THIẾT KẾ VÀ ĐIỀU CHỈNH DỮ LIỆU MỨC VẬT LÝ

Lời khuyên cho người khách hàng mà phàn nàn về lỗ dột trên mái nhà dột ướt bàn ăn đó là: hãy chuyển bàn đi nơi khác”

Kiến trúc sư Frank Lloyd Wright

Câu lệnh của một DBMS trên các truy vấn thường gặp và các hoạt động cập nhật điển hình là thước đo cuối cùng của một thiết kế cơ sở dữ liệu. 1 DBA có thể cải tiến câu lệnh bằng cách điều chỉnh 1 vài thông số trong DBMS(ví dụ: kích thược của vùng nhớ đệm hoặc tầng số của thực thi của checkpoint) hay là tìm ra những giới hạn về khả năng đáp ứng của phẩn cứng(bottlenecks) sau đó ta sẽ thêm những phần cứng để đáp ứng cho vận hành hệ thống.tuy nhiên bước trên hết, để có thể thực hiện câu lệnh tốt thì thiết kế dữ liệu cần phải tốt.

sau khi thiết kế lược đồ ở mức quan niệm(conceptual and external) là tạo ra những quan hệ và những khung nhìn cùng với những ràng buộc toàn vẹn, ta phải xác định mục đích câu lệnh trong thiết kế mức vật lý(thiết kế lược đồ vật lý). Khi người dùng yêu cầu tăng cao, thiết kế lược đồ mức vật lý rất cần thiết để hoà hợp, điều chỉnh, tất cả các khía cạnh của 1 thiết kế cơ sở dữ liệu cho câu lệnh thực thi tốt.

mục đích của chương này sẽ được tóm gọn như sau, chúng tôi đưa ra 1 cái nhìn chung về thiết kế và tinh chỉnh ở mức vậy lý trong phần 16.1. điều quan trọng nhất của thiết kế mức vật lý là giải quyết vấn đề liên quan đến index. Chúng tôi đưa ra những nguyên tắc cho việc quyết định lựa chọn tạo indexs trong phần 16.2. những nguyên tắc này được minh hoạ trong 1 số ví dụ và những vấn đề chuyên sâu hơn trong phần 16.3 đến 16.6. trong phần 16.3 chúng tôi đưa ra những ví dụ làm nổi bật những cách lựa chọn index cơ bản. trong phần 16.4 chúng ta xem kỹ hơn những vấn đề gom nhóm (clustering); ta bàn luận làm thế nào để chọn index gom nhóm và có lưu các bộ từ các quan hệ khác nhau gần với chúng không. Trong phần 16.5 ta bàn về việc dùng index với hỗn hợp hoặc multiple-attribute search keys . trong phần 16.6 ta nhấn mạnh làm cách nào lựa chọn tốt index có thể cho phép nhiều câu queries được trả lởi mà không cần xem những dữ liệu có thật( actual data records).

Trong phần 16.7 ta xem tổng quát vấn đề chính của tinh chỉnh dữ liệu. hơn nữa việc tinh chỉnh index, ta có thể phải tinh chỉnh lược đồ mức quan niệm, cũng như thường xuyên dùng câu query và những định nghĩa khung nhìn( view definitions) . ta bàn làm thế nào để tinh lọc lại lược đồ mức quan niệm trong phần 16.8 và làm thế nào để tinh chỉnh lại câu queries và khung khùng nhìn trong phần 16.9. ta bàn luận ngắn gọn sự ảnh hưởng lớn về các câu lệnh thực thi xảy ra đồng thời trong 16.10. và kết thúc chương với bàn luận ngắn về những tiêu chuẩn(benchmarks) về DBMS trong 16.11; tiêu chuẩn lựa chọn DBMS để giúp thực thi tốt các câu lệnh.

***Công cụ thiết kế vật lý****: cho đến nay RDBMSs (relational Database Management System) đã cung cấp vài công cụ để hỗ trợ việc thiết kế và tinh chỉnh lược đồ mức vật lý. MS SQL server có công cụ tinh chỉnh mà hỗ trợ về tạo index; nó cũng hỗ trợ xoá bỏ 1 index. IBM DB2 V6 cũng có công cụ tinh chỉnh và Oracle Expert hỗ trợ về thêm, xoá index.*

**16. 1 giới thiệu về thiết kế mức vật lý.**

Giống như các khía cạnh của các mức thiết kế dữ liệu khác, thiết kế vật lý phải phụ thuộc phù hợp với những dữ liệu có ngoài đợi thực và những dụng ý sử dụng nó. Một cách riêng biệt, nó rất quan trọng để biết được những công việc điển hình cần phải làm(typical workload) mà cơ sở dữ liệu phải cung cấp cho người dùng. Wordload bao gồm các câu query hoặc update dữ liệu . điều tất nhiên là người dùng sẽ yêu cầu về tốc độ của câu query, udate phải nhanh bao nhiêu, hay là có bao nhiêu transaction phải thực thi mỗi giây. Miêu tả của workload và yêu cầu về thực thi của người dùng là nhưng cái yêu cầu cơ bản mà mà cần phải giải quyết trong thiết kế mức vật lý.

