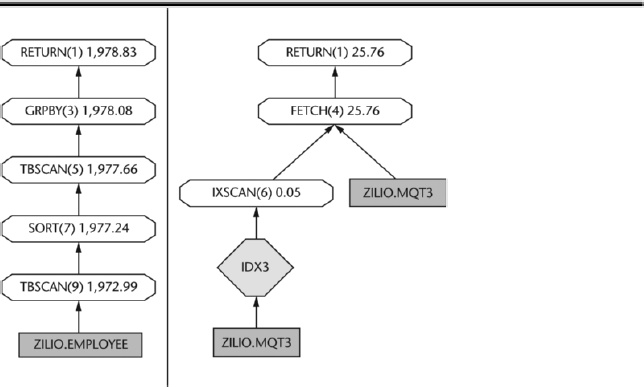
Khung nhìn thực sự về thiết kế cơ sở dữ liệu (bằng cách sử dụng cú pháp DB2):

* Tạo một khung nhìn có tên là MQT3, tạo bảng MQT3 như (select workdept, job, count(\*) as ct from employee where salary > 1000 group by job, workdept) dữ liệu mới ngay lập tức chuyển đến userspace1.
* Tạo một chỉ mục trên thực thể, trên cột workdept; createindex IDX3 MQT3 (workdept).
* Cập nhật các số liệu thống kê cho khung nhìn thực hiện trong danh mục hệ thống; xem số liệu thống kê trong bảng MYSCHEMA.MQT3 và lập chỉ mục cho tất cả; xác nhận .

Hình 11,7 cho thấy các đường dẫn truy cập trước và sau khi tạo khung nhìn. Trong ví dụ này, chúng tôi đang sử dụng công cụ Visual Explain của hệ quản trị DB2 . Trong định dạng này, mỗi node trong đường dẫn truy cập được gán một số trong dấu ngoặc xác định các nhà điều hành cho biết thêm chi tiết. Con số thứ hai bên cạnh mỗi nhà khai thác là một chi phí gần đúng cho các nhà điều hành và tất cả các toán tử phụ trong các chi nhánh của con đường truy cập.



Hình 11.7Query thực hiện kế hoạch (phải) và không có (bên trái) xem vật chất.



Hình 11.8Query thực hiện kế hoạch trước (trái) và sau (bên phải) MQT3 để truy vấn thứ hai.

kế hoạch cho thấy một bảng quét trên bàn DEPARTMENT xuất hiện để tiêu tốn một tỷ lệ đáng kể của tổng tiêu thụ tài nguyên cho các truy vấn.Nếu các hàng đủ điều kiện từ DEPARTMENT là một tỷ lệ phần trăm nhỏ của kích thước bảng tổng số (dưới 50%), một chỉ số trên DEPARTMENT có thể làm giảm chi phí truy cập.

Giá trị lớn các khung nhìn trực quan là một cái nhìn duy nhất có thể cung cấp lợi ích cho nhiều báo cáo truy vấn. Hãy xem xét xem vật chất cùng một MQT3 trên với một truy vấn mới:

select workdept, job, count(\*) as ct from employee

where salary > 1000 group by job, workdept;

Trước và sau khi kế hoạch thực hiện truy vấn được thể hiện trong hình 11,8. Một lần nữa, chúng ta thấy một sự khác biệt hiệu suất rất lớn giữa trước và sau khi tiêu thụ tài nguyên: 1.979 so với 26 đơn vị, cho thấy 76 lần giảm tiêu thụ tài nguyên và tăng tốc tương ứng với tiềm năng thực hiện truy vấn. Hai điểm rất quan trọng đáng chú ý: đầu tiên, cùng xem vật chất là hữu ích cho hai truy vấn khá khác nhau, và thứ hai, rằng trong cả hai trường hợp các văn bản của các truy vấn không có tham chiếu đến xem vật chất MQT3 . Trong cả hai trường hợp phát hiện MQT3 hữu ích và quyết định để khai thác quan điểm vật chất trong đường dẫn truy cập đã được thực hiện hoàn toàn bằng các trình biên dịch truy vấn cơ sở dữ liệu mà không có sự tham gia của các nhà thiết kế ứng dụng người đã soạn thảo các văn bản truy vấn.

Trong ví dụ cuối cùng, chúng ta hãy xem xét một kế hoạch thực hiện truy vấn cho một truy vấn duy nhất với và không có phân nhóm đa chiều (MDC ). Ví dụ này xem xét một truy vấn phạm vi rất đơn giản trên cột tiền lương trong một bảng EMPLOYEE . Người gọi có thể được tìm kiếm cao nhất hoặc thấp nhất nhân được trả lương, ví dụ, hoặc có thể tìm kiếm cho người lao động được thanh toán trong thời hạn một phạm vi nhất định, chẳng hạn như $ 50,000 - $ 60,000 mỗi năm. Hai kế hoạch thực hiện truy vấn được thể hiện trong hình 11,9.

