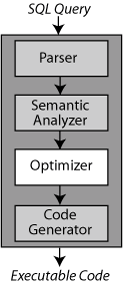
TimesTen và TimesTen Cache có một trình tối ưu hóa câu truy vấn dựa trên chi phí đảm bảo truy cập dữ liệu hiệu quả bằng cách tự động tìm kiếm cách tốt nhất để trả lời các truy vấn. Tối ưu hóa được thực hiện trong giai đoạn thứ ba của quá trình biên dịch. Các giai đoạn biên dịch được thể hiện trong hình sau:



Vai trò của trình tối ưu hóa là xác định gói chi phí thấp nhất để thực hiện câu truy vấn. Bằng "kế hoạch chi phí thấp nhất", có nghĩa là quá trình truy cập dữ liệu mất ít thời gian nhất. TimesTen sẽ gọi trình tối ưu hóa cho các câu lệnh SQL khi có nhiều hơn một kế hoạch thực thi. Trình tối ưu hóa chọn những gì nó nghĩ là kế hoạch tối ưu. Kế hoạch này vẫn tồn tại cho đến khi câu lệnh bị vô hiệu hoặc bị hủy thông qua ứng dụng.

Trình tối ưu hóa xác định chi phí của kế hoạch dựa trên:

* Thống kê bảng và cột
* Thông tin siêu dữ liệu (chẳng hạn như tính toàn vẹn tham chiếu, khóa chính)
* Lựa chọn chỉ mục (bao gồm tạo tự động temporary indexes)
* Phương pháp quét (quét toàn bộ bảng, tra cứu hàng loạt, quét range index, tra cứu bitmap index, tra cứu hash index)
* Việc lựa chọn giải thuật join (nested loop joins, nested loop joins with indexes, or merge join)

Thời gian tối ưu hóa và sử dụng bộ nhớ

Trình tối ưu hóa được thiết kế để tạo ra kế hoạch tốt nhất có thể trong thời gian hợp lý và hạn chế về bộ nhớ. Không phải trình tối ưu hóa luôn chọn được kế hoạch tối ưu nhất cho mọi truy vấn. Thay vào đó, mục tiêu của trình tối ưu hóa là chọn ra kế hoạch tốt nhất từ ​​một tập hợp các kế hoạch được tạo ra bằng cách sử dụng các chiến lược tìm ra các vùng có tiềm năng trong không gian tìm kiếm. Bởi vì tối ưu hóa thường chỉ xảy ra một lần cho mỗi truy vấn trong khi chính truy vấn có thể được thực hiện nhiều lần, nên trình tối ưu hóa ưu tiên thời gian thực thi hơn thời gian tối ưu hóa.

Các kế hoạch được tạo bởi trình tối ưu hóa nhấn mạnh hiệu suất hơn việc sử dụng bộ nhớ. Trình tối ưu hóa có thể chỉ định việc sử dụng một lượng đáng kể của không gian bộ nhớ tạm thời để tăng tốc thực thi. Trong môi trường hạn chế bộ nhớ, ứng dụng có thể sử dụng những gợi ý của trình tối ưu hóa để vô hiệu hóa việc sử dụng index và bảng tạm để tạo ra kế hoạch mà đánh đổi hiệu suất tối đa nhằm giảm mức sử dụng bộ nhớ.

Số liệu thống kê

Khi xác định đường đi thực thi cho câu truy vấn, trình tối ưu hóa kiểm tra thống kê về dữ liệu được tham chiếu bởi truy vấn, chẳng hạn như số hàng trong bảng, những giá trị cực đại, cực tiểu và số lượng giá trị unique trong cột, sự tồn tại của các khóa chính trong một bảng, kích thước và cấu hình của chỉ mục đang dùng. Các thống kê này được lưu trữ trong các bảng SYS.TBL\_STATS và SYS.COL\_STATS, được dùng khi một ứng dụng gọi lệnh ttIsql statsupdate, hoặc gọi những thủ tục có sẵn như , ttOptUpdateStats, ttOptEstimateStats

Trình tối ưu hóa sử dụng thống kê cho mỗi bảng để tính toán tính *selectivity* của các vị từ, chẳng hạn như t1.a=4 hoặc kết hợp các vị từ, chẳng hạn như t1.a=4 AND t1.b<10. *Selectivity* là một sự định giá số hàng trong bảng. Nếu một vị từ chọn một tỷ lệ phần trăm nhỏ các hàng, thì đó là *high* selectivity, trong khi một vị từ chọn một tỷ lệ lớn các hàng thì đó gọi là *low* selectivity.