Để tạo 1 dữ liệu mức vật lý tốt và để tinh chỉnh hệ thống cho câu lệnh trong việc đáp trả lại yêu cầu của người dùng, người thiết kế cần phải hiểu công việc của 1 DBMS, đặc biệt là về chỉ mục và cách thực thi câu query của DBMS đó. Nếu dữ liệu mong đợi được truy cập đồng thời bởi nhiểu ngưởi dùng hoặc là 1 cơ sở dữ liệu phân tán(distributed database), nhiệm vụ trở nên phức tạp rất nhiều và các tính năng khác của DBMS được dùng đến. ta bàn luận ảnh hưởng lớn về truy cập đồng thơi trên dữ liệu được thiết kế trong 16.10.

**16.1.1 database workloads**

1 workload description bao gồm các thành phần sau:

* 1 danh sách các query và tần số sử dụng chúng, như 1 phần nhỏ của tất cả query và update.
* 1 danh sách các câu lệnh update và tần số sử dụng nó.
* Mục đích câu lệnh cho mỗi loại query và update.

Mỗi câu query trong workload, ta phải xác định:

* Những quan hệ nào được dùng đến.
* Những thuộc tính nào được xuất ra (trong lệnh SELECT).
* Những thuộc tính nào có điều kiện lựa chọn hoặc là có điều kiện kết trong lệnh WHERE và chọn lựa những điều kiện này như thế nào.

Giống như trên, mỗi lệnh update trong workload, ta phải xác định

* Những thuộc tính nào có điều kiện lựa chọn hoặc là điều kiện kết (trong lệnh where) và chọn lựa những điều kiện này như thế nào.
* Loại của câu lệnh update náy (insert , delete, hay update) và quan hệ cần update.
* Các trường mà bị điểu chỉnh lại trong lệnh update này.

Nên nhớ rằng các câu lệnh query và update thường có các đối số, ví dụ thao tác về sổ ghi nợ hay tín dụng thì sẽ gọi đến mã số tài khoản. giá trị của những đối số này xác định độ chọn lọc của điều kiện lựa chọn và điều kiện kết.

Lệnh update có 1 thành phần query mà được dùng đề tìm các bộ cần update. Thành phần này có thể giúp ích cho thiết kế mức vật lý tốt và sự có mặt của index. Mặt khác, lệnh update thường yêu cầu thêm việc đề duy trì index trên các thuộc tính mà chúng cần chỉnh. Vì vậy, trong khi các lệnh query chỉ đươc lợi từ sự có mặt của 1 index, thì 1 index có thể hoặc tăng tốc độ truy vấn hoặc giảm tốc độ cho lệnh update. Người thiết kế cần lưu ý vấn đề này khi tạo index.

**16.1.2 giải quyết việc thiết kế và tinh chỉnh**

Những giải quyết quan trong suốt quá trình thiết kế và tinh chỉnh dữ liệu bao gồm các bước sau:

1. Những index nào được tạo.
   * Những quan hệ nào được chỉ mục và những thuộc tính nào hoặc tập hợp thuộc tính nào được chọn làm khoá index để tìm.
   * Mỗi index, nó nên được gom nhóm hay không gom nhóm(unclustered), nó có nên được nhiều hay thưa thớt
2. Ta có nên thay đổi lược đồ mức quan niềm để cải thiện truy vấn không. Ví dụ, ta phải xét điều sau:
   * Thay đổi lược đồ bình thường hoá: ta luôn có nhiều cách để phân tích 1 lược đồ thành dạng chuẩn mong muốn(BCNF hoặc 3NF). Lựa chọn có thể được dựa vào tiêu chuẩn thực thi.
   * Sự không bình thường hoá (denormalization): ta có thể muốn xem xét lại phân rã lược đồ đã lấy đi sự bình thường hoá trong suốt quá trình thiết kế lược đồ mức quan niệm để cải tiến câu lệnh query mà gọi các thuộc tính từ các quan hệ bị phân rã trước đây.
   * Sự phân chia thẳng đứng: ở những tình huống dĩ nhiên ta có thể muốn phân rã các quan hệ thành nhiều quan hệ nhỏ hơn nữa để tăng tốc độ thực thi của query do chỉ thực hiện trên ít thuộc tính.
   * Khung nhìn: ta có thể tạo nhiều khung nhìn để che đi những thay đổi trong lược đồ quan hệ không cho người dùng biết.
3. Có nên thường xuyên thực thi query và transaction được ghi lại nhiều lần để tôc độ nhanh hơn không.

Trong dữ liệu song song hoặc phân tán có nhiều lựa chọn còn xét đến ví dụ như có nên chia quan hệ qua các vị trí khac nhua không hoặc có nên lưu bản sao chép quan hệ ở nhiều địa điểm khác nhau không.

