部分組み立て法による複合オブジェクト集合に対する問合せ処理

6D－2

安村 義孝
NEC C&C研究所

1 はじめに

関係データベースに代表される従来のデータベースでは対象となるデータ構造が単純であるため、問合せ処理の高速化に関する研究はインデックスや結合演算の高速アルゴリズムに関するものがほとんどであったが、オブジェクト指向データベースではデータ同士が複雑に関連付けられる複合データ構造を持つために、効率的なデータベースアクセスを行うには工夫が必要となる。また、一般にオブジェクト指向データベースの問合せのためにSQLを拡張した言語が用意されており、複合オブジェクト集合に対する様々な検索条件を指定することが可能となっている。そこで本稿では、オブジェクト指向データベースにおけるデータベースアクセス高速化の一方式として、問合せ処理の実行に必要な複合データを構成するデータ属性を、対象となる複合オブジェクト集合から部分的に組み立てていくことにより、データベースへのアクセス回数を削減する手法について述べる。

2 複合オブジェクト集合

オブジェクト指向データベースでは実行時に永続的または一時的にオブジェクト集合を定義したり、任意のオブジェクト間の関連付けなどをすることにより様々な形式の複合オブジェクトを構成することができる。データベースに対する操作としてはこれら複合オブジェクト内におけるオブジェクト間の遷移以外に、複合オブジェクト集合に対する集合演算や問合せが考えられる。

オブジェクト指向データベースの問合せ言語にODMGで規定されているOQL[2]がある。OQLはSQLと同等の問合せ機能以外にもパス式やメソッド呼出しを記述することができ、集合演算を含む複合オブジェクト集合の操作が可能となっている。本稿では問合せ例としてOO7ベンチマーク[3]のQuery 5(一部改)を挙げる。この問合せではアセンブリを構成するコンポーネントより以前に作成されたベースアセンブリを持つモジュールを検索するものであり、OQLで記述すると以下のようになる。

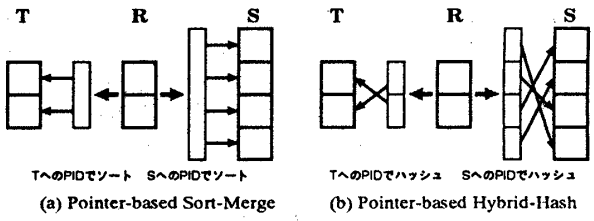
```
select b.module
from   b in BaseAssemblies,
       c in b.componentsPriv
where  c.buildDate > b.buildDate
```

BaseAssemblies はベースアセンブリのエクステント集合である。各ベースアセンブリは componentsPriv というコンポーネント集合を持ち、問合せの条件に双方の属性である buildDate の比較演算が指定されている。この様に、part-of関連を持つ集合同士が入れ子的な問合せではなく、直接条件式中に互いのデータ属性を記述することが可能である。

3 データベースアクセス方式

複合オブジェクト集合を対象としたデータベースアクセスを単純なオブジェクト遷移で行うと、同一のページを何度もアクセスすることがあり、大規模データベースの場合に効率が極めて悪くなる。これを避けるための方式としては、Pointer-based Join[5] と Assembly[4] が提案されており、実際の問合せ処理系の中で使われている事例がある[1]。以下では、提案方式の基礎となるこれらの方式を前節の問合せ例を利用して説明する。ここで、ベースアセンブリ集合を R、コンポーネント集合を S、モジュール集合を T で表す。R と S は1対多、R と T は1対1の関係を持つことになる。

Pointer-based Join はオブジェクト間のポインタを表現するオブジェクト識別子内に物理的なページ番号(PID)を持たせ、そのPIDによるソーティングやハッシングによってアクセスするデータベース内のページを順序付け、オブジェクトを辿る場合のディスクページの読み込み回数を削減している(図1)。この図のように辿る



(a) Pointer-based Sort-Merge　(b) Pointer-based Hybrid-Hash

図1: Pointer-based Join

先が集合(S)でもポインタ(T)でも同じようにアクセス可能である。しかし、Pointer-based Join は各集合が別のストレージに格納されていることが前提とされているため、クラスタリングの効果が得られないという欠点がある。

一方、Assembly は問合せ処理に必要な複合データを一度に取得していくのではなく、ウィンドウと呼ばれる領域にそのサイズ分の複合データを展開していく手法である。スケジューリングによりウィンドウ領域全体に渡ってディスクアクセスを最適化しながら複合データを同時に組み立てていく(図2)。部分集合として存在する S の要素 s はウィンドウ内で分割され、各複合データに対して R の要素 r と T の要素 t がそれぞれコピーされる。1つの複合データが形成されると問合せの条件判定

Query Processing for Complex Object Collections by the Partial Assembly Method
Yoshitaka Yasumura
C&C Research Laboratories, NEC Corporation

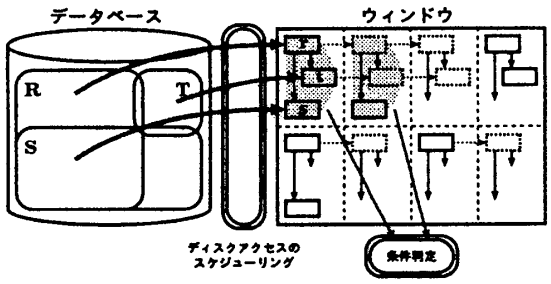


図 2: Assembly

が実行される。ただし、物理的なディスクアクセスのスケジューリング手法を要するので汎用的ではないと言える。また、問合せ処理の実行に必要な複合データが組み立てられた段階で処理が進められるので、本来必要とされないデータも取得してしまう可能性がある。

