オブジェクトレベル負荷情報導出の効率化について

富士通キャドテック 株式会社

■ 執筆者Ｐｒｏｆｉｌｅ ■

新　井　暁　士

1998 年　富士通キャドテック株式会社入社
富士通通信部門のデータベース管理業務に従事
2003 年　富士通 GLOVIA プロジェクトのインフラ整備業務に従事

■ 論文要旨 ■

データベース（※１）チューニングの要素の一つである負荷分散は，入出力時の負荷が大きいデータベースオブジェクト（テーブルやインデックスなど）を，それぞれ異なる物理領域に配置することで行われる．だがこの作業は，膨大な量の情報を分析しなければ実行できないため，大変な手間を要していた．

そこで当作業の円滑な遂行を目的に，以下の二項目に着眼して，ツール開発を含めた新たな手法の確立を図った．

① 負荷が大きいデータベースオブジェクトの識別
② 競合を起こしているデータベースオブジェクトの識別

当論文では①②の導出手法，及び手法に関する課題や今後の展望などについて解説する．

※１）
Oracle 社製のデータベースに限定する．

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

１．１　背景

これまで富士通（株）通信部門における基幹システムのデータベース管理者として，主に Oracle 社製データベース（以下，Oracle という）の管理業務に携わってきた．

トランザクション同士の入出力競合を解消し，スループットの向上を図る負荷分散作業は，特に重要なチューニング要素の一つとして位置づけられる．当作業では，テーブルやインデックスなどのデータベースオブジェクト（以下，オブジェクトという）を，効果的に分散させることが求められるが，そのためにはトランザクションによる入出力負荷が大きいオブジェクトや，競合を起こしているオブジェクトを判別しなければならない．

判別には最低でも以下の二つの情報が必要である．

① トランザクションの対象となったオブジェクトの名前．
② そのオブジェクトに対して入出力があったという事実，及びその発生頻度．

①②の情報をオブジェクトレベル負荷情報と仮称する．本来①②のような管理情報は，Oracle のデータディクショナリ（**※２**）や，動的パフォーマンスビュー（**※３**）から得ることができるが，①②に関して得られる情報は少なく，これを補う新たな情報源の確保が必要となった．

そこで普段からロギングやトレーシングなどを用い，日々のトランザクションに関する情報を収集～分析し，①②を推し測っていたが，収集した情報には不十分な点が多く，①②を得るまで非常に時間を要していた．

システムの導入部門が増えるにつれ，管理対象データベースの数も 50 を越え，管理上の限界に達したことから，より効率的なチューニング手法を作成し，問題の解決を図った．

以上の背景を経て確立した，効率的なオブジェクトレベル負荷情報の導出手法について，当論文では解説する．

※２）
Oracle の管理情報（メタデータ）の集まり．物理構成や論理構成に関する情報が格納されている．

※３）
Oracle がオープンしている間，管理情報が継続的に書き込まれるビュー．データディクショナリと同様，Oracle の管理に用いられる．

２．新手法解説

新手法では，以下の二つのプロセスを踏んで，オブジェクトレベル負荷情報を導出することにした．

① ログマイナー実行フェーズ
② ツール実行フェーズ

２．１ ログマイナー実行フェーズ

ログマイナー実行フェーズでは，Oracle の一機能であるログマイナーを用いて，オブジェクトレベル負荷情報の情報源確保を行う．

ログマイナーは Oracle に対して行われた変更（**※４**）を SQL レベルで情報化する機能であり，オブジェクトレベル負荷情報を導出するうえで十分な情報源を確保することができる（ログマイナーに関する詳細は付録を参照）．

※４）
挿入（INSERT），更新（UPDATE），削除（DELETE）を指す．

２．２ ツール実行フェーズ

ツール実行フェーズでは，ログマイナー実行フェーズで確保した情報の精査～表示を行う．

ログマイナー実行フェーズで確保した情報は，動的パフォーマンスビュー V$LOGMNR_CONTENTS に格納される．これらの情報には冗長な情報が多量に含まれているため，冗長な情報の排除，及びオブジェクトレベル負荷情報への変換を，当方開発のツールにより行う．ツールは変換した情報を当方設計のテーブル O3$CENTER に格納する．

