データベースにおける大量データの取り扱いについて

ソリューション本部　ソリューションサービス部

冨田　裕二

1. はじめに

HDD の大容量化と低価格化により、以前と比較し、サーバーHDD の大容量化が安価に実現出来るようになった。

そのため、顧客も大量のデータを取り扱う事に抵抗がなくなり、また、提案する側も、大量データを取り扱うシステムの提案がしやすくなった。その反面、取り扱うデータ量の増大に伴う処理速度の劣化が問題となる例もある。

当社の開発したシステムに、マスタを使用して、動的に見積もりを作成するシステムがある。

ある顧客においては、マスタの作成に複雑な計算が必要であり、コスト、安全の観点から別途構築したマスタ作成システムで作成する必要があった。作成されたマスタは、作成当時のデータとして使用する必要があり、データベース（以下、DB）内に蓄積されつづけた。

その結果、データは膨大なものとなり、マスタ作成システムの処理速度が劣化するという問題が発生した。

その問題を解決した際の話を基に、大量データを取り扱う際に気を付けなければならない点を紹介する。

2. 実行環境

データベースエンジンには「Microsoft SQLServer 2008 R2」を使用した。取り扱ったデータ容量は1 TB 程で、その中で最大のテーブルサイズは、データ件数が 5 億件程度（200GB）であった。

3. 発生した問題および解決に至るまで

発生した問題の概要及び、解決に至るまでの対応内容を時系列順に説明する。

3. 1　インデックス不足による処理速度劣化

DB でデータを扱う際には、インデックスの設定は必須であり、非常に重要である。

試験を行った際のデータ件数と、システムが稼働後蓄積されたデータ件数に、大きな乖離があったため、DB 設計の際に検討したインデックスでは、処理に対して十分ではない事を発見できなかった。不足したインデックスを追加した事で処理速度の改善は見られたが、それだけでは全ての問題の解決には至らなかった。

3. 2　ロジック不備による速度劣化

インデックスの追加だけでは、まだ、顧客の要求する処理速度は実現できなかったため、次にロジックの見直しを行った。

処理の区切りごとに、ボトルネックとなっている処理の洗い出しを行った。その結果、レコード件数の多いテーブルからデータを取得する SQL で、

UNION ALL 句、NOT EXISTS 句を使用していたことが問題であることが分かった。

上記の 2 つの演算子を使用した事で、メモリの使用量が増加し、処理速度の低下につながっていた。図 1 のように処理を分割する事で、メモリの使用量を抑え、約 67%の速度改善が見られた(図2)。

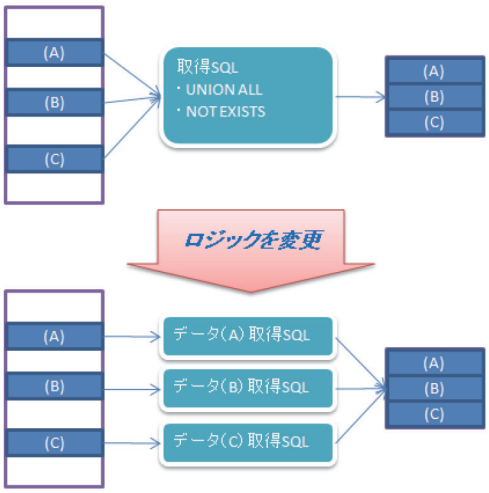

図 1 ロジックの変更内容

この問題も試験時のデータ件数とシステム稼働時のデータ件数の乖離により、試験時には発見できなかった。

SQL 文の作成時には、最終的なデータ量の見積もりも考慮し、使用する演算子(特に、UNION ALL と NOT EXISTS)には注意する必要がある。

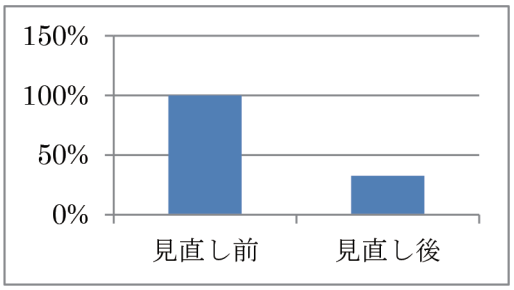

図 2 ロジックの見直しによる効果

3. 3 不要なデータの蓄積による速度の劣化

ロジックの見直しの際に、データの増加に伴い、処理速度の劣化が発生したのであれば、不要なデータを削除し、データの件数を減らすことで速度の改善が見込めるのではないかと考え、DB に残すデータの制限について検討した。