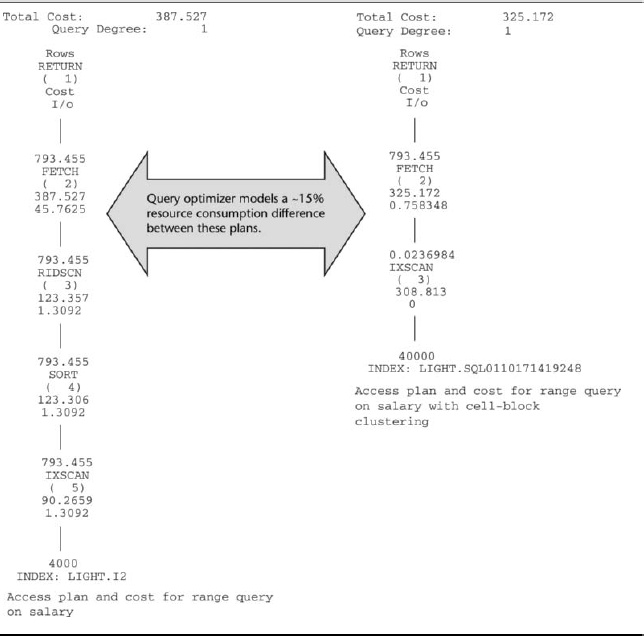
Kế hoạch thực hiện truy vấn đầu tiên là lựa chọn khi các cơ sở dữ liệu không sử dụng MDC, nhưng không có một chỉ mục trên các cột LƯƠNG của bảng EMPLOYEE . Kế hoạch thực hiện truy vấn khá đơn giản. Đây là một prefetch danh sách kế hoạch, lần đầu tiên được định danh kỷ lục (RIDs) của các hàng đủ điều kiện từ chỉ số. Các RIDs sau đó được sắp xếp (theo thứ tự trang dữ liệu) và danh sách được sắp xếp của RIDs được gửi đến prefetchers, sẽ nhận được tất cả các trang vào vùng bộ đệm (hy vọng) vào thời điểm chúng ta bắt đầu chế biến các hoạt động FETCH. Tổng tiêu thụ tài nguyên được ước tính bằng cách tối ưu hóa truy vấn là 387,527 đơn vị, mà là một sự pha trộn của I / O, CPU, và mạng lưới tiêu thụ kế hoạch thực hiện truy vấn sẽ yêu cầu . Trong ví dụ thứ hai, cơ sở dữ liệu đã được sửa đổi với các phân nhóm đa chiều về kích thước LƯƠNG của bảng EMPLOYEE. Một cột được tạo ra đã được bổ sung định nghĩa là INT (SALARY/1000) để phạm vi tiền lương sẽ được nhóm lại với nhau trong các khối MDC (xem Chương 8, các điều khoản areexplained ).

Sự khác biệt rõ ràng nhất trong kế hoạch thực hiện truy vấn kết quả cho truy vấn cùng một phạm vi trên cơ sở dữ liệu với MDC là kế hoạch kết quả rõ ràng là đơn giản, với những bước đi ít hơn và các nhà khai thác. Kế hoạch thực hiện truy vấn lựa chọn các khối từ một chỉ số khối được tạo ra trong nội bộ được gọi là SQL0110171419248, và các hàng đủ tiêu chuẩn vẫn đang được lấy và quay trở lại, đã được sắp xếp bởi các tế bào. Tối ưu hóa ước tính tiêu thụ tài nguyên cho các kế hoạch mới để được 325,172 đơn vị, khoảng 15% tiêu thụ ít tài nguyên hơn so với kế hoạch đầu tiên. Điều này cho chúng ta thấy rằng sự lựa chọn thiết kế cho MDC xuất hiện hy vọng sẽ mang lại lợi ích cho truy vấn này, quantifiably bao nhiêu, và cũng có thể cung cấp cho chúng tôi đảm bảo rằng các thuộc tính thiết kế sẽ được sử dụng bởi các truy vấn trong thời gian chạy.

1. **Chỉ số kế hoạch thực thi và cải tiến thiết kế vậy lý**

Có các chỉ số xác định trong một kế hoạch thực hiện truy vấn có thể báo hiệu một vấn đề với (hoặc có lẽ là một cơ hội để cải thiện) thiết kế cơ sở dữ liệu vật lý. Dưới đây là một số ví dụ về các tình huống để xem ra cho những điều sau đây:

1. Bảng quét. Nó gần như là không bao giờ lý tưởng cho một cơ sở dữ liệu để thực hiện việc quét của entiretable. Có hai trường hợp ngoại lệ cho quy tắc này: bruteforce truy vấn một cách rõ ràng yêu cầu, chẳng hạn như SELECT \* FROM MyTable, và các bảng nhỏ được lưu trữ trong toàn bộ của họ trong một khối trên đĩa lưu trữ duy nhất (ví dụ, thường ít hơn so với một vài trang của dữ liệu ). Tiết kiệm cho hai trường hợp ngoại lệ, một bảng quét thường là một chỉ báo rằng các thiết kế cơ sở dữ liệu vật lý có thể được cải thiện đáng kể bằng cách thêm một chỉ mục hoặc một số thuộc tính thiết kế vật lý khác.



Hình 11.9 Thực hiện kế hoạch truy vấn phạm vi có và không có MDC.