Những gợi ý của trình tối ưu hóa

Bạn có thể áp dụng các *gợi ý* để chuyển các chỉ dẫn tới trình tối ưu hóa câu truy vấn TimesTen. Trình tối ưu hóa xem xét các gợi ý này khi chọn kế hoạch thực thi tốt nhất cho câu truy vấn. Có thể áp dụng các gợi ý như sau:

* Để áp dụng gợi ý chỉ cho một câu lệnh SQL cụ thể, hãy sử dụng gợi ý trình tối ưu hóa cấp câu lệnh.
* Để áp dụng gợi ý cho toàn bộ giao tác (transaction), hãy sử dụng gợi ý trình tối ưu hóa cấp giao tác với thủ tục tích hợp TimesTen thích hợp.

## Index (chỉ mục)

Trình tối ưu hóa câu truy vấn sử dụng index để tăng tốc độ thực hiện truy vấn. Trình tối ưu hóa sử dụng index hiện có hoặc tạo ra các index tạm để sinh ra một kế hoạch thực thi. Sử dụng chiến lược index là điều cần thiết để đạt được hiệu suất tối đa từ một hệ thống TimesTen.

TimesTen sử dụng các loại index sau:

* *Hash index*: hữu ích cho việc tìm các hàng với giá trị chính xác trên một hoặc nhiều cột. Chúng có thể được chỉ định là unique hoặc không phải là unique. Nói chung, hash index nhanh hơn range index trong việc tìm kiếm giá trị chính xác và equi join. Tuy nhiên, hash index không thể được sử dụng để tìm kiếm liên quan đến giá trị theo khoảng hoặc tiền tố của khóa và có thể yêu cầu nhiều không gian hơn range index và bitmap index.
* *Range index*: hữu ích cho việc tìm các hàng với giá trị cột trong phạm vi được chỉ định là bình đẳng hoặc bất bình đẳng. Range index có thể được tạo trên một hoặc nhiều cột của bảng. Chúng có thể được chỉ định là unique hoặc không phải là unique. Khi bạn tạo một index bằng cách sử dụng câu lệnh CREATE INDEX và không chỉ định loại index thì TimesTen tạo ra một range index.
* *Bitmap index*: mã hóa thông tin của một giá trị unique trong một hàng của một bitmap. Mỗi bit trong bitmap tương ứng với một hàng trong bảng. Sử dụng bitmap index cho các cột không có nhiều giá trị unique, ví dụ như cột giới tính. Các bitmap index được sử dụng rộng rãi trong các môi trường data warehousing. Chỉ mục bitmap được nén và có yêu cầu lưu trữ nhỏ hơn so với các kỹ thuật chỉ mục khác.

Các Công cụ Index Advisor có thể được sử dụng để giới thiệu một tập hợp các chỉ mục có thể cải thiện hiệu suất của một khối lượng công việc SQL cụ thể. Để biết thêm chi tiết, hãy xem ["Sử dụng trình tư vấn chỉ mục để đề xuất các chỉ mục"](https://docs.oracle.com/cd/E11882_01/timesten.112/e21633/comp.htm#TTOPR740) trong *Hướng dẫn hoạt động cơ sở dữ liệu trong bộ nhớ Oracle TimesTen* .

