


[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18回 「そ」-ソート処理、ちっちゃい方から並べましょう



執筆者

春野 さくら

「DB2いろはがるた」を執筆するために参上した、なぞの女性。日本の古典文学を愛する。

今年の夏もいよいよまっさかりになりました。みなさんは夏休みをとられました？ 実はこのコラムも来週は夏休みです。お休みもとれずに、一生懸命はたしている方はごめんあそばせ。ほっほっほ！！勝ち誇った笑いですみません。今回はDBMSのソート処理のお話です。



ソート処理は、日本語で分類処理ともいいます。おおもとの意味はデータを値の順番に並び替えることを指しています。フラットファイルの中のデータを順番に並び替えるソート製品というの、市場でよく売られています。リレーショナル・データベースの基本的なコンセプトでは、データの物理的な配置は、論理的な構造と独立しています。つまり、同じSELECT処理を実行したとしても、DBMSがいつでも同じ順番でデータを戻すことを保証しません。そこで戻されるデータの順番を保証するために、考えられたのがSELECT文のOrder By 文節です。例えば、社員名簿表から入社が早い順番に従業員の名前をとりだすには、次のようなSQL文を実行しなければなりません。

```
SELECT empno, last_name FROM employee Order By hiered_date
```

DB2では、ソートされる情報がsortheap(ソートが実行されるたびに、データベースに接続されている各プロセスに割り当てられるプライベート・メモリー)に完全におさまらない場合、その情報はオーバーフローしたといい、一時表スペースの中の一時表にそっ・といわれます。(くだらない駄洒落ですみません) また、最終的なソート済みのデータを保管するための一時表を必要としないで、ソートした情報を直接プログラムに戻すことができる場合、そのソートはパイプ・ソートと呼ばれます。当然、オーバーフローしないソートは、オーバーフローするソートに比べて速くなり、パイプ・ソートは非パイプ・ソートよりも速くなります。DB2のオプティマイザーは、予想される結果セットと、SORTHEAP DB構成パラメーターを比較して、非オーバーフローソートを実行できるか、およびパイプ・ソートを実行できるかなどを判別してアクセスパ

スを作ります。

このソート処理は当然かなりCPUを使う処理です。もし、Order Byで指定する列に索引がすでにある場合は、データを取り出した後で並び替える必要はなくなるので、ソート処理は必要なくなります。この場合、データ自身がその順番でならんでいれば、もっとパフォーマンスはよくなります。この目的のために作成されるのが、クラスター索引とよばれるものです。

DB2のVisual Explainツールを使うと、オプティマイザーが行ったソート処理がひとめでわかります。このツールからは、おてんと様の下では誰も逃れらことはできません。例えば、次の図では索引がついてない2つの表をソート・マージ結合を行っている例です。

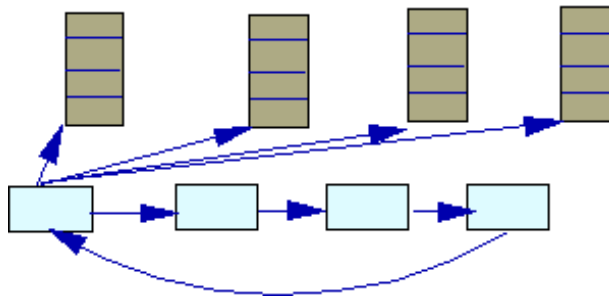


クリックで拡大(17K)

DB2 UDB サーバー上でソートを行うために割り当てられるメモリの合計量を制御するDBMS構成パラメーターもあります。これはSHEAPTHRESと呼ばれ、一つの時点で割り当てる必要があるすべてのソートヒープの合計量です。実は多重処理のところでお話したイントラ・パラレルが使える場合、ソート処理も並列に行われます。パラレル・ソートにはいろいろな種類があるので

