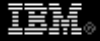



[ホーム](#) | [製品](#) | [サービス & ソリューション](#) | [サポート & ダウンロード](#) | [マイアカウント](#)

[DB2 Developer Domain](#) > [製品別技術情報](#) > [DB2いろはがるた](#) >

DB2いろはがるた



第44回 「ひ」-表スキャン、索引スキャンに変えたいな

今回はリレーショナル・データベースにとって、ハート中のハートといわれる、スキャン方法のお話しです。スキャンというのは、表の中にあるデータをDBMSが読み取る作業のことを指します。マニュアルではよく、日本語で走査と訳することがあります。



DB2ではオプティマイザーがSQL文を解釈して、もっとも効率のよいアクセスパス(つまりもっとも早く答えが得られるという意味)を、システム・カタログ表の中にある統計情報をもとにして作成するというのを、この連載で既にお話ししました。

リレーショナル・データベースでは、アクセスパスで、SQL文のパフォーマンスを考える場合、もっとも重要なのは、そのアクセス・パスが最善の索引を使っているかということにつきます。適切な索引が存在しない場合や、またはSQL文のWhere文節などで条件が指定されていない場合は、しかたありません。オプティマイザーは表の中の全ての行を最初から最後まで読み取ります。これを表スキャンと呼びます。たまた、表の中に存在する行数が極端に少ない場合は、索引があっても表スキャンの方のアクセス・パスが選ばれることがあります。

反対に、適切な索引が存在した場合、オプティマイザーは必ず索引スキャンを選ぼうとします。つまり索引を使って、欲しいデータをそのデータが存在するデータ・ページだけを直接さすことによって取り出します。索引スキャンと表スキャンのパフォーマンスの差は歴然としています。もちろん表のサイズによりますが、表スキャンが索引スキャンにかわるだけで、あつというまに応答時間が10倍・100倍以上になったという話がよくあります。なぜなら索引スキャンでは欲しい行を手にいれるために、表スキャンのように、最初から最後まで読みとりにいくようなことをしません。索引のBツリーの中には、リーフページと呼ばれる、キー値とそのキー値を含む行が存在している行へのポインターが保管されています。それによって、欲しい行を直接読み取りにゆくの。つまり、ある本の中である単語の意味を知りたい時に、本の最初のページから最後のページまで探すのではなくて、後ろの索引のページにあるその単語のページ番号から、直接そのページを開いてその意味を手に入れるようなものなのです。



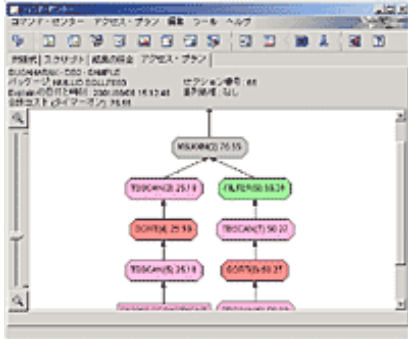
表スキャンさん

索引スキャンくん

さあ、2人のパフォーマンスは？
それぞれにカーソルを合わせてみてください。

SQL文のWhere文節で複数の条件を指定する場合があります。おのこの列に対して索引があるとします。その場合、どちらの条件を先に実行したほうがよいのか、すなわちどちらの索引を先に使ったほうがよいのか、ここがオプティマイザーの思案のしどころなわけです。それによっては、パフォーマンスが大幅にかわる可能性があるからです。どのような順番をとるかは、オプティマイザーは統計情報から判断します。また、複数の表を結合する場合も同じです。どのような結合方法をとるのか、どちらの表を外部表にするかで、パフォーマンスに重大な違いがでてきます。

DB2ではこのようなアクセス・パスを直感的に判断するために便利なGUIツールを提供しています。Visual Explainです。これでアクセス・パスを見ると、表スキャンなのか、索引スキャンなのか、またどの索引が使われているかなどが瞬時にわかる優れたもののツールなのです。ぜひ使ってみてください。下の例では2つの表の結合を表スキャンをつかってソート・マージ結合で行っています。



DB2ではその他に、索引を作成する場合にINCLUDE文節というものを指定することができます。これは、表スキャンを意図的に索引ONLYスキャンにしたい場合に使います。つまり、下のようなSQL文があるとします。

```
Select name, age from employee where name=smith
```

Name列に索引が作られているとします。この場合普通の索引では、nameの値は索引リーフの中にあるので、結果の値はそこから即座に手にいれられますが、ageはそのname索引リーフの中のポインターがさすデータ・ページを読み込んでage列を含む行を取り出す必要があります。その場合、I/Oの数としては、索引を読むのに最低一回(当然階層の数によります)、データ・ページを読むのに1回かかります。

そのような場合、Includeを索引に指定すると、データ・ページをよまなくてもよくなります。つまり下のように索引を作ります。

```
Create index ix1 on employee(name) include age
```

こうすることによって、age列も索引のリーフページに含まれます。必要なデータ、name, ageが索引ページを読むだけで手に入れることができるのです。ただし、このage値はユニークである必要はありません。索引リーフに含まれても、それがキー値ではないわけです。

くどいようですが、リレーショナル・データベースではパフォーマンス・チューニングを考える場合は、この索引スキャンがキーになります。いかに最適な索引を指定できるか！適切な索引の設計が、データベース設計の中心といっても過言ではありません。

[↑ 上に戻る](#)