**16.1.3 sự cần thiết của tinh chỉnh dữ liệu**

Một cach chính xác, thông tin chi tiết của workload có thể khó để nhận biết được trong khi bắt đầu thiết kế của hệ thống. do đó, điều chỉnh dữ liệu sau khi nó thiết kế và triển khai thì rất quan trọng- ta phải tinh chỉnh lại thiết kế ban đầu trong đúng với cách diễn tả của đời thực để thu được những câu lệnh thực thi tốt nhất có thể.

Sự khác biệt giữa thiết kết dữ liệu và tinh chỉnh dự liệu là 1 vài thứ không bị bó buộc. ta có thể xét quy trình thiết kế hơn 1 lần lược đồ ban đầu ở mức khái niệm được thiết kế và tập hợp các chỉ mục và giải quyết vấn đề gom nhóm được thực hiện. bất cứ sự thay đổi xảy ra sau khi đã tạo lược đồ mức quan niệm hoặc chỉ mục rồi thì ta có thể gọi đó là việc tinh chỉnh. Như một lựa chọn, ta có thể xét 1 vài cách tinh chỉnh lược đồ mức quan niệm là 1 phần của quá trình thiết kế vật lý.

Ranh giới giữa việc thiết kế và tinh chỉnh không cần thiết phải rõ ràng ( nó không quan trọng), và ta sẽ bàn luận 1 cách đơn giản vấn để lựa chọn chỉ mục và tinh chỉnh dữ liệu mà không cần quan tâm đến khi nào việc tinh chỉnh được thực hiện.

**16.2 hướng dẫn cho lựa chọn index**

Để quyết định chỉ mục nào được dùng thì trước tiên ta sẽ làm việc với danh sách query mà ban đầu ta đã tìm. Một cách rõ ràng, chỉ những quan hệ mà tham gia vào câu query thì mới được xét đến để xác định làm chỉ mục, và việc lựa chọn thuộc tính để tạo chỉ mục trên được hướng dẫn bởi các lệnh điều kiện xuất hiện trong lệnh WHERE của câu query, 1 index mà thích hợp với lược đồ thì sẽ cải tiến rất lớn cho tốc độ truy vấn.

1 cách lựa chọn chỉ mục là phải xét những câu query quan trọng, và mỗi câu query đó phải xác định chỉ mục thích hợp mà nằm trong danh sách chỉ mục hiện có.sau đó ta xét có có nên đi đến 1 kế hoạch tốt hơn với những chỉ mục khác không; nếu những chỉ mục thêm vào này mà là ứng cử bao hàm trong danh sách chỉ mục của ta. nhìn chung, phạm vi thu hồi sẽ được giúp bởi chỉ mục B+, và sự thu hổi chính xác sẽ được giúp bởi 1 chỉ mục kém. Việc gom nhóm sẽ giúp ích cho dãy câu query, và nó sẽ giúp các query chính xác nếu 1 vài thực thể có chứa cùng khoá.

Tuy nhiên Trước khi thêm 1 chỉ mục vào danh sách, t phải xét xem chỉ mục này có ảnh hưởng đến lệnh update trong danh sách ta liệt kê trong workload không. Nếu ta chú ý sớm, mặc dù 1 chỉ mục có thể tăng tốc thành phần truy vấn trong lệnh update nhưng tất cả chỉ mục trên 1 thuộc tính được update – trên tất cả thuộc tính, trong trường hợp insert, deletes phải được update bất cứ khi nào giá trị thuộc tính thay đổi. vì thế, thình thoảng ta phải xét sự hại của việc làm chậm lệnh update và có lợi cho truy vấn.

Rõ ràng hơn, việc chọn 1 bộ chỉ mục thích hợp đáp ứng yêu cầu trong workload yêu cầu phải hiểu kỹ thuật index có sẵn, và hiểu hoat động query. Những nguyên tắc dưới đây cho phần lựa chọn chỉ mục được tóm gọn như sau:

Guideline 1 (whether to index): quan điểm rõ ràng thường là điều quan trọng nhất. không xây dựng chỉ mục trừ khi nhiều câu truy vấn - bao gồm cả những thành phần query trong lệnh update- sẽ có ích cho chúng. Bất cứ khi nào có thể, chọn những chỉ mục mà làm tốc độ cho nhiều câu query.

Guideline 2 ( choice of search key): những thuộc tính được đề cập trong lệnh WHERE là những ứng cử viên lựa chọn làm chỉ mục.

* + 1 điều kiện lựa chọn đề nghị rằng ta nên xét 1 chỉ mục trên những thuộc tính đã được chọn.
  + 1 dãy điều kiện lựa chọn đề nghị rằng ta nên xét đến chỉ mục B+ tree (hoặc ISAM) trên những thuộc tính được lựa chọn. 1 chỉ mục B+ tree luôn thích hợp hơn 1 chỉ mục ISAM. 1 chỉ mục ISAM có thể có giá trị nếu quan hệ này thường xuyên phải update, nhưng ta sẽ thừa nhận rằng 1 chỉ mục B+ tree luôn luôn là lựa chọn tốt hơn chỉ mục ISAM.

