データベースシステムの性能問題解決のための一手法

4E－1

町原　宏毅
ＮＴＴ情報通信研究所

１．はじめに

情報システムの開発の中で行われるデータベースの設計では、特定の業務処理プログラム（AP）の性能を満たすように、あえて正規形を崩して冗長性を持たせたり、インデックスというデータベース管理システム（DBMS）特有の機能を用いるようなさまざまなチューニングを行う[1][2]。

従来このようなチューニングを行うには、DBMSの特性や内部処理などさまざまな知識を必要としており、専門家が行ってきたのが現状である。しかしチューニングのためのノウハウは多数存在するものの、それらの効果は環境によって異なるためその適応順序については専門家によって異なり、安定した品質を得ることは困難であった[3]。

そこで本稿では、複数存在する性能チューニングのための設計項目をその修正稼働と修正した場合の効果について分析し順序付けることにより、最も効果の高いと思われる手順を求めた結果を示す。

２　設計チューニングの流れ

データベース設計におけるチューニングの難しさは、複数のAPが１つのデータを共有しうるためにある設計が全てのAPにとって最適な設計にはなりえないことにある。

そこで全体的な性能のバランスを考慮に入れながらチューニングを行わなければならない。本稿で提案する手順は、まず設計者が提案する情報を基に性能評価を行い、性能問題の検出されたAPを抽出し、そのAPを高速化するための設計変更案を検討し、もう一度性能評価を行っていくものであり、この作業を繰り返すことにより性能条件を満たすデータベースの設計が実現できるようにするものである。

チューニングの流れを図１に示し、その概要を次説で記述する。

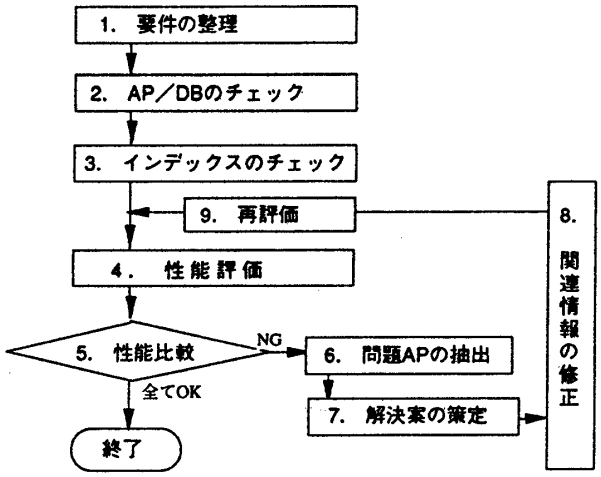

図１　チューニングの流れ

３　手順の概要

3.1　要件の整理

ここでは開発対象としている情報システムについて以下の3つの情報を整理する。

・テーブル情報の整理

列の属性や桁数、テーブルの行数、値の分布状態などを調べる。

・業務情報の整理

各APがどのようなデータベースアクセスをどのような順序で行っているかを調べ、それらがどのようなタイミングで発行されるかについても調査する。

・要求される性能条件の整理

各APがどのような性能（スループットと応答時間）を満たさなければならないのかについて調査する

3.2　AP/DBのチェック

ここでは、事前にAPやデータベースを調査し、明らかに負荷の高い処理やDB構造上無駄の多いテーブルなどをチェックし、必要に応じて修正する。

3.3　インデックスのチェック

DB構造や検索条件などから一般的に効果があると思われる列に対してインデックスが付与されているかをチェックし、必要に応じて修正する。

3.4　性能評価

同一タイミングで走行されうる業務を同時に走行させてみた場合の性能を評価し、各APの応答時間／スループットを求める。（性能の評価方法としては実測させたり、机上で計算したりするなどの方法があるがここでは規定しない）

3.5　性能比較

性能条件と性能評価結果とを比較して、各業務がそれを満たしているかどうかを比較する。

3.6　問題APの抽出

性能条件と性能評価結果とを比較した結果、比率的に最も悪い業務を抽出する。

3.7　解決案の策定

データベースシステムを構築する場合の設計項目は独立に行われるのではなく、性能に関して複雑に関係しあい、影響を及ぼしあう。

A Database Design Methodology for Solving Performance Problems.
Hiroki MACHIHARA
NTTInformation and Communication Systems Labs.