V$LOGMNR_CONTENTS から O3$CENTER への変換ルールを以下の**表１**に示す．

表１　オブジェクトレベル負荷情報への変換ルール

	変換元の列	変換先の列	列の内容	変換ルール
①	TIMESTAMP	TIME	変更が行われた時刻.	DATE 型から CHAR 型(YYMMDDHH24MI 形式)に変換.
②	SEG_OWNER	OWNER	変更の対象となったスキーマの名前.	変換せずにそのまま移行.
③	SEG_NAME	NAME	変更の対象となったオブジェクトの名前.	変換せずにそのまま移行.
④	OPERATION	OPERATION	変更の種類.	変換せずにそのまま移行.
⑤	SQL_REDO	DETAIL	変更に用いられた SQL.	冗長部分を排除して列名と列値だけの記述に変換.
⑥	SQL_UNDO	ORIGINAL	変更を取り消すために用意された SQL.	冗長部分を排除して列名と列値だけの記述に変換.
⑦	－	QUANTITY	同一の SQL による変更が１分以内に行われた回数.	TIMESTAMP, SEG_OWNER, SEG_NAME, OPERATION, SQL_REDO, SQL_UNDO によるグループ化でまとめられた行の数を設定.
⑧	（以降の列）	－	－	変換、移行を行わない.

表１の各変換は以下の三つの条件のもと行われる.

・SEG_OWNER が SYS, SYSTEM, UNKNOWN ではない.
・SEG_TYPE_NAME（オブジェクトの種類）が TABLE である.
・OPERATION が INSERT, UPDATE, DELETE である.

表１の①～③, ⑦の変換により, インデックスを除く全オブジェクトの負荷情報が導出される. インデックスに関する負荷情報を導出するには, ①～⑦の全変換が必要である.

インデックスは, テーブル検索のパフォーマンスを向上させるオブジェクトであり, テーブル内の情報の一部（キー列値）を格納している. この情報（テーブル）に対して変更が行われると, 関連づけられたインデックスにも同様の変更が行われる. ただし条件によっては変更が行われない場合があり, ツールはこれを④～⑥の変換で判断する. 一連の条件を以下に示す.

・テーブルに対して行われた変更の種類が挿入（INSERT）であった場合, 変更後の全キー

列値が NULL（※５）である．

・テーブルに対して行われた変更の種類が更新（UPDATE）であった場合，変更前／後の全キー列値が NULL である．

・テーブルに対して行われた変更の種類が削除（DELETE）であった場合，変更前の全キー列値が NULL である．

※５）

何も値が入っていないこと．

このように，インデックスでは多くの情報を以上の条件のもと変換しなければならないため，ほかのオブジェクトよりも時間がかかる．なお一度導出したオブジェクトレベル負荷情報は，O3$CENTER に履歴情報として格納されるため，再度変換する必要がない．

