2N-8

OODBMS ライブラリ“db4o”の高効率化に関する一検討

田島　陽介†　　田胡　和哉‡

†東京工科大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻
‡東京工科大学コンピュータサイエンス学科コンピュータサイエンス学部

1　はじめに

現在，主流の関係データベースシステム（以下 RDBMS）において，インピーダンスミスマッチに対応する作業によるパフォーマンスの低下や，データモデルの表現の限界が指摘されている．それらの解決策の一つに，データベースにオブジェクト指向データベースシステム（以下 OODBMS）を選択する考えがある[2].

そこで本論文では OODBMS に着目し，その中でオープンソースの“db4o”について，データベースの高効率化の一検討を述べる．

“db4o”における問題点の解決のため，効率的なデータベースのガベージコレクション（以下 GC）およびパーティショニングの手法を提案した．現在は，パフォーマンスを重視しつつ，新たな手法の検討をしながら，実装を進めている．

2　OODBMS

OODBMS とは，主にアプリケーションが扱うオブジェクトというデータ形式を，構造も含めて永続化することができるシステムである．

技術的な課題には，データベースの GC，パーティショニング，データアクセス手法などが挙げられる．OODBMS における GC とは，ルート・オブジェクト（以下ルート）から辿ることができないオブジェクトを解放することを指す．

OODBMS に対する技術的な課題に対しては，1990 年代後半から既に研究が行われている[1].

3　db4o

3．1　db4o とは

“db4o”とは，オープンソースの OODBMS であり，容易なコーディングと高いパフォーマンスを併せ持ち，以下の機能が特徴的である．

- 一行保存：あらゆるオブジェクトを，一行のコードを実行するだけで永続化できる．
- ネイティブクエリ：プログラミング言語で，データベースに直接アクセスできる．
- カスケード削除：構造オブジェクトの削除を，一度の命令でまとめて処理できる．

A study on the efficiency of OODBMS library "db4o"
†Yousuke TAJIMA (g211103369@gmail.com)
‡Kazuya TAGO (ktago@gmail.com)
Computer Science, Tokyo University of Technology (†) (‡),
1404-1 Katakura, Hachioji, Tokyo, 192-0982, Japan

3．2　db4o の問題点

“db4o”のカスケード削除機能は，他のオブジェクトが削除対象のオブジェクトを参照しているにもかかわらず，削除してしまうという重大な問題がある．これは“db4o”がデータに対して GC 機構を持たないことを意味する．

また“db4o”では，データベースを単一のファイルで永続化しているため[3]，そのファイルの一部が破損すると，データベース全体に影響を及ぼす危険性がある．

4　提案

4．1　OID 参照関係リスト

上記の“db4o”の問題点を解決する一検討として，OID 参照関係リスト（以下参照リスト）というものを利用するアプローチをとる．

参照リストの概要を図 1 に示す．〇はオブジェクト，〇内の数字は OID を表現している．

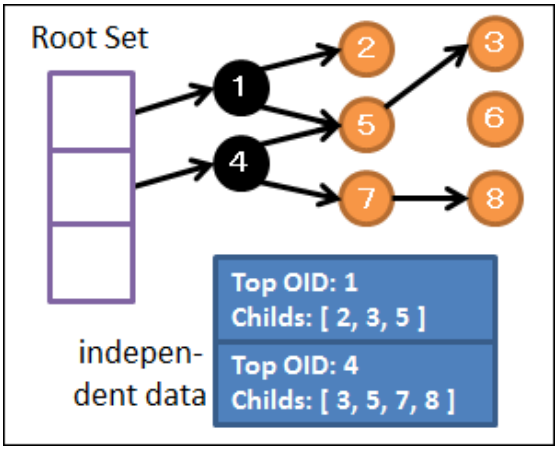

図 1　参照リストの概要

データベースが図 1 の上部の通りに構成されている場合，参照リストは図 1 の下部の通りに構成され，データとして扱われる．

このとき，参照リストにおいて，技術的に考えられる問題点を以下に挙げる．

I. 1 つのルートだけで，データベース内のすべてのオブジェクトを参照できてしまう．
II. データ総数に対して，ルートの数が多すぎる場合，参照リストが大量に作成される．
III. 参照リストそのものが，データベースを占有する存在になる．

データベースが I. の状態では，1 つしか構成されない参照リストは意味がないため，参照リストの分割が必要になる．そこで，以下のオブジェクトを参照リストの Top OID にする基準点とし，参照リストを分割する手法を提案する．