1. Sắp xếp lớn. Các loại có thể tốn kém khi họ đang lớn. Phân loại có thể phù hợp hoàn toàn trong sự cố tràn bộ nhớ vào đĩa (tạm thời), phát sinh không chỉ tài nguyên CPU đáng kể để thực hiện việc phân loại mà còn mở rộng I / O để quản lý phần đổ tạm thời của dữ liệu. Loại lớn có thể là một chỉ báo rằng một chỉ số là mất tích hoặc là một cái nhìn hiện thực hay MDC thiết kế có thể được sử dụng để loại bỏ sự cần thiết phải sắp xếp các. Nếu sắp xếp được sử dụng cho một nhóm hoạt động, nó có thể hình thành một phần tuyệt vời của một cái nhìn vật chất.
2. Nhỏ các loại. Loại nhỏ thường không phải là một vấn đề nếu họ hiếm khi xảy ra. Tuy nhiên, một loại nhỏ trên hàng ngàn các truy vấn chạy đồng thời có thể là một thảm họa. Một lần nữa, một chỉ số, MDC, hoặc xem vật chất thông thường sẽ khắc phục vấn đề.
3. Liên phân vùng thông tin liên lạc trên các hệ thống chia sẻ, không có gì. Nếu kế hoạch thực hiện truy vấn cho thấy truyền thông interpartition rất cao, nó có thể là một chỉ báo rằng các bảng tham gia vào nhiều tham gia kém collocated . Một thiết kế phân vùng mới cho các bảng (hoặc sử dụng nhân rộng quan điểm vật chất) có thể giúp giải quyết vấn đề này .
4. Rất cao thực hiện chi phí tiêu thụ tài nguyên Nếu các chi phí thực hiện của một truy vấn là rất cao, nó là có thể đi một trong các truy vấn chậm nhất và sẽ tiêu thụ nhiều tài nguyên hệ thống (có thể CPU, I / O, mạng, truy vấn thực hiện kế hoạch sẽ cho bạn biết). Điều này có thể không là một vấn đề truy vấn có thể thực sự tốn kém để thực hiện vì sự phức tạp của nó. Tuy nhiên, chi phí cao có thể dẫn bạn đến các truy vấn cần sự quan tâm của bạn.
5. Thường xuyên xảy ra các vị predicates.Commonly định kỳ có thể hình thành tiềm năng các phím bắt đầu / dừng cho một chỉ số. Hãy xem xét các cột được sử dụng thường xuyên xảy ra các vị là ứng cử viên cho các cột hàng đầu của một chỉ số mới .
6. Thiết kế các thuộc tính không được sử dụng. Nếu các thuộc tính bạn đã thêm vào thiết kế cơ sở dữ liệu của bạn (chỉ số, quan điểm vật chất, MDC, vv) không được sử dụng, bạn có thể có những giả định không chính xác về các cơ sở dữ liệu và khối lượng công việc, hoặc có thể tối ưu truy vấn đã sai lầm và lựa chọn một thực hiện truy vấn tối ưu kế hoạch mà không sử dụng thiết kế thuộc tính bạn đã được xây dựng vào hệ thống. Dù bằng cách nào, đó là một vấn đề cần chú ý. Nếu một số thuộc tính thiết kế ra không có ích, bạn nên kiểm tra chi phí bảo trì giữ chúng xung quanh và xem xét loại bỏ chúng.

Có một số lý do rộng lớn hơn lý do tại sao có thể nhìn thấy kế hoạch thực hiện truy vấn được lựa chọn bởi quá trình tối ưu truy vấn có thể rất có lợi cho thiết kế cơ sở dữ liệu:

1. Chụp kế hoạch thực hiện truy vấn cho phép người sử dụng để duy trì một lịch sử tốn kém nhất, và do đó thường khó giải quyết nhất, executionplans truy vấn qua giai đoạn chuyển tiếp.
2. Bổ sung chỉ số mới và các thuộc tính thiết kế khác, chẳng hạn như phạm vi và danh sách phân vùng, MDC, quan điểm vật chất, và sự lựa chọn phân vùng có thể được examinedfor tác động của họ vào kế hoạch thực hiện truy vấn của các truy vấn thường issuedagainst cơ sở dữ liệu.
3. Cập nhật dữ liệu lớn và / hoặc bổ sung sẽ thay đổi đáng kể số liệu thống kê của các đối tượng cơ sở dữ liệu. Các thống kê này được sử dụng bởi các trình biên dịch truy vấn trong quá trình tối ưu hóa và thay đổi cho họ thường kết quả trong kế hoạch thực hiện truy vấn mới. Nó được phổ biến rằng các vấn đề hiệu suất có thể xảy ra trong một hệ thống cơ sở dữ liệu sau đây thay đổi dữ liệu lớn như vậy vì hai lý do:

* Các dữ liệu đã thay đổi và kế hoạch thực hiện truy vấn cũ là không tối ưu cho các dữ liệu mới trong hệ thống .

• Các dữ liệu đã thay đổi và các số liệu thống kê đối tượng cho một hoặc nhiều đối tượng tham chiếu trong truy vấn đã được cập nhật, kết quả trong việc lựa chọn của một kế hoạch thực hiện truy vấn sẽ xảy ra là không tối ưu do để lựa chọn người nghèo của trình biên dịch truy vấn (tức là, trong quá trình tối ưu hóa) .