## Phương pháp scan

Trình tối ưu hóa có thể chọn từ nhiều loại phương pháp scan. Các phương pháp scan phổ biến nhất là:

* Scan toàn bộ bảng
* Tìm kiếm theo rowid
* Range index scan (trên index vĩnh viễn hoặc tạm thời)
* Tìm kiếm theo hash index (trên index vĩnh viễn hoặc tạm thời)
* Tìm kiếm theo bitmap index (trên index vĩnh viễn)

TimesTen và TimesTen Cache thực hiện các tìm kiếm giá trị chính xác nhanh thông qua các hash index, bitmap index và rowid lookup. Chúng thực hiện việc tìm kiếm giá trị theo khoảng thông qua các range index. Các gợi ý của trình tối ưu hóa có thể được sử dụng để cho phép hoặc không cho phép trình tối ưu hóa xem xét một số phương pháp scan nhất định khi chọn một kế hoạch truy vấn.

Việc *scan toàn bộ bảng* là kiểm tra mỗi hàng trong một bảng. Bởi vì nó là cách hiệu quả nhất để tìm một vị từ truy vấn, việc quét toàn bộ chỉ được sử dụng khi không còn phương pháp nào khác.

TimesTen gán một ID duy nhất, được gọi là rowid, cho mỗi hàng được lưu trữ trong một bảng. Việc *tìm kiếm theo rowid* được áp dụng nếu, ví dụ, một ứng dụng trước đó đã chọn một rowid và sau đó sử dụng một mệnh đề ”WHERE ROWID= “ để tìm nạp cùng một hàng. Tìm kiếm theo rowid nhanh hơn so với tìm kiếm theo index.

*Range index scan* sử dụng range index để truy cập vào một bảng. Việc scan như vậy được áp dụng cho các vị từ tìm kiếm giá trị chính xác như “t1.a=2” hoặc cho các vị từ tìm kiếm theo khoảng như “t1.a>2” và “t1.a<10”, miễn là cột được sử dụng trong vị từ đó đánh range index. Nếu range index được dùng trên nhiều cột, nó có thể được sử dụng cho nhiều vị từ cột. Ví dụ, các vị từ “t1.b=100” và “t1.c > 'ABC' ” sẽ dùng range index scan nếu cột t1.b và cột t1.c đánh range index.

*Tìm kiếm theo hash* *index* sử dụng hash index để tìm hàng với giá trị chính xác trên một hoặc nhiều cột. Tìm kiếm như vậy được áp dụng cho các tìm kiếm bình đẳng trên một hoặc nhiều cột được chỉ định.

*Tìm* *kiếm theo bitmap index* sử dụng bitmap index để tìm các hàng thỏa mãn một vị từ bình đẳng như customer.gender='male'. Bitmap index thích hợp cho các cột có vài giá trị duy nhất. Chúng đặc biệt hữu ích trong việc tìm một vị từ mà nó có thể sử dùng tìm kiếm theo bitmap index vì các vị từ kết hợp có thể được tìm một cáh hiệu quả thông qua các phép toán bit trên các index. Ví dụ, nếu bảng Customer có một bitmap index trên cột “gender” và nếu bảng Sweater có bitmap index trên cột :color”, thì vị từ “customer.gender='male'và sweater.color ='pink' “ có thể nhanh chóng tìm thấy tất cả khách hàng nam đã mua áo len màu hồng bằng cách thực hiện phép toán logic AND trên hai bitmap index đó.

## Những phương pháp join

The optimizer can select from multiple join methods. When the rows from two tables are joined, one table is designated the *outer table* and the other the *inner table*. The optimizer decides which of the tables should be the outer table and which should be the inner table. During a join, the optimizer scans the rows in the outer and inner tables to locate the rows that match the join condition.

The optimizer analyzes the statistics for each table and, for example, might identify the smallest table or the table with the best selectivity for the query as outer table. If indexes exist for one or more of the tables to be joined, the optimizer takes them into account when selecting the outer and inner tables.

If more than two tables are to be joined, the optimizer analyzes the various combinations of joins on table pairs to determine which pair to join first, which table to join with the result of the join, and so on for the optimum sequence of joins.

The cost of a join is largely influenced by the method in which the inner and outer tables are accessed to locate the rows that match the join condition. The optimizer selects from two join methods when determining the query optimizer plan

Trình tối ưu hóa có thể chọn từ nhiều phương pháp join. Khi các hàng từ hai bảng được join, một bảng được chỉ định *outer table* và *bảng* kia là *inner table*. Trình tối ưu hóa sẽ phải quyết định bảng nào sẽ là outer và bảng nào sẽ là inner. Trong quá trình join, trình tối ưu hóa sẽ scan các hàng trong bảng outer và inner để xác định các hàng khớp với điều kiện join.