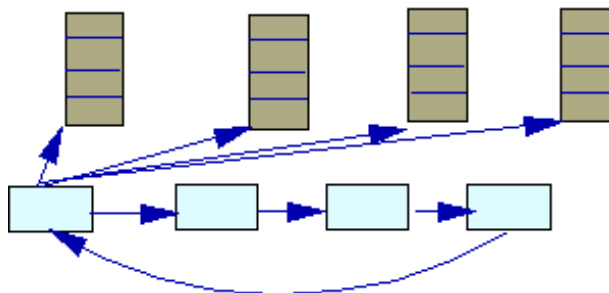
ラウンド・ロビン・ソートはデータを全てのサブセクション・ピース(サブエージェント)に再分配する効果的なシェアード・メモリー・ソートです。これはラウンド・ロビン・タイプのアルゴリズムをデータの平等分配の為に使用しています。ソート結果領域はそれぞれのサブエージェントが含まれるメモリーの中に作成されます。ソートの挿入段階の間、サブ・エージェントはデータをそれぞれのソート結果メモリー領域の中に挿入するのを交代して行います。

平等再分配



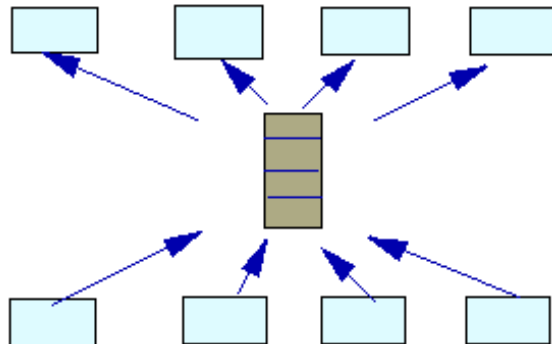
パーティション・ラウンド・ロビン・ソートは、ソート結果がそれぞれのサブエージェント毎のプライベート・メモリーの中に作成されるもう一つのラウンド・ロビン・ソートです。違う点はサブ・エージェントは、ソート列をハッシュして行を挿入するソート結果領域を決定します。これは列の値によってデータを分けて、ソート結果は後でマージ・ジョインや集合の演算で使用することができます。

ハッシュ再分配



レプリケートッド・シェアード・ソートは全てのサブエージェントがソートの結果の全ての行を読む必要がある場合に使用されます。1個のソート結果領域がメモリの中に作成され、その後複数のサブエージェントは同期をとりながら、ソート結果領域に書き込みます。ソートが完了した時、それぞれのサブエージェントの次の演算はソート結果の全体を読みます。これは表の行の数が少ない時に使用されます。

平等再分配



動的シェアード・ソートは、サブエージェントの次の演算がソート結果領域にパラレル・スキャンでオープンをかける以外は、レプリケートッドと同じです。シェアード・ソートは、次の演算で全てが同じ量の処理データを受取り、処理されるデータがサブエージェントをまたがって平等に分配されるという点で、ラウンド・ロビン・ソートと同様に作用します。

DBMS構成パラメーターのsheapthresはプライベート・ソートとシェアード・ソートでは使用方法が異なります。プライベート・ソートの場合、このパラメーターは一時点でプライベート・ソートが消費できるメモリの合計量に対する、インスタンス・レベルのソフト・リミットになります。(ソフトという意味は、この限界に達すると、それ以降のプライベート・ソート要求に対して割り当てられるメモリは大幅に削減されるということです。)シェアード・ソートの場合、このパラメーターは一時点でシェアード・ソートが消費するメモリの合計に達すると、データベース・レベルのハード・リミットになります。(ハードという意味は、この限界に達すると、以降のシェアード・ソートのメモリ要求はシェアード・ソートの合計メモリ消費量がSheapthresに指定された限界を下回るまで許可されなくなるということです。)Sheapthresは、そのインスタンス内のデータベースの最大のsortheap値の少なくとも2倍にする必要があります。

いやあ～、一口にソートといっても結構深いものがあります。こうやって並べてみるとあらためて感動します。そういえば、数年前にトロント研究所のDB2開発部門でソート処理の機能を開発している人に会ったのですが、IBMはソート処理で業界最先端の技術の特許申請しているんだとっていました。その人は結構オタクばい人で、ソート命という感じで、やっぱりこういう人でなくっちゃと妙に納得しました。彼も今ごろますます、ソート技術に磨きをかけていることでしょう。

[↑ 上に戻る](#)