1. Trong khi nâng cấp sản phẩm, phiên bản mới nhất của bất kỳ sản phẩm cơ sở dữ liệu quan hệ có thể bao gồm thông tin cập nhật đến các mô hình toán học được sử dụng để tối ưu hóa truy vấn. Thay đổi trong mô hình có thể dẫn đến những thay đổi trong kế hoạch thực hiện truy vấn. Nếu kế hoạch mới thực hiện tồi tệ hơn những cái cũ, nó có thể rất có lợi để gỡ lỗi các nguyên nhân gốc rễ của vấn đề trước và kế hoạch thực hiện sau khi nâng cấp câu truy vấn có sẵn để so sánh và độ tương phản. Điều này đúng cho cả thiết kế cơ sở dữ liệu và khách hàng nhà cung cấp các nhóm dịch vụ họ nên cần phải được kêu gọi để được trợ giúp.
2. Cuối cùng, thiết lập cấu hình cơ sở dữ liệu được sử dụng như là một phần lựa chọn kế hoạch thực hiện truy vấn. Ví dụ, số lượng bộ nhớ có sẵn cho dữ liệu bộ nhớ đệm thường được sử dụng trong một mô hình ngẫu nhiên để xác định khả năng của một trang dữ liệu được lưu trữ tại thời điểm thực hiện truy vấn (như trái ngược với việc yêu cầu I / O từ đĩa ). Nếu kết quả cấu hình thay đổi trong việc giảm hiệu suất, tương phản kế hoạch thực hiện truy vấn trước và sau khi thay đổi cấu hình có thể xác định nếu thay đổi kế hoạch thực hiện truy vấn xảy ra mà có thể là nguyên nhân gốc rễ của sự thay đổi hiệu suất, và nếu vậy, những gì thay đổi cụ thể đã dẫn sự lựa chọn của một kế hoạch thực hiện truy vấn mới .
3. **Tìm hiểu cách tối ưu mà không làm thay đổi thiết kế**

Nhiều nhà cung cấp cho phép người dùng thực hiện những gì nếu phân tích thiết kế cơ sở dữ liệu vật lý mà không thực sự làm thay đổi toàn diện cơ sở dữ liệu của họ. Có nhiều cách này được thực hiện.

Đầu tiên, người dùng có thể trích xuất một bản sao của thiết kế cơ sở dữ liệu để tạo mới, nhưng dữ liệu, cơ sở dữ liệu. Việc thiếu các dữ liệu làm cho thiết kế khai thác dễ dàng, kể từ khi thay đổi thiết kế có thể được thực hiện mà không có quá trình tốn kém và tốn thời gian thực tế của việc tạo ra các chỉ số hoặc xem vật chất, chuyển đổi bảng MDC, phạm vi hoặc danh sách phân vùng, hoặc thay đổi sự lựa chọn phân vùng-không có gì chia sẻ. Tất cả những thay đổi này thiết kế là rất tốn thời gian khi một lượng lớn dữ liệu được tham gia, nhưng là tương đối nhanh chóng trong một môi trường dữ liệu miễn phí .

Chúng tôi đã đề cập trước đó rằng kế hoạch thực hiện truy vấn phụ thuộc vào kích thước và phân phối dữ liệu trong cơ sở dữ liệu. Vì vậy, người ta có thể tự hỏi làm thế nào một mô phỏng hợp lệ của cơ sở dữ liệu có thể được thiết lập dữ liệu miễn phí, nhưng vẫn cung cấp một mô phỏng công bằng của kế hoạch thực hiện truy vấn. Câu trả lời nằm trong thực tế là tối ưu hóa truy vấn sử dụng các số liệu thống kê được lưu trữ trong danh mục hệ thống và các thống kê xếp vào mục lục có thể được sao chép cơ sở dữ liệu dữ liệu miễn phí. Kết quả là, mặc dù cơ sở dữ liệu không có dữ liệu trong đó, trình biên dịch truy vấn (và tối ưu hóa) nhìn thấy một cơ sở dữ liệu đầy đủ các dữ liệu với số liệu thống kê liên quan. Thành phần tối ưu hóa trình biên dịch truy vấn, trong bản chất là một mô hình toán học và như vậy nó là hoàn toàn không biết cho dù dữ liệu tồn tại trong các đối tượng được tham chiếu bởi incomingquery một. Mô hình này tiêu thụ số liệu thống kê được tìm thấy trong danh mục hệ thống và assumesthey là đại diện.

Chiến lược thứ hai là để thực hiện điều gì sẽ xảy ra nếu phân tích trong cơ sở dữ liệu chứa dữ liệu. Điều này có thể được thực hiện bởi các nhà cung cấp nhiều thông qua sự hỗ trợ của các đối tượng "ảo". Các đối tượng ảo là đối tượng mà người dùng định nghĩa như là mặc dù họ đã thực sự, nhưng chỉ thị các cơ sở dữ liệu để tạo ra chúng hoàn toàn như định nghĩa cho mục đích của điều gì sẽ xảy ra nếu phân tích. Một số cơ sở dữ liệu hỗ trợ việc tạo ra các quan điểm người dùng xác định các đối tượng ảo forindexes và vật chất. Như chúng ta sẽ thấy trong Chương 12, hầu hết các nhà cung cấp sử dụng các đối tượng ảo để thực hiện hỗ trợ thiết kế tự động cho cơ sở dữ liệu, bao gồm một phạm vi rộng hơn so với chỉ số chỉ và quan điểm vật chất. Kết quả là, có khả năng rằng trong những năm tới, hỗ trợ cho người dùng định nghĩa các đối tượng ảo sẽ mở rộng, tạo cho thiết kế cơ sở dữ liệu nhiều để khám phá những lựa chọn thiết kế trong cơ sở dữ liệu chính mà họ sử dụng để lưu trữ dữ liệu.