Trình tối ưu hóa phân tích thống kê cho mỗi bảng, ví dụ như có thể xác định bảng nhỏ nhất hoặc bảng có tính chọn lọc tốt nhất cho việc truy vấn để làm bảng outer. Nếu index tồn tại trong một hoặc nhiều bảng được join, trình tối ưu hóa sẽ tính đến chúng khi chọn các bảng outer và inner.

Nếu có nhiều hơn hai bảng được join, trình tối ưu hóa phân tích các kết hợp khác nhau của phép join trên các cặp bảng để xác định cặp nào nên được join trước, bảng nào sẽ kết hợp với kết quả của phép join,…

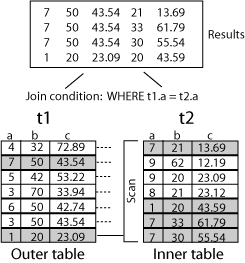
Chi phí của việc join phần lớn bị ảnh hưởng bởi phương pháp mà các bảng inner và outer được truy cập để xác định vị trí các hàng khớp với điều kiện kết nối. Trình tối ưu hóa chọn từ hai phương pháp join sau khi xác định kế hoạch tối ưu hóa câu truy vấn:

* [Nested loop join](https://docs.oracle.com/cd/E11882_01/timesten.112/e21631/query.htm#BAABBBCF)
* [Merge join](https://docs.oracle.com/cd/E11882_01/timesten.112/e21631/query.htm#BAAGAGEG)

### **Nested loop join**

Trong nested loop join không có index, một hàng trong bảng outer được chọn một tại một thời điểm và khớp với mỗi hàng trong bảng inner. Tất cả các hàng trong bảng inner được scan nhiều lần bằng số hàng trong bảng outer. Nếu bảng inner có đánh index trên cột join, index đó được sử dụng để chọn ra các hàng đáp ứng điều kiện join. Các hàng từ mỗi bảng thỏa mãn điều kiện join được trả về. Index có thể được tạo ra ngay lập tức cho các bảng inner trong các vòng lặp lồng nhau và kết quả từ các lần inner scan có thể được thực hiện trước khi join.

Xét ví dụ về nested loop join. Điều kiện join là: t1.a = t2.a



Đối với ví dụ này, trình tối ưu hóa đã quyết định t1 bảng outer và t2 là bảng inner. Các giá trị trong cột a của bảng t1 khớp với các giá trị trong cột a của bảng t2 là 1 và 7. Kết quả join nối lại các hàng từ t1 và t2.

### **Merge join**

A merge join is used only when the join columns are sorted by range indexes. In a merge join, a cursor advances through each index one row at a time. Because the rows are already sorted on the join columns in each index, a simple formula is applied to efficiently advance the cursors through each row in a single scan. The formula looks something like:

* If Inner.JoinColumn < Outer.JoinColumn, then advance inner cursor
* If Inner.JoinColumn = Outer.JoinColumn, then read match
* If Inner.JoinColumn > Outer.JoinColumn, then advance outer cursor

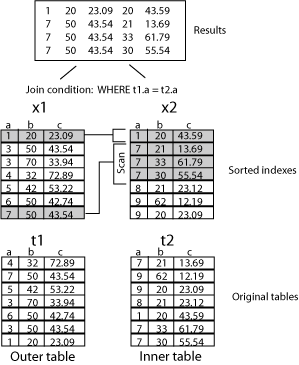
Unlike a nested loop join, there is no need to scan the entire inner table for each row in the outer table. A merge join can be used when range indexes have been created for the tables before preparing the query. If no range indexes exist for the tables being joined before preparing the query, the optimizer may in some situations create temporary range indexes in order to use a merge join.