Guideline 3 (multiple-attribute search keys): những chỉ mục với khoá tìm kiếm đa thuộc tính nên được xét đến tron 2 tình huống sau:

* + Lệnh WHERE chứa các điều kiện có hơn 1 thuộc tính trong quan hệ.
  + Chúng chỉ cho phép chiến lược định giá chỉ mục ( i.e.., việc truy cập quan hệ có thể được tránh) cho những câu query quan trọng. (tình huống này có thể dẫn đến những thuộc tính đang là khoá tìm kiếm thậm chí chúng không xuất hiện trong lệnh WHERE).

K khi tạo những chỉ mục trên các khoá tìm kiếm với nhiều thuộc tính, nếu dãy query được mmong đợi , hãy cẩn thận với những thuộc tính trong khoá tìm kiếm để thực hiện query.

Guideline 4 (whether to cluster): ở hầu hết 1 chỉ mục trong 1 quan hệ có thể được gom nhóm, và việc gom nhóm làm ảnh hưởng lớn đến lệnh thực thi; vì thế lựa chon gom nhóm chỉ mục rất quan trọng.

* + Như 1 luật thống trị, dãy query thì giống như làm lợi cho đa số việc gom nhóm. Nếu 1 vài dãy query được đặt ra trên quan hệ, việc bao gồm các tập hợp thuộc tính khác nhau, xét đến tính chọn lọc của những câu query và tần số liên kết của chúng trong workload khi quyết định chọn chỉ mục nào gom nhóm.
  + Nế u 1 chỉ mục cho phép chiến lược ước lượng chỉ 1 chỉ mục cho câu query thì được có dụng ý để tăng tốc độ, chỉ mục không cần gom nhóm. (chỉ mục chỉ có ý nghĩa quan trọng khi chỉ mục đó được dùng để lấy ra những bộ từ quan hê).

Guideline 5 (hast versus tree index): 1 chỉ mục B+ tree thì luôn được chuộng hơn bởi vì nó hỗ trợ cho dạng truy vấn range query(truy vấn có liên quan đến phạm vi giá trị thuộc tính) cũng như các truy vấn bằng(equality query). 1 hash index (A  Hash index stores key value pairs based on a pseudo randomizing function called hash function) thì tốt hơn trong trường hợp sau:

* + Chỉ mục được thiết kế để hỗ trợ chỉ mục mà được lồng vào các phép kết lặp; quan hệ có chỉ mục là quan hệ nội bộ, và khoá tìm kiếm chứa các cột mà thực hiện phép kết. trong trường hợp này, hash index sẽ làm tăng tốc độ thực thi hơn B+ tree cho câu lệnh select bằng(equality selections) , bởi vì 1 lệnh select bằng (equality selection) được sinh ra cho mỗi bộ trong quan hệ ngoài vùng.
  + Có câu lệnh bằng(equality query) rất quan trọng, và câu lệnh đó không c1o truy vấn dạng range queries, bao hàm cả những thuộc tính khoá tìm kiếm.

Guideline 6 (balancing the cost of index maintenance): sau khi tạo được những index như mong đợi thì hãy xét các ảnh hưởng của mỗi index trong những lệnh update nằm trong workload như thế nào.

* + Nếu việc giữ lại các chỉ mục mà gây ra thực thi chậm cho các lệnh update thì nên gõ bỏ nó đi.
  + Tuy nhiên, luôn nhớ rằng việc thêm 1 chỉ mục cũng có thể làm tăng tốc các lệnh udate. Ví du : 1 chỉ mục trên thuộc tính employee ids có thể làm tăng tốc cho thao tác lệnh: tăng lương cho các employee

**16.3 1 số ví dụ cơ bản cho việc lựa chọn chỉ mục**

Ví dụ dưới đây minh hoạ làm thế nào lựa chọn các chỉ mục trong quá trình thiết kế dữ liệu. lược đồ được dùng trong ví dụ này thì không diễn tả chi tiết; thông thường chúng bao gồm các thuộc tính named trong lệnh query. Thông tin thêm được lấy ra khi cần thiết.

Ta hãy bắt đầu với lệnh query đơn giản sau:

SELECT E .ename, D.mgr

FROM Employees E, Departments D

WHERE D.dname = ‘Toy’ AND D.dno = D.dno

Những quan hệ được đề cập đến trong câu query này là Employees và Departments, và 2 điều kiện trong WHREW đều là loại điều kiện bằng. theo những nguyên tắc đã nói thì ta nên xây dựng những hash index trên thuộc tính dname của Departments. Nhưng xét đẳng thức E.dno = D.dno, ta có nên xay 1 chỉ mục (dĩ nhiên là hash index) trên 2 thuộc tính của Departments hoặc Employees( hay cả 2) không? Theo trực giác thì ta nên lấy những bộ của Departments sử dụng chỉ mục trên dname vì có ít bộ thoả mãn đẳng thức D.dname = ‘Toy’. Vì cứ tướng ứng với những bộ mà thoả mãn đẳng thức trên thì ta phải dùng đến index trên thuộc tính dno trong Employees tương ứng với những bộ trong Departments đó. Vì thế, ta nên xây dựng 1 chỉ mục trên trường dno của Employees. Lưu ý là tốc độ ttăng thêm bởi việc xây dựng index trên trường dno của Departments bởi vì các bộ của Departments mà được lấy ra thì sử dụng chỉ mục dname)