1. **Các vấn đề bất buộc khi tối ưu hóa truy vấn**

***11.7.1 Ba chiến lược thiết yếu***

Mặc dù 30 năm nghiên cứu tiên tiến vào mô hình hệ thống và tối ưu hóa truy vấn, cuối ngày, tối ưu hóa truy vấn sử dụng mô hình toán học, và các mô hình này không hoàn hảo. Nếu không có đi sâu vào một danh sách liệt kê chi tiết về lý do tại sao nó là như vậy khó khăn để tạo ra một truy vấn tối ưu hoàn hảo, chúng tôi yêu cầu các độc giả của chúng tôi để chấp nhận điều này là thực tế và tập trung vào các câu hỏi quan trọng hơn phải làm gì về nó. Có ba kỹ thuật chính để làm việc xung quanh vấn đề của tôi ưu hoa truy vấn lựa chọn một kế hoạch thực hiện truy vấn người nghèo, có lẽ điều đó không khai thác các thuộc tính thiết kế vật lý mà bạn đã thiết kế thành cơ sở dữ liệu. Đây là những điều sau đây:

1. Thay đổi độ sâu tối ưu hóa truy vấn tìm kiếm.

2. Sửa đổi một số số liệu thống kê cho các đối tượng truy cập.

3. Kế hoạch lựa chọn lực lượng nhất định bằng cách sử dụng các "gợi ý truy vấn.

Như chúng ta đã thấy, quá trình tối ưu hóa truy vấn là rất phức tạp và có thể liên quan đến tìm kiếm rất sâu sắc và đầy đủ để tìm thấy một kế hoạch lý tưởng. Các truy vấn phức tạp hơn và các thuộc tính thiết kế nhiều hơn để lựa chọn, lựa chọn hơn và phức tạp phải đối mặt với tối ưu hóa truy vấn. Trong thực tế, thời gian cần thiết cho quá trình tối ưu hóa truy vấn có thể là đáng kể . Hầu hết các cơ sở dữ liệu cố gắng để tự động xác định độ sâu của việc tìm kiếm tối ưu hóa (hoặc sử dụng một độ sâu tìm kiếm mặc định là hợp lý cho hầu hết các truy vấn, hầu hết thời gian). Tuy nhiên, thường có ghi đè. Khi tối ưu hóa truy vấn xuất hiện là làm cho sự lựa chọn sai và lựa chọn một kế hoạch truy vấn thực hiện tối ưu, một trong những chiến lược là để thay đổi độ sâu tìm kiếm bằng cách hoặc tăng lên hoặc giảm nó, hy vọng kế hoạch adifferent và cấp trên sẽ được lựa chọn.

Một chiến lược để lựa chọn kế hoạch thiên vị là để thay đổi các số liệu thống kê tôi ưu hoa dựa vào khi lựa chọn các kế hoạch thực hiện truy vấn. Nhiều nhà cung cấp cơ sở dữ liệu cung cấp quan điểm người dùng có thể cập nhật những thống kê này . Theo mặc định, những quan điểm người sử dụng sửa đổi cho thấy các giá trị tương tự cho các số liệu thống kê đối tượng là danh mục chính lưu trữ các số liệu thống kê thực tế. Khi những quan điểm được cập nhật, các giá trị mới supercede số liệu thống kê thực tế. Bằng cách này, thiết kế cơ sở dữ liệu có thể thiên vị truy cập của một đối tượng. Ví dụ, bạn có thể làm cho một chỉ số xuất hiện nhiều hơn mong muốn bằng cách tăng số liệu thống kê phân nhóm của nó hoặc bằng cách giảm kích thước của nó (số cây hoặc số lượng trang).

Cuối cùng, cách rõ ràng nhất của xu hướng kế hoạch thực hiện truy vấn là thông qua các gợi ý truy vấn. SQL Server, Oracle, DB2 cung cấp một cơ sở cho các gợi ý truy vấn, mặc dù các giao diện khác nhau đáng kể. Gợi ý nên được sử dụng thận trọng và chỉ khi truy vấn tối ưu sẽ xuất hiện để gặp khó khăn nghiêm trọng việc tìm kiếm một kế hoạch thực hiện truy vấn hợp lý cho một truy vấn. Gợi ý là phổ biến nhất là sử dụng lực lượng truy cập chỉ mục hoặc tham gia vào một phương pháp cụ thể, hoặc cơ bản đóng băng một kế hoạch truy vấn thực hiện dễ bay hơi cho một truy vấn phức tạp, để giảm nguy cơ thời gian chạy của một sự lựa chọn kế hoạch đột ngột nghèo trên một hệ thống sản xuất.