その結果として、不要となったデータを削除する事で、約 40%の処理速度の改善がみられた(図3)。 残せるデータは可能な限り残す、とした設計であったが、DB に保存すべきデータは必要最小限にとどめるべきであり、データの精査は必要であった。

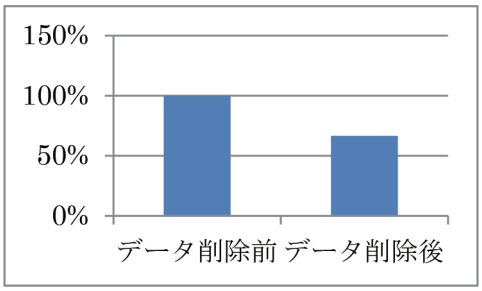

図 3 不要データの削除による効果

3. 4 tempdb の設定不足による速度劣化

インデックスとロジックの見直し、及び DB のサイズ縮小を行ったが、それだけではまだ目標をクリアできず、次に着眼したのは、ミドルウェアのチューニングであった。

SQL Server は、tempdb という、「グローバルまたはローカルな一時テーブル、一時ストアドプロシージャ、テーブル変数、カーソルなど処理中に保持する、インスタンスに接続しているすべてのユーザが使用できるリソース」[1]を持つ。

複数のクライアントからの利用を簡易に制御するために、一時テーブルを使用したシステムを構築

したが、tempdb の設定が初期値のままとなっており、そのことが処理速度低下の要因となっていた。

CPU のコア数に合わせ tempdb の物理ファイルを増設し、初期ファイルサイズを十分なサイズに拡張する対応を行った。初期ファイルサイズが小さい場合、処理データ量が大きいと、DB の自動ファイルサイズ拡張機能により、高頻度でファイル拡張が行われ、速度劣化につながる。

適切な設定値に変更する事で、約 53%の理速度の改善がみられた。(図 4)。

ただし、tempdb の最適化は、端末に依存する。

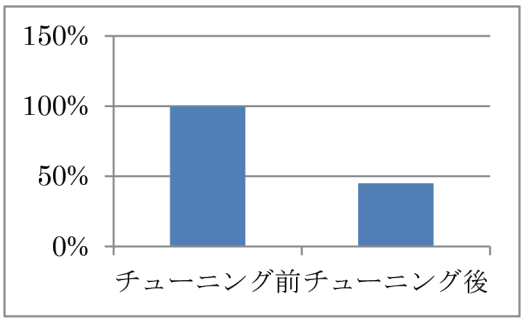

図 4 tempdb の最適化による効果

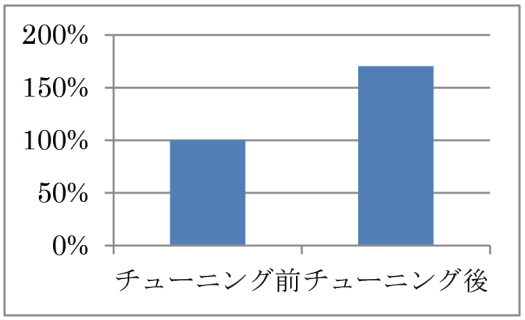

図 5 tempdb の設定が最適でない場合の処理速度比

図 5 は、図 4 で示した tempdb のチューニング内容を、CPU コア数の少ない環境に適用した場合の結果である。端末に適したチューニングを行わないと、結果的に処理速度の劣化を招く可能性がある。

3. 5 I/O による性能劣化

上述の対応により、当初よりも格段に処理速度は改善されたが、さらにもう一段の速度改善を顧客から要望された。

SQL Server 2008 R2 には、パフォーマンスデータコレクションというグラフィカルにパフォーマンス監視できる機能がある。詳細は割愛するが、その中で、クエリ統計を確認できる機能がある。この機能を使用して、さらに改善出来る個所が無いかの調査を行った。