4 部分組み立て法

本方式ではPointer-based JoinとAssemblyを組み合わせることにより、これら単体で実行した場合の欠点を改善する。Assemblyをベースにしてウィンドウ内に複合データを展開し、オブジェクト間の遷移にはPointer-based Joinを利用する。データの取得はオブジェクトキャッシュ経由とし、ポインタ変換(swizzle)[6]も考慮する。また、展開される複合データには部分集合もそのままの形で扱い、ウィンドウ内に全ての複合データを取得する前の部分的に構成された場合でも、条件判定に必要なデータが揃えば処理を進めるようにする。

この例の場合の処理の流れは次のようになる(図3)。

1. 展開する複合データのルートとなるRの要素rをウィンドウ内に取得する。
2. r内に存在する部分集合Sの要素sへのポインタを検査する。sがオブジェクトキャッシュ上に読み込まれていればウィンドウ内に取得し、そうでなければPIDでハッシュ(ポインタ変換されていない場合)またはアドレスでソート(ポインタ変換されている場合)しておく。
3. ウィンドウ領域が全て埋まったら、sが1つでも展開されている複合データを選び出して条件判定を行う。条件が成立した複合データはそのまま結果集合に格納し、ウィンドウ領域から削除する。条件が不成立でsの残りがない場合にもウィンドウ領域から削除する。空いたウィンドウ領域には次のRの要素を取得する。
4. ハッシュテーブルから1つのバケットを選択し、該当するPIDを持つページをオブジェクトキャッシュに読み込み、2に戻る。ハッシュテーブルが空の場合はソートされたアドレス順にsを取得する。
5. rから関連付けられたTの要素tについても2～4の処理を行うが、条件判定には必要ないのでページの読み込みがある場合は結果集合の要素のみについて行う。
6. 上記の処理をRの要素がなくなるまで続ける。

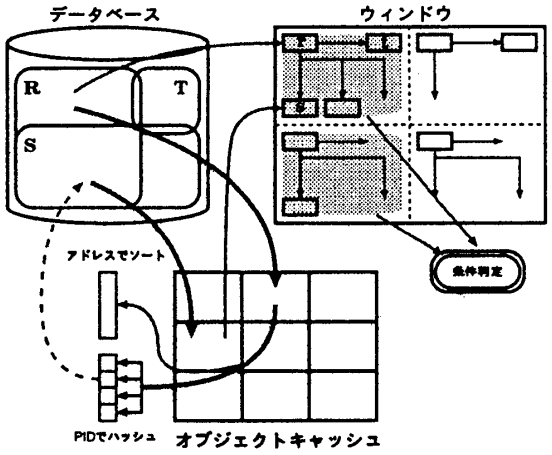


図 3: 部分組み立て法

このように問合せ処理の実行に必要な複合データのルートからデータ属性を順次取得していくが、それらのデータ属性が存在するデータベース内のページを読み込みながら、同一ページに存在する他のデータ属性も同時に取得するという方針を採っている。これにより、クラスタリングの効果が問合せの処理効率に顕著に表れるようになる。取得した複合データは部分的に構成されている場合でも、問合せの条件判定が可能であればその処理を行い、できるだけ冗長なページアクセスを削減している。また、ポインタ変換を考慮することによりオブジェクトキャッシュを有効利用することが可能となる。

5 おわりに

本稿では、複合オブジェクト集合を対象とするデータベースアクセス方式として部分組み立て法を提案した。本方式の特徴は、問合せ処理の実行に必要な複合データの取得を、データベースへのアクセス効率を考慮して部分的に組み立てていくことである。これにより、クラスタリングによる性能向上が見込まれることや、冗長なページアクセスを削減することができるという効果が得られる。今後は、本方式に基づく問合せ処理をオブジェクト指向データベース管理システムPERCIO上に実装し、評価を行っていく予定である。さらには、問合せ最適化との連携も考えていく必要がある。

参考文献

[1] Blakeley, J.A., McKenna, W.J. and Graefe, G., "Experiences Building the Open OODB Query Optimizer," *Proc. ACM SIGMOD*, pp. 287–296, 1993.

[2] Cattell, R.G.G., *The Object Database Standard: ODMG-93, Release 1.1*, Morgan Kaufmann, 1994.

[3] Carey, M.J., DeWitt, D.J. and Naughton, J.F., *The OO7 Benchmark*, CS Tech Report, Univ. of Wisconsin-Madison, 1993.

[4] Keller, T., Graefe, G. and Maier, D., "Efficient Assembly of Complex Objects," *Proc. ACM SIGMOD*, pp. 148–157, 1991.

[5] Shekita, E.J. and Carey, M.J., "A Performance Evaluation of Pointer-Based Joins," *Proc. ACM SIGMOD*, pp. 300–311, 1990.

[6] White, S.J. and DeWitt, D.J., "A Performance Study of Alternative Object Faulting and Pointer Swizzling Strategies," *Proc. VLDB*, pp. 419–431, 1992.