***11.7.2 Giới thiệu gợi ý truy vấn***

Sau đây là một số hiểu biết cú pháp nhanh chóng bằng cách sử dụng các gợi ý truy vấn cho các nhà cung cấp lớn, mà không có bất kỳ nỗ lực được toàn diện. Gợi ý truy vấn SQL Server có thể được thêm trực tiếp đến các văn bản SQL bằng cách thêm các khoản OPTION . Điều này có định dạng sau :

OPTION (hint1 [, ... hintn])

Ví dụ sau đây sử dụng các điều khoản OPTION lực touse tối ưu hóa truy vấn một phương pháp băm tham gia khi tham gia hai bảng.

SELECT \*

FROM receipts.supplier AS t1

INNER JOIN receipts.address AS t2

ON t1.supplier\_id = t2.supplier\_id

WHERE Territory\_id = 12 OPTION (HASH JOIN);

Ví dụ sau đây sử dụng OPTION (tối ưu hóa cho ...) khoản để lựa chọn  
kế hoạch thực hiện truy vấn rằng sẽ là tốt nhất cho city\_name 'New York, mặc dù vị city\_name thực tế là' Toronto.

DECLARE city\_name nvarchar(50)

SET city\_name = 'Toronto'

SELECT \* FROM employee.Address

WHERE City = city\_name

OPTION (OPTIMIZE FOR (city\_name = 'New York'))

SQL Server gợi ý bao gồm lựa chọn, và một mô tả trong số này có thể được tìm thấy trực tuyến tại Microsoft SQL Server các trang web:

EXPAND VIEWS, FAST n, FORCE ORDER, { HASH | ORDER }

GROUP, KEEPFIXED PLAN, KEEP PLAN, { LOOP | MERGE | HASH }

JOIN, MAXDOP number, { MERGE | HASH | CONCAT } UNION,

OPTIMIZE FOR, ROBUST PLAN

Oracle hỗ trợ một mảng rộng các gợi ý. Những người ủng hộ đề nghị này là minh chứng cho sự phong phú của Oracle tính năng tối ưu, trong khi đối thủ đã lên tiếng rằng nó có thể gợi ý một điểm yếu trong mô hình tối ưu hóa Oracle, đòi hỏi phải sử dụng nhiều công nghệ gợi ý để tránh các kế hoạch thực hiện truy vấn xấu. Bất kể nơi bạn đứng trong cuộc tranh luận này, có rất ít nghi ngờ danh sách các gợi ý hỗ trợ bởi Oracle là rất toàn diện . Không giống như SQLServer, trong đó bao gồm các gợi ý bằng cách sử dụng một từ khóa, các gợi ý của Oracle được thêm vào như một bình luận trên các văn bản SQL. Dưới đây là một vài ví dụ. Ví dụ đầu tiên cho thấy làm thế nào gợi ý Oracle có thể được sử dụng để thực hiện chính xác như nhau, như trong ví dụ SQL Server trên: lực lượng ahash tham gia bằng cách sử dụng các gợi ý USE\_HASH.

SELECT \*

/\*+ USE\_HASH (supplier, address) \*/

FROM receipts.supplier AS t1

INNER JOIN receipts.address AS t2

ON t1.supplier\_id = t2.supplier\_id

WHERE Territory\_id = 12;

Ví dụ sau đây cho thấy làm thế nào một gợi ý Oracle được sử dụng để ép buộc việc sử dụng một chỉ mục bằng cách sử dụng các gợi ý INDEX:

SELECT \*

/\*+ INDEX (employees employee\_dept\_inx)\*/

employee\_id, department\_id

FROM employees

WHERE department\_id < 35;

Danh sách sau đây cho thấy phạm vi cũng hỗ trợ Oracle gợi ý và gợi ý cung cấp tài liệu.

ALL\_ROWS, AND\_EQUAL, ANTIJOIN, APPEND, BITMAP, BUFFER,

CACHE, CARDINALITY, CHOOSE, CLUSTER FULL, CPU\_COSTING,

DRIVING\_SITE, FACT, FIRST\_ROWS, HASH, ROWID, HASH\_AJ,

HASH\_SJ, INDEX, INDEX\_ASC, INDEX\_COMBINE, INDEX\_DESC,

INDEX\_FFS, INDEX\_JOIN, INLINE, LEADING,

MATERIALIZE, MERGE, MERGE\_AJ, MERGE\_SJ, NL\_AJ, NL\_SJ,

NO\_ACCESS, NO\_BUFFER, NO\_EXPAND,

NO\_EXPAND\_GSET\_TO\_UNION, NO\_FACT, NO\_INDEX, NO\_MERGE,

NO\_MONITORING, NO\_PUSH\_PRED, NO\_PUSH\_SUBQ, NO\_QKN\_BUFF,

NO\_SEMIJOIN, NOAPPEND, NOCACHE, NOPARALLEL,

NOPARALLEL\_INDEX, NOREWRITE, OR\_EXPAND, ORDERED,

ORDERED\_PREDICATES, PARALLEL, PARALLEL\_INDEX,

PQ\_DISTRIBUTE, PUSH\_PRED, PUSH\_SUBQ, REWRITE ,RULE,

SELECTIVITY, SEMIJOIN, SEMIJOIN\_DRIVER, STAR,

STAR\_TRANSFORMATION, SWAP\_JOIN\_INPUTS, USE\_ANTI,

USE\_CONCAT, USE\_HASH, USE\_MERGE, USE\_NL, USE\_SEMI.