オブジェクトレベル負荷情報は，以下の GUI を介してグラフ化される．

図１　GUI タイトル画面

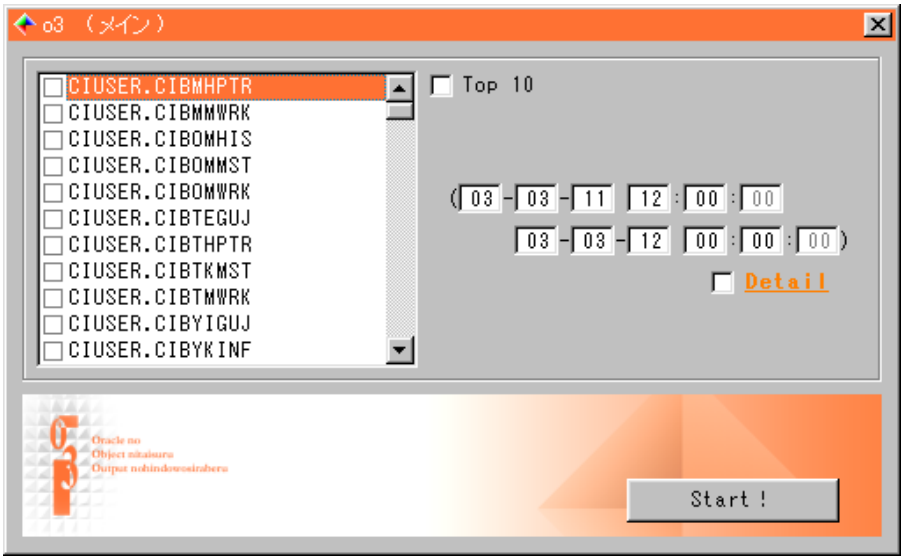


図２　表示条件指定画面

ツールは，**図１**のタイトル画面が表示されている間に，オブジェクトの名前とタイムスタンプを読み込む．前者は O3$CENTER の NAME（**表１**③）に格納された重複分を除く全名前であり，後者は TIME（**表１**①）に格納された最新／最古のタイムスタンプである．読み込まれたこれらの情報は，**図２**の表示条件指定画面に初期表示される．

ユーザは表示条件指定画面でグラフ化したいオブジェクトの名前を選択し，開始／終了時刻によってグラフの表示範囲を指定する．こうして得られたグラフが**図３**である．

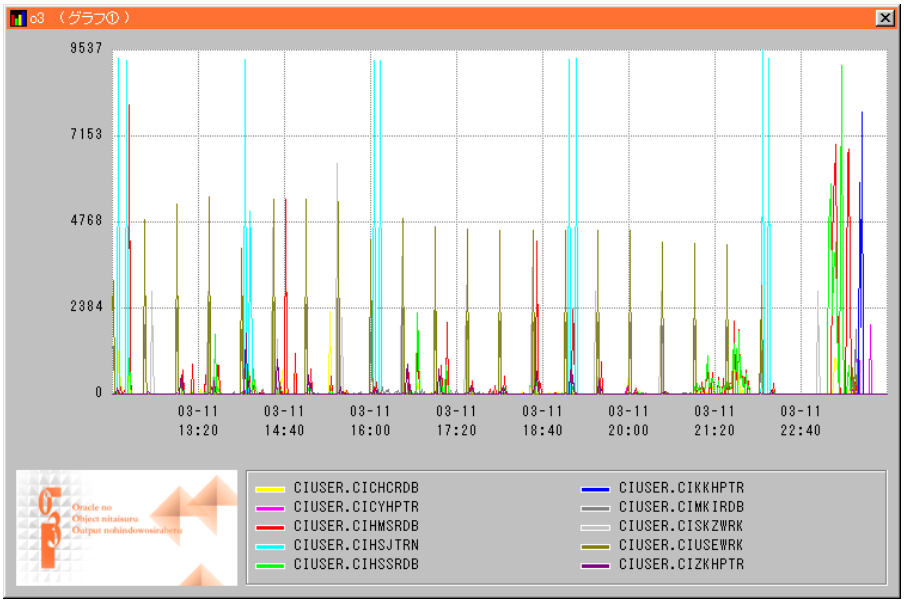


図３　結果表示（グラフ）

グラフでは，オブジェクトに対する負荷が折れ線で表される．一枚のグラフには複数のオブジェクトを含めることができ，負荷が大きいオブジェクトや競合を起こしているオブジェクトを一目で識別できるようになっている．また未知のシステムでもボトルネックとなっているオブジェクトを即座に識別できるよう，負荷が大きいオブジェクトを自動で選択～グラフ化する機能を持たせている．

ツールには，オブジェクトレベル負荷情報を導出～グラフ化する以外にも，様々な機能を持たせている．

図２の表示条件指定画面のチェックボックス Detail をオンにすると，Excel を介してより詳細な情報を生成することができる．Excel による結果表示を以下の**図４**に示す．