Lựa chọn những chỉ mục của chúng ta được chỉ dẫn bởi 1 query evaluation plan mà ta cần dùng. Sự cân nhắc của plan evaluation tiềm năng này là chung chung trong khi quyết định thiết kế dữ liệu. việc hiểu sự tối ưu hoá câu query là rất quan trọng cho thiết kế vật lý. Ta thể hiện dự định mong muốn cho query này trung hình 16.1.

Giả sử câu query trên có 1 chút thay đổi ở WHERE như sau: D.dname = ‘Toy AND E.dno = D.dno And E.age = 25. Chúng ta hãy xét lựa chọn những dự định ước lượng. 1 dự định tốt là để lấy ra những bộ Departments mà thoả với đẳng thức dname và lấy ra những hộ Employees bằng cách dùng chỉ mục trên trường dno; việc chọn age được ứng dụng on-the-fly. Tuy nhiên, không giống như query trước, ta không cần có 1 chỉ mục trên trường dno của Employees nếu ta có 1 chỉ mục trên age. Trường hợp này ta có thể lấy lấy các bộ Departments mà thoả mãn đẳng thức trên dname(bằng cách dùng index trên dname, như trước đó). Lấy các bộ Employees mà thoả mãn các lựa chọn trên age mà dùng index trên age, và kết những bộ này lại. vì tập các bộ mà ta kết rất nhỏ nên chúng hợp với bộ nhớ và phép kết không quan trọng. dự định này giống như là thứ gì đó kém hơn so với dùng 1 chỉ mục trên dno, nhưng nó là reasonable alternative. Vì thế, nếu ta có 1 chỉ mục trên age rồi thì sự khác biệt của mẫu query này không khẳng định việc tạo chỉ mục trên trường dno của Employees.

Sau đây là ví dụ query loại range selection:

SELECT E.ename, D.dname

FROM Employees E, Departments D

WHERE E.sal BETWEEN 10000 AND 20000

AND E.hobby = ‘Stamps’ AND E.dno = D.dno

Câu truy vấn này minh hoạ việc dùng BETWEEN cho diễn tả phạm vi lựa chọn. nó tương đương với điều kiện sau:

10000 <= E.sal AND E.sal <= 20000

Quay lại ví dụ về câu truy vấn , cả 2 selecttion đều trên quan hệ Employees. Vì thế, rõ ràng 1 dự định mà Employees là quan hệ ngoại và Departiments là quan hệ nội là tốt nhát, vì trong truy vấn trước, ta nên xây dựng 1 hash index trên thuộc tính dno của Departments. Nhưng chỉ mục nào nên được dùng trên Employees? 1 chỉ mục B+ trên thuộc tính sal sẽ giúp lựa chọn thuộc range selection, đặc biệt nếu nó là gom nhóm. 1 hash index trên thuộc tính hobby sẽ giúp lựa chọn loại equality selection. Nếu 1 trong số các chỉ mục có sẵn, ta có thể lấy các bộ của Employees dùng chỉ mục này, từ đó lấy các bộ Departments bằng cách dùng chỉ mục trên dno, và ứng dụng tất cả các lựa chọn còn lại và và các dự án on-the-fly. Nếu cả 2 chỉ mục đã sẵn sàng, người dùng thông minh sẽ chọn nhiều selective truy cập quỹ đạo để gửi các query; đó là, nó sẽ xét cái selection nào có ít hơn các bộ thoả mãn. Nhìn chung, cái nào truy cập quỹ đạo là có lựa chọn nhiều hơn phụ thuộc vào dữ liệu. nếu có rất ít người có salaries nằm trong phạm vi lựa chọn và nhiều người thu nhặt tem, thì dùng B+ là tốt nhất. cách khác, 1 hash index trên hobby là tốt nhất.

Nếu câu truy vấn ràng buộc được biết(như ví dụ của chúng ta), những cái chọn lọc có thể là ước lượng nếu thoả nếu đã thống kê dữ liệu. mặt khác, 1 equality selection giống như nhiều selective, và 1 reasonable decision có thể là tạo 1 hash index trên hobby. Thỉnh thoảng, các câu truy vấn ràng buộc thì không được biết – ta phải kiếm 1 truy vấn bằng cách mở rộng 1 câu truy vấn bằng khung nhìn, hoặc ta phải có 1 truy vấn động SQL, cái mà cho phép ràng buộc được định rõ như trong thời gian chạy. trong trường hợp này, nếu truy vấn rất quan trọng, ta có thể chọn tạo 1 chỉ mục B+ trên sal và 1 hash index trên hobby và bỏ lựa chọn được làm bởi optimizer run-time.