Undocumented hints:

BYPASS\_RECURSIVE\_CHECK, BYPASS\_UJVC, CACHE\_CB,

CACHE\_TEMP\_TABLE, CIV\_GB, COLLECTIONS\_GET\_REFS, CUBE\_GB,

CURSOR\_SHARING\_EXACT, DEREF\_NO\_REWRITE, DML\_UPDATE,

DOMAIN\_INDEX\_NO\_SORT, DOMAIN\_INDEX\_SORT,

DYNAMIC\_SAMPLING, DYNAMIC\_SAMPLING\_EST\_CDN,

EXPAND\_GSET\_TO\_UNION, FORCE\_SAMPLE\_BLOCK,

GBY\_CONC\_ROLLUP, GLOBAL\_TABLE\_HINTS,

HWM\_BROKERED, IGNORE\_ON\_CLAUSE, IGNORE\_WHERE\_CLAUSE,

INDEX\_RRS, INDEX\_SS, INDEX\_SS\_ASC, INDEX\_SS\_DESC,

LIKE\_EXPAND, LOCAL\_INDEXES, MV\_MERGE,

NESTED\_TABLE\_GET\_REFS, NESTED\_TABLE\_SET\_REFS,

NESTED\_TABLE\_SET\_SETID, NO\_FILTERING, NO\_ORDER\_ROLLUPS,

NO\_PRUNE\_GSETS, NO\_STATS\_GSETS, NO\_UNNEST,

NOCPU\_COSTING, OVERFLOW\_NOMOVE, PIV\_GB, PIV\_SSF, PQ\_MAP,

PQ\_NOMAP, REMOTE\_MAPPED, RESTORE\_AS\_INTERVALS,

SAVE\_AS\_INTERVALS, SCN\_ASCENDING, SKIP\_EXT\_OPTIMIZER,

SQLLDR, SYS\_DL\_CURSOR, SYS\_PARALLEL\_TXN, SYS\_RID\_ORDER,

TIV\_GB, TIV\_SSF, UNNEST, USE\_TTT\_FOR\_GSETS.

**11.7.3 Những gợi ý Truy vấn SQL chưa được thực hiện để sửa đổi**

Trong đa số trường hợp các thiết kế cơ sở dữ liệu không phải là cùng một người là nhà thiết kế ứng dụng và họ thường không có quyền truy cập vào mã ứng dụng để thêm các gợi ý. Bởi vì điều này, một chiến lược mạnh mẽ để gợi ý là một trong những cho phép quản trị cơ sở dữ liệu để thêm các gợi ý mà không sửa đổi các văn bản SQL của truy vấn đến. Các chiến lược cơ bản mà các nhà cung cấp cơ sở dữ liệu sử dụng để thực hiện việc này là để cho phép một quản trị viên để xác định một khuôn mẫu cho các truy vấn với một gợi ý liên quan. Khi aquery xâm nhập vào hệ thống, nó là một thời gian ngắn kiểm tra để xem nếu nó phù hợp với một trong những mẫu này và, nếu có một trận đấu, các quy tắc cho mẫu đó được áp dụng cho các truy vấn đến. Oracle gọi điều này là Phác thảo lưu trữ và DB2 gọi điều này là Profiles Tối ưu hóa. Lưu trữ Phác thảo và Profiles Tối ưu hóa cung cấp thông qua một bảng tất cả các ứng dụng đóng gói sẵn để được "gợi ý" mà không cần mở chúng để chỉnh sửa các câu lệnh SQL. Một bất lợi là phương châm bên ngoài phải được kết hợp với câu lệnh SQL khá chặt chẽ . Các sản phẩm như DB2 và Oracle cố gắng bình thường hóa các báo cáo càng nhiều càng tốt bằng cách ép không gian màu trắng và uppercasing, nhưng nếu văn bản tuyên bố được sắp xếp lại ở tất cả, nó sẽ thường xuyên phù hợp với không còn.