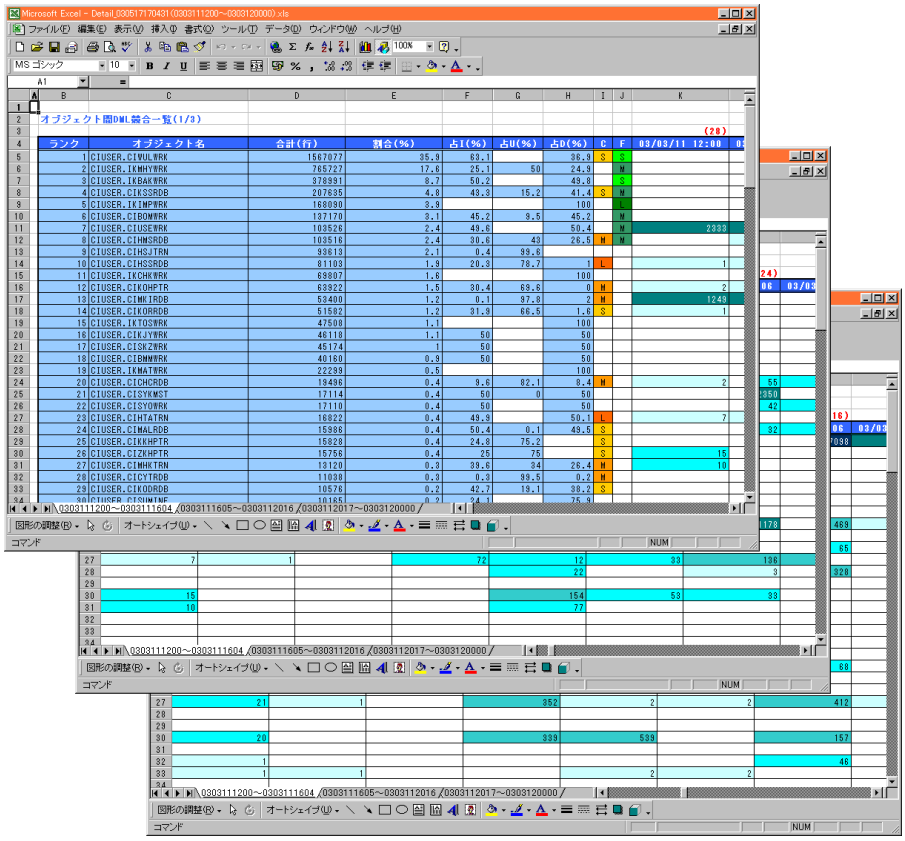


図４　結果表示（Excel シート）

シート１列目）
　列名：ランク
　列の内容：オブジェクトの順位．順位が上がるほど，そのオブジェクトに対する負荷は大きくなる．

シート２列目）
　列名：オブジェクト名
　列の内容：オブジェクトの名前．一意な名前となるようスキーマ名が付加される．

シート３列目）
　列名：合計（行）
　列の内容：変更行の合計．**図２**の表示条件指定画面に初期表示された開始／終了時刻の範囲で合算される．

シート４列目）
　列名：割合（%）
　列の内容：変更行の合計が全体に占める割合．

シート５列目）
　列名：占Ｉ（%）
　列の内容：変更のうち挿入（INSERT）が占める割合.

シート６列目）
　列名：占Ｕ（%）
　列の内容：変更のうち更新（UPDATE）が占める割合.

シート７列目）
　列名：占Ｄ（%）
　列の内容：変更のうち削除（DELETE）が占める割合.

シート８列目）
　列名：C
　列の内容：競合を起こしているオブジェクトに付加されるマーカ．S（Small），
　　M（Middle），L（Large）の三段階で表される.

シート９列目）
　列名：F
　列の内容：フラグメンテーション（**※６**）対応の必要があるオブジェクトに付加される
　　マーカ．S，M，L の三段階で表される.

シート 10 列目以降）
　列名：（日付と時刻）
　列の内容：変更行数．分単位で表される.