- クロス参照（2 つ以上からの参照，図 1 中での⑤がその例）されているオブジェクト
- ルートから次の階層のオブジェクト群

ここで，ルートから次の階層を基準点とする場合は，クロス参照による基準点が総データ数に対して少なかった場合に限定する．

参照リストの構成時，基準点に到達した場合は，そこから辿るのを止める．そしてその基準点を Top OID として参照リストを構築することで，他の参照リストに対し，基準点以外の重複している OID の追加を減らすことができ，パフォーマンスの向上に繋がることが期待される．

例として，図 1 の状態から，⑤のクロス参照に対して分割処理を行った結果を図 2 に示す．

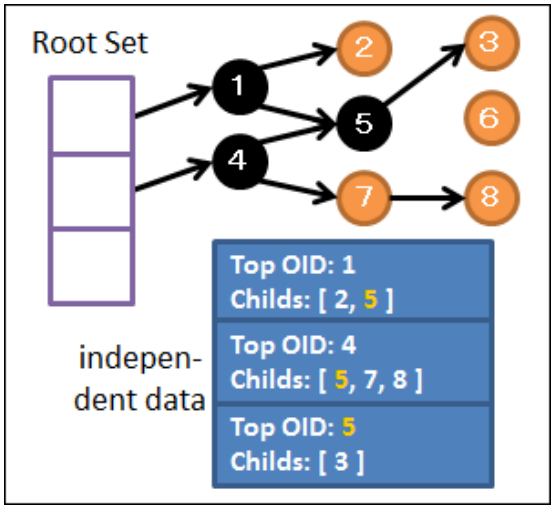

図 2　参照リストの分割例

また，II. の状態では，III. で説明した問題に直結する．その解決策として，ある一定数のルートを 1 つのグループにまとめ，データベースをグループ単位で分割（パーティショニング）する手法を提案する．

技術的には，データベース内のルートに対して，データベースの性質から分割基準となる分割キーを開発者が判断し，データベースを分割する．このとき，分割キーの値は Top OID，リストの追加は“パーティション番号－OID”という形式として，参照リストを構成する．

4．2　参照リストを用いた GC の手法

参照リストを利用した“db4o”での GC のアルゴリズムを，次の手順で提案する．

1. データベースの一定区画内（および全て）のオブジェクトの OID を取得する．
2. その中から各ルートが持つ参照リストの OID と同一のものを次々とリストから除外する．（一定区画内から取り出した場合，別の区画から参照されているオブジェクトも除外する．）
3. 最後までリストに残った OID を持つオブジェクトが不要と判断され，解放される．

5　実装

実装環境には，db4o のバージョンは 8.0，OS は Widows 7，開発言語は Java™ 6，IDE には NetBeans 7.0.1 を用いた．

現在，参照リストおよびそれを用いた GC アルゴリズムの実装は完了している．クロス参照の検出は，参照カウント[4]が 2 以上のオブジェクトを対象に実装しているが，参照カウントによるパフォーマンスの低下を考慮しているため，他の手法を考える必要があるか検討中である．

6　むすび

参照リストを用いた GC は，オブジェクトの階層構造を辿る手間を省けるため，“db4o”に限らず，OODBMS における GC の実行時間を短縮できることが期待されるが，GC の性能に関しては，引き続き実装を進め，インクリメンタル GC[4]の実装の有無によるパフォーマンスの検証実験を行うなど，性能評価を行う必要がある．

また，参照リストの更新に関し，参照関係の変化の有無を検出する手法も，コードレベルでの実装方法を考案する必要がある．

参考文献

[1] Jonathan E. Cook, Alexander L. Wolf, Benjamin G. Zorn, “*A Highly Effective Partition Selection Policy for Object Database Garbage Collection*”, IEEE Transactions on Knowledge and Data Engineering, Jan/Feb, 1998

[2] Java を使ってる？ ならばオブジェクトデータベースだ！， @IT, 2010/06/29, http://www.atmarkit.co.jp/news/201006/29/intersystems.html

[3] 多忙な Java 開発者のための db4o ガイド: 紹介と概要，IBM developerWorks, 2007/03/20, http://www.ibm.com/developerworks/jp/java/library/j-db4o1.html

[4] GC アルゴリズム詳細解説，livedoor Wiki, 2010/03/14, http://wiki.livedoor.jp/author_nari/