**Những mẹo tìm hiểu và sự hiểu biết về cơ sở dữ liệu chuyên dụng**

* Mẹo 1. Bảng quét và loại lớn xuất hiện trong kế hoạch thực hiện truy vấn là chuông báo động. Phải ngay lập tức thu hút sự chú ý của bạn và thăm dò kịp thời cho các tính năng thiết kế để thêm hoặc loại bỏ .
* Mẹo 2. Xác định phần lớn thời gian thực hiện truy vấn được chi . Sử dụng kế hoạch thực hiện truy vấn rất nhiều trong quá trình thiết kế cơ sở dữ liệu để phát hiện các điểm tiêu thụ tài nguyên lớn. Những điểm này trong kế hoạch thực hiện truy vấn là những khu vực nơi bạn nên focusy nỗ lực thiết kế của chúng tôi để cải thiện hiệu suất truy vấn và hiệu quả hệ thống do đó.
* Mẹo 3. Đảm bảo tính năng thiết kế của bạn đang được sử dụng. Sử dụng phân tích truy vấn kế hoạch thực hiện cho thấy nếu chỉ số, MDC, quan điểm vật chất, và phân vùng phạm vi thực sự là một cách thích hợp được sử dụng bởi các trình biên dịch truy vấn theo cách mà bạn mong đợi.
* Lời khuyên 4. Chỉ số cardinality. Sử dụng phân tích truy vấn kế hoạch thực hiện để xác định tính chọn lọc của các vị áp dụng trên các truy cập bảng cơ sở, điều này có thể là một chỉ báo tốt cho dù một chỉ số có thể giúp giảm thời gian truy cập bảng. Nếu chọn lọc rất cao (tức là, không nhiều hàng có được lọc ra), hơn 40% hoặc hơn, nó là không một chỉ số sẽ giúp giảm I / O, trừ khi chỉ số được thiết kế để bao gồm tất cả các cột cần thiết bởi nhà điều hành trên bàn.
* Mẹo 5. Cheat nếu cần thiết . Bạn có thể buộc các vấn đề bằng cách thay đổi độ sâu tìm kiếm của tối ưu hóa, sửa chữa số liệu thống kê, hoặc bằng cách thêm gợi ý . Bạn nên xem xét những nỗ lực cuối mương để. Trước khi dùng đến gợi ý, bạn có thể để có được một kế hoạch truy vấn thực hiện tốt hơn mà không cần buộc các trình biên dịch xuống một con đường truy cập cụ thể chỉ đơn giản bằng cách thay đổi gây hấn hoặc xem các dữ liệu bằng cách điều chỉnh các số liệu thống kê trên các đối tượng cơ sở dữ liệu. Tại sao không chỉ nhảy thẳng đến gợi ý? Tối ưu hóa có thể vẫn còn có thể tìm thấy một kế hoạch tốt hơn nhiều so với một trong những bạn sẽ có hiệu lực với các gợi ý truy vấn.
* Mẹo 6. Hãy nhớ rằng các chi phí hoạt động viết . Thêm tính năng thiết kế giống như quan điểm vật chất và chỉ số đáng kể có thể cải thiện hiệu suất của các truy vấn, nhưng họ có thể có một tác động bất lợi về chèn, cập nhật, và xóa (IUD) hoạt động . Mỗi viết hoạt động sẽ yêu cầu công việc được thực hiện để giữ cho các đối tượng này . Trong trường hợp các quan điểm vật chất, hầu hết các nhà cung cấp hỗ trợ khả năng cập nhật các quan điểm vật chất chỉ định kỳ (ví dụ, hàng tuần, hàng tháng, vv), mà có thể làm giảm các chi phí vật chất quan điểm gây ra cho các dụng cụ tử cung, nhưng là một kế hoạch kết quả thực hiện truy vấn là truy cập những quan điểm vật chất sẽ trả lại dữ liệu cũ !
* Mẹo 7. Chia sẻ không có gì cụ thể. Nếu máy chủ cơ sở dữ liệu của bạn sử dụng một kiến trúc không có gì chia sẻ (xem Chương 6), một mối quan tâm chung để xem ra cho là tài nguyên thông tin liên lạc dự kiến trong kế hoạch thực hiện truy vấn. Các chi phí truyền thông cao có thể là một chỉ báo dữ liệu kém collocated gây ra không hiệu quả tham gia và khối lượng vận chuyển dữ liệu cao. Nếu bạn có một hệ thống đang chạy, có thường theo dõi dữ liệu có sẵn mà có thể chỉ cho bạn (không chỉ tối ưu hóa dự kiến) tỷ giá thực tế, bạn có thể tương phản với ước tính của tôi ưu hoa để kiểm tra nếu có thể là một vấn đề với kế hoạch .

1. **Tóm tắt**

Khả năng để xem kế hoạch thực hiện truy vấn và một số tài nguyên liên quan và chọn lọc cho các nhà khai thác trong kế hoạch là một công cụ quan trọng trong quá trình thiết kế cơ sở dữ liệu vật lý. Vì vậy, nhiều như vậy mà không có cơ sở dữ liệu hiện đại là có hiệu quả được thiết kế ngày nay mà không khám phá các đường dẫn truy cập. Phân tích đường dẫn truy cập có thể giúp các nhà thiết kế cơ sở dữ liệu nhìn thấy nơi mà các điểm tài nguyên nóng, và xem các mẫu thiết kế do họ đã tạo ra trong cơ sở dữ liệu đang được khai thác. Khi thiết kế cơ sở dữ liệu có vẻ tốt, nhưng tôi ưu hoa truy vấn không phải là lựa chọn các tính năng thiết kế có sẵn, một số điều chỉnh có thể là cần thiết bởi người quản trị để thiết lập những điều. Các khả năng bao gồm: điều chỉnh độ sâu tìm kiếm tối ưu hóa, sửa chữa các số liệu thống kê cho các đối tượng được sử dụng trong các truy vấn vi phạm, hoặc thêm gợi ý truy vấn (hoặc cấu hình) để ép buộc những lựa chọn thực hiện kế hoạch truy vấn cho các truy vấn cụ thể.