※６）
　変更を繰り返すことによって生じる領域の虫食い現象．当現象が進行すると，断片化した小さな空き領域が各所に点在することになり，連続領域を必要とする大きな情報を書き込めなくなる．フラグメンテーション対応は当現象を解消するために行われる.

　Excel では，各オブジェクトに対する負荷が時系列に沿って細かく表される．各オブジェクトは負荷の大小によってソートされ，負荷の大きいオブジェクトほど上位に表されるため，ボトルネックとなっているオブジェクトを即座に識別することができる．また Excel シートの**８列目／９列目**により，競合を起こしているオブジェクトや，フラグメンテーションが進行しているオブジェクトを識別することが可能である.

３．効果

３．１　オブジェクトレベル負荷情報の導出効率向上

旧手法では，オブジェクトレベル負荷情報を導出するために，以下の三つの作業を行う必要があった．

① システム上のデータの流れを把握する．
問題点：情報を収集するためにロギングやトレーシングなどを数週間かけて行う必要があった．

② オブジェクトの用途を確認する．
問題点：数百～数千ものオブジェクトを保有する規模の大きなシステムでは非常に時間を要していた．

③ 導出したオブジェクトレベル負荷情報の正確性と信頼性を高める．
問題点：開発／運用部門の各担当者の協力による確認作業が必要であった．

オブジェクトレベル負荷情報の導出に有効な情報源を持たない旧手法にとり，①～③は断片的ながら情報を確保することのできる重要な作業であった．これに対し，新手法では確たる情報源の確保が行えるため，①～③をすべて省略し週レベルでの大幅な工数短縮を図ることができた．また旧手法が Oracle に関する知識をある程度要求するのに対し，新手法では特に Oracle を意識することなく作業を行えるようになった．

３．２　フラグメンテーション対応の作業効率向上（二次的効果）

新手法はフラグメンテーション対応の作業効率も向上させている．

フラグメンテーション対応は，オブジェクトのフラグメンテーションを解消するために行う作業であり，行退避，行削除，行戻しの三つの手順を踏まなければならない．この三つの手順に要する工数は，オブジェクトの情報量によって大きく異なってくる．

規模の大きなシステムでは，その保有オブジェクトの数が膨大なことから，各オブジェクトに対するフラグメンテーション対応の要／不要を判断しきれず，つい全オブジェクトを対象にフラグメンテーション対応を行ってしまいがちである．この場合，フラグメンテーション対応の必要がなく，かつ情報量の多いオブジェクト（例えば履歴テーブル）なども対象に含まれてしまい，作業の大幅な遅延を招く危険性があった．

新手法では，フラグメンテーション対応の必要があるオブジェクトを即座に識別することができるため，効果的で無駄のないフラグメンテーション対応のプランを立てることができる．ひいてはフラグメンテーション対応時の作業工数を短縮することが可能である．

４．おわりに

４．１　今後の課題と展望

これまでは困難だったオブジェクトレベル負荷情報の導出が，ツールとログマイナーの導入によって効率的に行えるようになり，オブジェクトの分散配置に関する一連の作業を単純化することができた．

当論文で紹介した手法は，解説からも分かるように，変更によって生じた負荷を対象としている．今後は参照によって生じた負荷も対象に含める予定であり，現在以下の二つの案を検証中である．

① ANALYZE による統計情報の収集．
② 監査証跡によるオブジェクトアクセスの監査．

①②の案はいずれも Oracle の機能を活用したものであるが，現在のところ実現の見込みは立っていない．①においては Oracle の動作ルールが変わってしまうことから，②においてはオーバヘッドによる影響が大きいことから，それぞれ適用に踏み切ることができないでいる．