**16.4 gom nhóm và chỉ mục**

Range queri là ứng cử tốt cho sự cải tiến với chỉ mục gom nhóm( clustered index):

SELECT E.dno

FROM Employees E

WHERE E.age >40

Nếu ta có 1 chỉ mục B+ trên age , ta có thể dùng nó để lấy các bộ mà thoả lệnh E.age > 40. Như 1 chỉ mục là quan trọng phụ thuộc lệnh điều kiện lựa chọn mà đứng đâu tiên hay không, phần nào của employees thì già hơn 10? Nếu trên thực tế tất cả mọi người đều lớn hơn 40 tuổi thì ta không phải nỗ lực nhiều trong việc tạo chỉ mục trong age; việc duyệt quan hệ hầu hết có thể tốt. tuy nhiên giả sử chỉ có 10% employees có tuổi lơn hơn 40 thì giờ chỉ mục có ích hay không? Câu trả lời phụ thuộc vào chỉ mục có được gom nhóm hay không, nếu chỉ mục không được gom nhóm thì ta có thể có 1 trang I/O cho mỗi employee thoả điều kiện, và điều này có thể rất tốn hơn là duyệt theo thứ tự thậm chí chỉ có 10% employees thoả điều kiện! mặt khác, 1 chỉ mục B+ gom nhóm trên age yêu cầu chỉ 10% I/)s cho 1 lần duyệt thứ tự.

Như 1 ví dụ khác, xét câu lệnh đươc sửa lại như sau:

SELECT E.dno, COUNT(\*)

FROM Employees E

WHERE E.age > 10

GROUP BY E.dno

Nếu 1 chỉ mục B+ có sẵn trên age, ta có thể lấy các bộ bằng cách dùng nó, sắp xếp các bộ nhận được theo dno, và sau đó trả lời câu truy vấn. tuy nhiên điều này có thể là 1 dự kiến không tốt nếu thực tế tất cả employees đều lớn hơn 10 tuổi, dự kiến này sẽ không hay nếu chỉ mục không được gom nhóm.

Ta hảy xét xem 1 chỉ mục trên dno có thể tiện lợi cho giả thiết hơn không, ta có thể dùng chỉ mục để nhận lấy tất cả các bộ, nhóm dno, và mỗi dno ta đếm số lượng bộ mà có age >10. Nhắc lại, hiệu quả phụ thuộc chủ yếu vào việc có nên dùng chỉ mục được gom nhóm hay không. Dự định này sẽ là tốt nhất nếu như không có điều kiện về tuổi. nếu 1 chỉ mục không có gom nhóm, ta có thể thực thi 1 trang I/O mỗi bộ trong Employees, và dự định này thật là kinh khủng, thực tế, nếu 1 chỉ mục không được gom nhóm, người dùng thông minh sẽ chọn dự định không phức tạp dựa vào sắp xếp theo dno. Vì thế, câu truy vấn này để nghị ta nên xây dựng 1 chỉ mục gom nhóm trên dno nếu điều kiện về age rất không có lựa chọn.nếu điều kiện rất có lựa chọn , ta nên tạo 1 chỉ mục trên age.

Việc gom nhóm cũng quan trọng cho 1 chỉ mục trên khoá tìm kiếm mà không chứa 1 khoá ứng củ, đó là 1 chỉ mục mà 1 vài thực thể có thể có cùng giá trị khoá trên nó. Để làm rõ vấn đề này, ta có truy vấn sau:

SELECT E.dno

FROM Employees E

WHERE E.hobby = ‘Stamps’

Nếu nhiều người sưu tầm tem, thì việc lấy các bộ trong chỉ mục trên hobby mà không có gom nhóm có thể rất không hiệu quả. Nó có thể đơn giản hơn duyệt quan hệ để lấy tất cả các bộ và để ứng dụng vào lựa chọn nhanh với các bộ nhận được. vì thế, nếu 1 truy vấn mà quan trọng thì ta nên xét đến tạo ra 1 chỉ mục trên hobby 1 chỉ mục gom nhóm. Mặt khác, nếu ta nhận rằng eid là 1 khoá của Employees, và thay thế điều kiện E.hobby = ‘Stamps’ bằng E.eid = 552, ta biết rằng hầu hết 1 bộ của 1 Employees sẽ đáp ứng điều kiện select. Trong trường hợp này, không có lợi gì khi làm chỉ mục gom nhóm.

