実行計画のためのコスト計算式

説明対象のデータベース管理システムは Oracle です

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。

この資料のコスト計算式は、実際の製品で使われる計算式に完全に一致するものではなく、その基本となる考え方を説明するものです。

table access full

COST　NDB / k

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	

連続ブロックは、readv などのベクトル I/O を行うシステムコールで、メモリ上の非連続な複数バッファへ読み出されます。

index fast full scan

COST　NLB / k

index unique scan

COST　BLV + 1

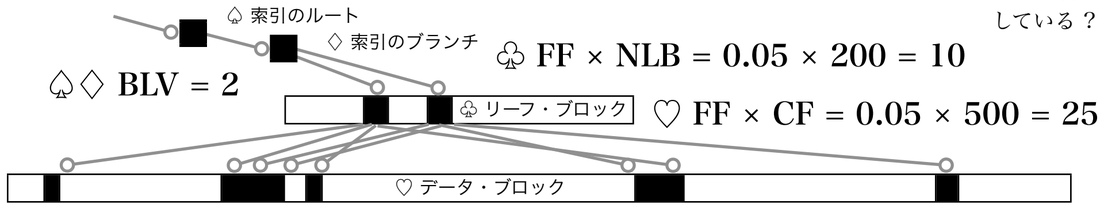
♢♧ コスト　♡ コスト

BLV	索引の深さ (≥ 0)	B-tree level
FF	データ選択率	filtering factor
CF	clustering factor	

index range scan

COST　BLV + (FF × NLB) + (FF × CF)

♤♢ 上位構造のコスト　♧ リーフ・ブロックのコスト　♡ データ・ブロックのコスト



指定された条件で、データ全体の 5 % を、索引経由で読むとします。選択率は FF = 0.05 です。索引の深さ BLV = 2 とし、リーフが 200 ブロック、CF = 500 とします。そうすると、索引のリーフは合計 10 ブロック、表のデータは延べ 25 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、ブロックの切り替わりを数えた延べブロック数を、CF として事前に計測しておきます。

NL: nested loop

COST　RX + (NX × RY)

一意検索では ♡ 索引ルートのコストを省略している？

NL 結合は、ふたつの表を結合する基本アルゴリズムです。外側のループで表 X を読み、内側のループで表 Y を読み、X と Y を結合します。

RX	駆動表 X を読み出すコスト
NX	駆動表 X の検索結果の行数
RY	被駆動表 Y 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 NX = 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。

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ません。平方計算量 O (n^2) は、データ量が増えるとあからさまに遅くなります。

よく使う計算量

定数	O (c)
対数	O (log n)
線形	O (n)
線形対数	O (n log n)
平方	O (n^2)

約 n 行のふたつの表のすべての行の組み合わせは約 n^2 行あります。

HA: hash

HA 結合は、索引の代わりに、表 X 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 Y の各行に対し、そのハッシュ表を検索することで、X と Y を結合します。

COST　RX + BX + RY

RX	構築表 X を読み出すコスト
BX	構築表 X からハッシュ表を作るコスト
RY	探索表 Y を読み出すコスト

表 X は構築表 build table とよばれ、表 Y は探索表 probe table とよばれます。ふたつの表の小さい方を構築表に選ぶとハッシュ表がメモリ内に収りやすくなり、BX も下がるので、全体のコストが下がります。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、NX が 1 万行だと、RY 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、NL の RY 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、そのようなときは、不適切な実行計画が選択されることがあります。

SM: sort-merge

COST　RX + SX + RY + SY

RX	表 X を読み出すコスト
SX	表 X を整列するコスト
RY	表 Y を読み出すコスト
SY	表 Y を整列するコスト

SM 結合は、整列されたふたつの表 X と表 Y 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。

この結合方式は、表 X と表 Y 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 RX + RY だけになります。整列済みでなければ、整列コスト SX + SY 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、全体の計算量も、O (n log n) になります。マージという語は、整列済みの列を合併して、整列済みの列をつくるときに使われます。