クエリ統計では、過去の指定した時間内のクエリの統計情報を確認する事が出来るため、チューニング対象の発見に非常に有益である。

図 6 は、実際のクエリ統計より取得した結果を見やすいようにしたものである。1 秒間あたりの累積実行時間の大きいもの上位 10 クエリをレポートとして確認する事ができる。

上位 10 クエリの中で、累積実行時間が最も大きかったクエリの詳細な情報より、実行ごとの処理時間の項目を抜粋したものが図7となる。

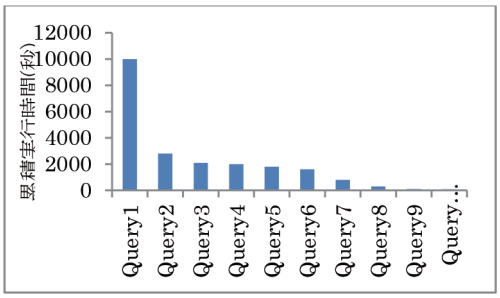

図 6 クエリ毎の累積実行時間

実行ごとの平均CPU時間(ミリ秒):	0.1
実行ごとの平均実行時間(ミリ秒):	0.8
実行ごとの物理読み取りの平均数:	0
実行ごとの論理書き込みの平均数:	0

図 7 クエリの実行時間の内訳

クエリの実行時間は、平均実行時間に対し、CPU 実行時間が小さいことから、CPU 処理以外の部分で実行に時間がかかっている事が解った。このことから、何かしらの待ちが発生していると推測し、どのような待ちが発生しているかを確認した結果が図8となる(待ちも、クエリ統計より確認できる)。

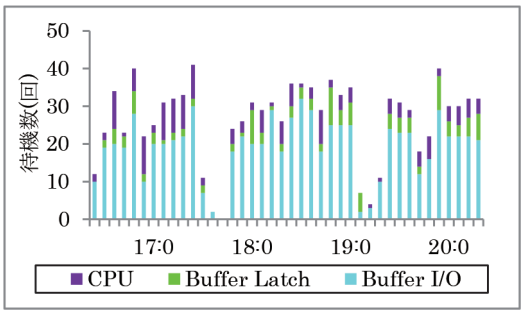

図 8 クエリの待機数の内訳

待機数を確認すると、Buffer I/O(I/O 待ち)が大きな割合を占めている事が解った。

原因を調査すると、SQL Server のシステム領域と、データ領域が同じドライブに配置されていたためではないかという事が推測された。しかし、ドライブの変更を行うためには、システムを停止する必要があり、その実施は運用上、非常に困難なため、顧客と協議の結果、実施を見合わす結果となった。

3. 6 その他諸問題

今回、大量データを取り扱うこととなったため、システムとしては、処理速度の問題が発生した。しかし、他に「時間」が問題となったケースを紹介する。

(1) 検証用データの用意に時間がかかる

データ量が多いため、問題の検証を行うためのデータを用意するのにも時間がかかってしまう。実施計画を立てる段階で、正確にデータ量を想定し、必要な検証用データを作成する時間を見積もっておく必要がある。

(2) バックアップに時間がかかる

障害の対応のため DB に変更を実施する際は、DB のバックアップを取得する必要がある。しかし、データ量が多い場合、そのバックアップに時間がかかってしまい、作業計画に致命的な影響を及ぼす事がある。事前に、データ量とバックアップにかかる時間を把握したうえで、作業実施計画を立てる必要がある。

4. 最後に

今回、データが増加したことによる速度劣化を中心に事例を紹介した。これらの問題の解決には非常に長い時間を要した。

今回の件を踏まえ、システムを設計する場合は、最低限、以下の点に注意したい。

1) 将来的なデータ件数を見越して、試験を行う。
2) SQL 文構築時には十分に吟味する(特に、UNION、NOT EXISTS は要注意)。
3) データベースエンジンのチューニングは端末の仕様に合わせて適切に設定する。
4) データベースエンジンのインストール時にはシステム領域とデータ領域を分ける。

今後も、大量のデータを取り扱う案件は増えてくると考えている。自分の得た経験を共有し、大量データを取り扱える人員の育成に貢献できたらと考えている。

＜参考文献＞

1) 「Microsoft Developer Network tempdb データベース」(http://msdn.microsoft.com/ja-jp/library/ms190768(v=sql.105).aspx)