Chỉ mục gom nhóm có thể đặc biệt quan trọng trong khi truy cập quan hệ nội bộ trong 1 chỉ mục được lồng vào kết lặp. để hiểu quan hệ giữa gom nhóm chỉ mục và phép kết, ta xem ví dụ sau:

SELECT E.ename, D.mgr

FROM Employees E, Departments D

WHERE D.dname = ‘Toy’ AND E.dno = D.dno

Ta kết luận rằng 1 dự định ước lượng tốt là dùng chỉ mục trên dname để lấy các bộ Departments thoả điều kiện trên dname và từ đó tìm bộ Employees bằng cách dùng chỉ mục trên dno. Ta có nên cho các chỉ mục này gom nhom không? Cho giả định của ta rằng 1 số lượng bộ thoả D.dname = ‘Toy’ dường như ít, ta nên xây dựng 1 chỉ mục không gom nhóm trên dname. Mặt khác, Employees là 1 quan hệ nội bộ trong 1 chỉ mục được lồng vào phép kết lặp, và dno thì không là khoá . tình huống này là 1 lý lẽ mạnh cho biết dùng chỉ mục gom nhóm trên dno.

Xét ví dụ sau:

SELECT E.ename, D.mgr

FROM Employees E, Departments D

WHERE E.hobby = ‘Stamps’ AND E.dno = D.dno

Câu truy vấn này khắc truy vấn trước ở điều keịn6 E.hobby = ‘Stamps’ thay thế bằng D.dname = ‘Toy’. Dựa vào giả thiết rằng có ít employees trong Toy department, ta chọn chỉ mục mà dễ thuận tiện cho lặp được lồng chỉ mục kết với join Department như quan hệ ngoại. giờ ta giả thiết rằng nhiều employees sưu tầm tem. Trong trường hợp này, 1 block nested loops hay soet-merge kết có thể gây nhiều rắc rối. 1 kết sort-merge có thể lấy lợi ích của 1 chỉ mục B+ trên dno của Departments để lấy các bộ và tránh việc sắp xếp Departments. Chú ý là 1 chỉ mục không gom nhóm thì không có ích – vì tất cả bộ được nhận, việc thực thi 1 I/O mỗi bộ dường như chi phí cao. Nếu không có chỉ mục trên dno của Employees, ta có thể nhận bộ Employees, ứng dụng Selection E.hobby = ‘Stamps’ nhanh, và xếp các bộ thoả theo dno.

Hình 16.2

**16.4.1 gom nhóm 2 quan hệ.**

Summarize co-clustering:

* Nó có thể làm tăng tốc độ phép kết, đặc biệt khoá ngoại kết tương ướng với quan hệ 1:N.
* Duyệt tuần tự của quan hệ sẽ làm chậm hơn.
* Lệnh insert, delete, update thay đổi độ dài dữ liệu sẽ trở nên chậm đi, nhờ các chi phí liên quan đến việc duy trì các gom nhóm.

**16.5 chỉ mục trên khoá tìm kiếm mà nhiều thuộc tính**

Thỉnh thoảng sẽ rất tốt nếu xây dựng 1 chỉ mục trên khoá tim kiếm mà chứa nhiều thuộc tính.ví dụ, nếu ta muốn lấy các dữ liệu trong Employees với điều kiện age = 30 và sal = 4000, thì 1 chỉ mục với khoá tìm kiếm là (age, sal) (hoặc sal, age) thì cao cấp hơn chỉ mục với khoá tìm kiếm là age hoăc là sal. Nếu 1a có 2 chỉ mục, 1 là trên age và 1 là tên sal, ta có thể dùng chúng để trả lời câu truy vấn bằng cách lấy và giao nhau. Tuy nhiên nếu ta xét đến những chỉ mục nào mà được tạo ra vì mục địch như câu truy vấn này thì dùng chỉ mục ghép lại là tốt hơn.

Khi gặp các vấn đề như có tạo chỉ mục gom nhóm hay không gom nhóm, hay thưa thớt hay không thì cũng liên quan đến lựa chon khoá tìm kiếm. ta sẽ gọi những chỉ mục có khoá tìm kiếm nhiều thuộc tính là chỉ mục hỗn hợp(composite indexes). Composite index có thể dùng đề phục vụ cho truy vấn phạm vi nhiều chiều.

Ví dụ:

SELECT E.eid

FROM Employees E

WHERE E.age BETWEEN 20 AND 30 AND E.sal BETWEEN 3000 AND 5000

Composite index trên (age, sal) giúp khi có yêu cầu trong lệnh WHERE có sự lựa chọn 1 cách rõ ràng. Một cách rõ ràng, 1 hash index không thể giúp , 1 B+ index được yêu cầu làm, khi ta thấy 1 chỉ mục gom nhóm sẽ giúp cải tiến hơn chỉ mục không gôm nhóm. Trong ví dụ này ta thấy với chỉ mục B+ có khoá tìm kiếm là (age, sal) thì cũng như chỉ mục B+ có khoá là (sal, age). Tuy nhiên thứ tự các thuộc tính trong khoá tìm kiếm đôi khi cũng có nhiều sai khác lớn, như query dưới đây:

SELECT E.eid

FROM Employees E

WHERE E.age = 25

AND E.sal BETWEEN 3000 AND 5000

Đây là câu lệnh có chỉ mục composite index với chỉ mục B+ gom nhóm, trên (age, sal), điều này cho thực thi của câu lệnh tốt vì dữ liệu được sắp xếp bởi age và sau đó theo sal. Như vậy, tất cả các dử liệu nào có age = 25 thì cùng gom 1 nhóm. Mặt khác, 1 chỉ mục hỗn hợp B+ có gom nhóm khoá trên(sal, age) sẽ cho thực thi không tốt. vì dữ liệu hàng sẽ được xếp theo sal trước và khi đó 2 hàng có cùng giá trị age có thể nằm ở xa nhau. Kết quả, chỉ mục này cho phép ta dùng lựa chọn phạm vi cho sal nhưng không có lựa chon cho age để lấy các bộ ra.