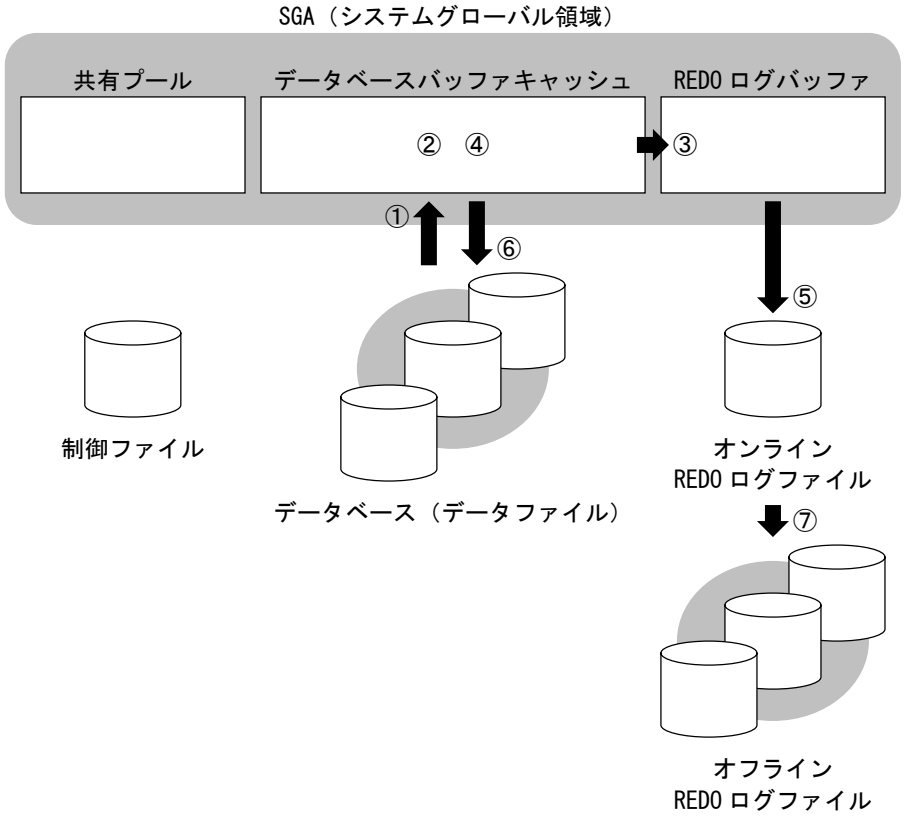
以上の問題を解決し，変更／参照の両面からオブジェクトレベル負荷情報を導出できるようにすることが今後の課題である．また現在当社が推進している一般企業へのビジネス展開を視野に入れ，一連の手法を販売可能なレベルにできるよう，今後も様々な改善／拡張を重ねていく方針である．

付録

付録. 1　ログマイナーの概要

当項では，Oracle の一機能であるログマイナーの概要について解説する．

Oracle は以下の流れに沿って変更を行う．



① 変更対象行が入ったブロック（データブロック）がデータベースバッファキャッシュにない場合，データファイルからデータベースバッファキャッシュにブロックを読み込む．

② 変更対象行をロックする．

③ 変更内容を REDO ログバッファに書き込む．

④ 変更内容をデータベースバッファキャッシュのブロックに書き込む．

⑤ REDO ログバッファの変更内容をオンライン REDO ログファイルに書き込む．

⑥ データベースバッファキャッシュからデータファイルにブロックを書き込む．

⑦ オンライン REDO ログファイルをオフライン REDO ログファイル（アーカイブ REDO ログファイル）として複製する（Oracle が ARCHIVELOG モードである必要がある）．

図５　Oracle における変更の流れ（概略）

図５の流れにおいて，ログマイナーと大きく関係しているのが③⑤⑦である．Oracle は変更が行われると，変更内容をオンライン REDO ログファイルに書き込む（③⑤）．オンライン REDO ログファイルは飽和に達すると，オフライン REDO ログファイルとして複製され，異なる領域に退避～ストックされる（⑦）．オンライン／オフライン REDO ログファイルは Oracle のリカバリに用いられるため，Oracle に対して行われた変更の全内容をカバーしている．

ログマイナーはオンライン／オフライン REDO ログファイルの内容を情報化する機能であり．主に論理障害の原因究明や，論理回復の手段として用いられる．新手法では，オブジェクトレベル負荷情報の情報源を確保するために当機能を用いている．