Merge join chỉ được sử dụng khi các cột join được sắp xếp theo range index. Trong merge join, một con trỏ tiến lên qua từng một hàng index tại một thời điểm. Bởi vì các hàng đã được sắp xếp trên các cột join trong mỗi index, nên một công thức đơn giản được áp dụng để nâng cao hiệu quả các con trỏ thông qua mỗi hàng trong một lần scan. Công thức trông giống như sau:

* Nếu Inner.JoinColumn < Outer.JoinColumn, thì con trỏ inner tiến lên vị trí tiếp theo.
* Nếu Inner.JoinColumn = Outer.JoinColumn, thì đọc hàng trùng khớp đó
* Nếu Inner.JoinColumn> Outer.JoinColumn, thì con trỏ outer tiến lên vị trí tiếp theo.

Không giống như nested loop join, không cần phải scan toàn bộ bảng inner cho mỗi hàng trong bảng outer. Merge join được sử dụng khi rang index được tạo cho các bảng trước khi chuẩn bị truy vấn. Nếu không có range index tồn tại cho các bảng được join trước khi chuẩn bị truy vấn, trong một vài trường hợp, trình tối ưu hóa có thể tạo các range index tạm để sử dụng merge join.

Xét ví dụ về merge join. Điều kiện join là: t1.a = t2.a



x1 là index cho bảng t1, sắp xếp trên cột a. x2 là inex cho bảng t2, sắp xếp trên cột a. Kết quả phép merge join là nối các hàng x1 với các hàng x2 trong đó các giá trị trong cột a khớp nhau.

## Optimizer plan

Like most database optimizers, the query optimizer stores the details on how to most efficiently perform SQL operations in a query execution plan, which can be examined and customized by application developers and administrators.

The execution plan data is stored in the TimesTen SYS.PLAN table and includes information about which tables are to be accessed and in what order, which tables are to be joined, and which indexes are to be used. You can use either the ttSqlCmdQueryPlan built-in procedure or the ttIsql explaincommand to display the query plans for recently executed SQL statements. Users can direct the query optimizer to enable or disable the creation of an execution plan in the SYS.PLAN table with the generate plan optimizer hint. (For transaction level hints, use the GenPlan optimizer flag in the ttOptSetFlag built-in procedure; for statement level hints, use the TT\_GenPlan hint in the SQL statement.)

The execution plan designates a separate step for each database operation to be performed to execute the query. The steps in the plan are organized into levels that designate which steps must be completed to generate the results required by the step or steps at the next level.

Giống như hầu hết các trình tối ưu hóa cơ sở dữ liệu, trình tối ưu hóa câu truy vấn lưu trữ các chi tiết về cách thực hiện hiệu quả nhất các tác vụ SQL trong một kế hoạch thực thi câu truy vấn.

Dữ liệu kế hoạch thực hiện được lưu trữ trong bảng TimesTen SYS.PLAN và bao gồm thông tin về các bảng nào sẽ được truy cập và theo thứ tự nào, các bảng nào sẽ được join và các index nào sẽ được sử dụng. Bạn có thể sử dụng thủ tục có sẵn ttSqlCmdQueryPlan hoặc lệnh ttIsql explain để hiển thị các kế hoạch truy vấn cho các câu lệnh SQL được thực hiện gần đây. Người dùng có thể điều khiển trình tối ưu hóa câu truy vấn để để cho phép hoặc vô hiệu hóa việc tạo kế hoạch thực thi trong bảng SYS.PLAN do trình tối ưu hóa gợi ý.

Kế hoạch thực thi chỉ định một bước riêng biệt cho mỗi hoạt động cơ sở dữ liệu được thực hiện để thực hiện truy vấn. Các bước trong kế hoạch được tổ chức thành các cấp để chỉ định các bước nào phải được hoàn thành để tạo ra các kết quả theo yêu cầu của bước đó hoặc của các bước ở cấp độ tiếp theo.

Xét đoạn query sau:

SELECT COUNT(\*)

FROM t1, t2, t3

WHERE t3.b/t1.b > 1

AND t2.b <> 0

AND t1.a = -t2.a

AND t2.a = t3.a;

Trong ví dụ này, trình tối ưu hóa chia nhỏ câu truy vấn thành các phép toán riêng lẻ của nó và tạo ra một kế hoạch thực thi gồm 5 bước được thực hiện ở ba mức, như hình sau:

