実行計画のためのコスト計算式一



Reference A Look Under the Hood of CBO: The 10053 Event

説明対象のデータベース管理システムは Oracle です

Table full scan

cost NDB/k

Index fast full scan

COST

COST

NLB / k

Index unique scan

COST BLV + 2

Index range scan

NDB 表のデータ・ブロック数

NLB 索引のリーフ・ブロック数

k ベクトル I/O 係数

BLV 索引の木の深さ

B-tree level

FF データ選択率

filtering factor

CL clustering factor

連続ブロックは、メモリ上の非連続な複数 バッファへ、readv などのベクトル I/O を 行うシステムコールで読み出します。 関係データベース管理システム

RDBMS は、問い合わせを実行するとき、計算式の同値変形、内部演算の導入、アルゴリズムの選択などの最適化を行い、その結果を実行計画として提示します。この資料は、そのときの性能予測の評価材料であるコストの計算方法を、初学者向けに説明しています。

NL: Nested loop

COST

NLは、ふた

つの表を結合

する基本アル

す。外側のル

を読み、内側

のループで表

B を読み、A

と B を結合

します。

ゴリズムで

ープで表 A

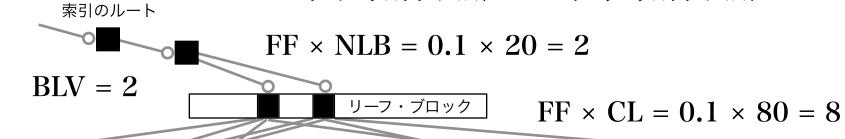
 $RA + (CA \times RB)$

ません。O (n²) は、データ量が増えるとあからさまに遅くなります。
よく使う ^{定数} O (c)

対数計算量 O (log n) は著しく速いため、線形対数計算量

O (n log n) は、線形計算量 O (n) の性能と、あまり変わり

 $BLV + (FF \times NLB) + (FF \times CL)$ $V = (FF \times CL)$



データ・ブロック

指定された条件で、データ全体の 10% を、索引経由で読むとします。つまり、FF=0.1です。 索引の深さ BLV=2 とし、リーフが 20 ブロック、CL=80 とします。そうすると、索引のリーフは合計 2 ブロック、表のデータは延べ 8 ブロック読むと推定されます。索引順に読むと、データ・ブロックを行ったり来たりして読むことになるので、索引の 100% を順番に読んだとき、同一ブロックも重複して数えた延べブロック数を、CL として事前に計測しておきます。

RA 駆動表 A を読み出すコスト

CA 駆動表 A の検索結果の行数

RB 被駆動表 B を読み出すコスト

外側の表は、内側のループを駆動するので駆動表 driving table とよばれます。駆動表から CA = 50 行が検索されると、内側のループは、50 回実行されます。 $O(\log n)$ で検索可能な索引を使って内側の表を検索すれば、NL 全体は、50 にほぼ比例の $O(n \log n)$ で計算されます。索引を使えなければ、計算量は $O(n^2)$ になり、行数の増加とともに性能があからさまに悪化します。

約 n 行のふたつの 表のすべての行の組 み合わせは約 n² 行 あります。

HA: Hash

HAは、索引の代わりに、表Bの行を指定条件で高速に検索するためのハッシュ表を検索時に構築して、表Aの各行に対し、そのハッシュ表を検索することで、AとBを結合します。

SM: Sort-merge

COST

RA + SA + RB + SB

COST

RB + HB + RA

RB 構築表 B を読み出すコスト

HB 構築表 B からハッシュ表を作るコスト

RA 探索表 A を読み出すコスト

表 B は構築表 build table とよばれ、表 A は探索表 probe table とよばれます。A と B の小さい方を構築表に選ぶと HB が下がり、その結果、全体のコストも下がります。NL とコスト比較した場合、NL の駆動表の行数が増えると、HA が有利になります。というのは、CA が 1 万行だと、RB が 2 でも、それだけで 2 万のコストが生じるからです。しかし、実際には、RB はキャッシュが効くことが多いため、ずっと低いコストしかかからず、そのときは、不適切な実行計画が選択される可能性が高くなります。

RA 表 A を読み出すコスト

SA 表 A を整列するコスト

RB 表 B を読み出すコスト

SB 表 B を整列するコスト

SM は、整列されたふたつの表 A と表 B を交互に確認しながら、小さい方から大きい方に向かって、結合可能な行を順番に見つけ出すアルゴリズムです。

このアルゴリズムは、表 A と表 B がすでに整 列済みであれば、主要コストは読み出しにかかる RA + RB だけになります。整列済みでなければ、整列コスト SA + SB がかかります。読み出しには O (n)、整列には O (n log n) がかかるので、全体として O (n log n) になります。このアルゴリズムが選択されることは、ほとんど、ありません。