Mặt trái của composite index: phải được update đáp lại cho bất kỳ thao tác nào mà làm chỉnh sửa(insert, delete, update) đến các thuộc tình mà tham gia làm khoá tìm kiếm. 1 composite index dường như lớn hơn chỉ mục có khoá là 1 thuộc tính vì kích thước của các entry lớn hơn.

**16.6 chỉ mục cho phép những dự án chỉ có chỉ mục(indexes that enable index-only plans)**

Mục này xét về 1 lượng lớn truy vấn mà ta có thể tìm những dự định mà mang lại hiệu quả để tránh lấy ra các bộ tự 1 trong những quan hệ mà phải liên kết đến(referenced) cho chúng. 1 chỉ mục chỉ được dùng để duyệt mà không phải gom nhóm vì những bộ mà từ quan hệ được chỉ mục thì không được lấy ra! Tuy nhiên, chỉ có chỉ mục chằn chịt(dense) mới có thể được dùng cho index-only.

Ví dụ:

SELECT D.mgr

FROM Departments D, Employees E

WHERE D.dno = E.dno

Quan sát thấy không có thuộc tính nào của Employees được lấy ra. Nếu ta có dense index trên trường dno của Employees, sự tối ưu hoá của việc tạo 1 chỉ mục được lồng vào kết lặp mà sử dụng 1 index-only để duyệt quan hệ nội bộ thì thích hợp.

**16.7 xem qua tinh chỉnh dữ liệu (overview of database tuning)**

sau khi thiết kế dữ liệu mức vật lý thì ta có bước tình chỉnh dữ liệu, vì trong quá trình thiết kế sẽ khó tránh khỏi nhưng sai xót, do đó bước tình chỉnh sẽ là bước hoàn thiện dữ liệu mức vật lý. Tinh chỉnh dữ liệu giúp cho khả năng tăng tốc thực thi cao hơn. Trong phần này ta bàn về tinh chỉnh chỉ mục, tinh chỉnh lược đồ quan hệ, tinh chỉnh câu truy vấn.

**16.7.1 tuning indexs**

Việc lựa chọn chỉ mục ban đầu cũng có thể được tinh chỉnh lại do 1 số lý do. Lý do đơn giản nhất là dựa vào khối lượng công việc quan sát cho thấy rằng 1 số các truy vấn và cập nhậ được coi là quan trọng trong đặc tả khối lượng công việc ban đầu không phải là rất thường xuyên. Và khối lượng công việc quan sát cũng cho ra những truy vấn quan trọng nhưng trong đặc tả của ta ban đâu không có. Vì thế 1 số chỉ mục sẽ cần được tạo ra hoặc bị gỡ bỏ. lúc này ta sẽ thực hiện tinh chỉnh chỉ mục.

Lý do: có 1 số chỉ mục ta áp dụng không đúng, khi đó cần lựa chọn lại loại chỉ mục nào chình xác để tối ưu truy vấn: index-only, index clustered, index unclustered…

Chú ý: tối ưu truy vấn dựa vào sự thống kê đã duy trì trong hệ thống catalog. Những thống kế này được update khi 1 chương trình đặc biệt được chạy; chắc rằng để chạy….

16.7.2 tuning the conceptual schema

Như ta biết, trong thiết kế lược đồ, lược đồ ở mức quan niềm mà ta lựa chọn chưa hẳn là lược đồ hợp lý so với thực tế (workload) yêu cầu mặc dù ở mức thiết kế vật lý làm hết sực khả năng có thể. Vì vậy ta có thể phải điều chỉnh lại lược đồ mức quan niệm của ta cho phù hợp với workload. Một vài điều phải xét đến khi tinh chỉnh lược đồ quan hệ;

* Ta cần quyết định chọn dạng chuẩn thiết kề nào: 3NF hay BCNF.
* Nếu có 2 cách để phân rã cho ra lược đồ từ dạng chuẩn 3NF hay BCNF , thì ta cần dựa vào workload để lựa chọn lược đồ nào.
* Thỉnh thoảng ta nên quyết định phân rã quan hệ nữa không mặc dù nó đã đạt BCNF.
* Trong 1 số tình huống ta có thể không theo tiêu chuẩn hoá. Đó là, ta cần phải lựa chọn để loại bỏ 1 quan hệ mà có được do phân rã từ quan hệ lớn hơn