そこで各設計項目に対して変更を行った場合の影響度を調査し、その少ない項目から自由度の高いものと見なした（図２）。さらに各設計項目に対して変更を行った場合の性能改善の効果について過去の経験から順序付けを行った。

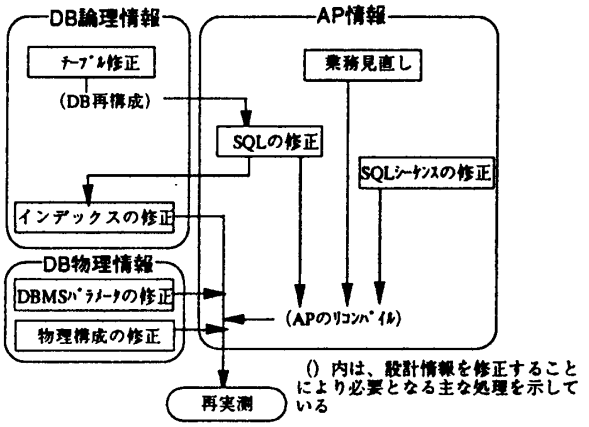

図２　設計情報間の影響範囲

各設計項目に対して上記２つの観点で分析した結果、最も効果的と思われる自由度が高くかつ効果の大きい項目から性能改善を行っていく実施手順を求めた（図３）。

データベースに関する設計項目			チューニング 自由度	チューニング 効果	改善順位
DB	分析	正規化	正規形を作成	−	−	−
	テーブル設計	RDB化	リレーショナルモデル	−	−	−
		統合／分割	冗長性を持たせる	×	△	7
	列設計	属性設計	型、桁数の設計	○	△	
		統合／分割	列の分割／統合	×	△	
	インデックス設計		どの列に付与するか	○	◎	2
	DBMSﾊﾟﾗﾒｰﾀ	アクセス設計	負荷分散	○	○	5
			メモリDB化	○	○	
		DBﾊﾟﾗﾒｰﾀ	ﾎﾞﾘｭｰﾑ割り付け	○	△	
		DBMSﾊﾟﾗﾒｰﾀ	ユーザバッファ	○	△	
業務	SQLの記述	特異な処理	負荷の高い処理をなくす	○	◎	1
		記述方法	カラムや条件の指定順序	△	○	3
	SQLのシーケンス		結合処理と副照会の利用	△	○	4
	APの実行パターン		実行順序とプロセス数	△	○	6
その他	非DB化		DBをﾌｧｲﾙとして扱う	○	○	8
	運用対処化		性能条件を緩くする	○	○	
	設備設計		高CPU、高速DISKにする	○	○	

図３　性能改善施策分類表

1) 「特異な処理」の見直し
＜明らかに負荷の高い処理をしていないか？＞
→SQLを修正して再実測
2) 「ｲﾝﾃﾞｯｸｽ設計」の見直し
＜最適なインデックスが付いているか？＞
→インデックスを修正して再実測
3) 「SQLの記述方法」の見直し
＜アクセス方法にそった記述になっているか？＞
→SQLを修正して再実測
4) 「SQLのシーケンス」の見直し
＜無駄のないシーケンスになっているか？＞
→APを修正して再実測
5) 「DBMSパラメータ」の見直し
＜最適なパラメータになっているか？＞
→パラメータを修正して再実測
6) 「APの実行パターン」の見直し
＜同時走行APで同一ﾃﾞｰﾀの待ちが起きてないか？＞
→SQL、インデックスを修正して再実測
7) 「テーブル／列設計」の見直し
＜結合処理が多発するか？＞
→テーブルを修正して再実測
8) 「その他」の見直し
＜負荷の高いｵﾝﾗｲﾝ処理が存在するか？＞
→ﾊﾞｯﾁ処理にして再実測

3.8　関連情報の修正

解決策が導き出されたらその設計情報を１つの候補とし、その設計候補にした場合に必要となる関連する情報の修正を行う。

設計情報間の影響範囲については、図３に示す。

3.9　再評価

設計候補にもとづいて関連情報を修正したら、もう一度評価を行ってみる。この後、同じようにチューニングを繰り返し実行していく。

4．チューニングの限界

データベースを用いた情報システムでは１つのデータを複数のAPで利用することがある。そのため性能を評価した結果、問題箇所が特定され設計を変更した場合に、今まで問題のなかったAPが悪くなってしまう場合もある。つまり設計の変更が複数のAPに影響を及ぼし、それを改善しようと変更するとまた元のAPが悪くなってしまうことがある。

このように設計だけではどうしても解決できないような場合は、業務自身を見直してもらったり、性能要件を緩めたり、あるいは良い高い能力のハードウエアで対処する必要もでてくることは避けられない。

5．まとめ

データベース設計を行うにあたって、どのような手順で実施していけばより効率的に性能要求を満たす設計ができるようになるかについて示した。ここで示した手順は、厳密に言えば利用するDBMSや与えられた環境によって異なるため、如何に一般化するかが重要な課題となる。今後は、試行錯誤的にチューニング作業を繰り返していく場合に利用する評価環境やツールについて検討していく予定である。

参考文献

[1]大久保、町原、他：" ﾃﾞｰﾀﾍﾞｰｽ論理設計" ，阿部出版，1993
[2]味村，山田，堀内：" ﾃﾞｰﾀﾍﾞｰｽｼｽﾃﾑの設計と開発" ，ｵｰﾑ社，1983
[3]P.Corrigan：" ｵﾗｸﾙﾊﾟﾌｫｰﾏﾝｽﾁｭｰﾆﾝｸﾞ" ，ｵｰﾑ社，1994