データベース・システム 84−11
（１９９１．７．１８）

OODBMS Odinの問い合わせ処理系

波内 みさ
日本電気株式会社 C&C システム研究所

オブジェクト指向データベース管理システム Odin における問い合わせ処理系の実現方式について述べる。本問い合わせ処理系は、オブジェクト指向の概念を用いて問い合わせ処理を構成する個々の処理をモジュール化することにより、DB 操作関数の選択方法や、問い合わせの最適化手法を容易に拡張出来ることを特徴とする。また、汎用性のある問い合わせの内部表現を導入することにより、複数の問い合わせ言語をサポートすることが出来る。

A Query Processor for the OODBMS Odin

Misa Namiuchi
C&C Systems Research Laboratories
NEC Corporation
4-1-1 Miyazaki, Miyamae-ku, Kawasaki 216, Japan

A query processor is proposed which is now under development for an OODBMS *Odin*. The query processor itself is designed based on the object-oriented paradigm, and the query processing functions are divided into several classes (modules). According to the paradigm, this architecture makes it easier to add new methods for query optimization. In addition, a generic internal expression for queries is provided. Owing to the expression, any query language can be supported by preparing a translator from the language to the internal expression.

1 はじめに

関係データベースに対する問い合わせ処理を効率良く行なうための、問い合わせ変換による最適化手法は、早くから研究が行なわれ[1, 7]、大きな成果を上げている。

これに対して、オブジェクト指向データベースに対する問い合わせ最適化手法は、「問い合わせの構造的類似性により、関係問い合わせに対する最適化技術とほとんど同じである」[2]との主張もあるが、オブジェクトの等価性の取り扱いの問題[5]や、ユーザ定義のメソッドの最適化をどのように行なうか等、多くの問題が残されている。

本稿では、現在我々が開発を行なっているオブジェクト指向データベース管理システム *Odin* における問い合わせ処理系の実現方式について述べる。本問い合わせ処理系は、オブジェクト指向の概念を用いて問い合わせ処理を構成する個々の処理をモジュール化することによって、DB 操作関数の選択方法や、問い合わせの最適化手法を容易に拡張することが出来る。

2 オブジェクト指向 DBMS Odin の概要

現在我々は、次世代データベースの研究・開発の一環として、オブジェクト指向データベース管理システム Odin の開発を行っている[3, 4, 6]。本システムは、現在までに開発されてきたオブジェクト指向データベース管理システム (OODBMS) と比べ、次のような特長を備えている。

1. メソッド SELECT により、C++ から直接に複合条件検索が可能
2. ビューが設定可能
3. 管理システム自体をクラスの集合で実現することによる DBMS 自体の部品化、および、物理構造仮想化機能[6]
4. 可変長 / 長大オブジェクトの管理

本システムでは、DB 操作言語として、C++ にデータベース操作機能を埋め込んだ Odin/C++[3] を提供している。この Odin/C++ で書かれたデータベースへの問い合わせは、まず、プリプロセッサによって C++ 言語にコンパイルされ、さらに C++ 処理系によって実行形式のプログラムに変換された後、実行される (図 1)。

現在、本システムにおける問い合わせ処理高速化のための一アプローチとして、問い合わせ最適化技法を含む問い合わせ処理系についての研究・開発を行っている。以下に、その概要を示す。

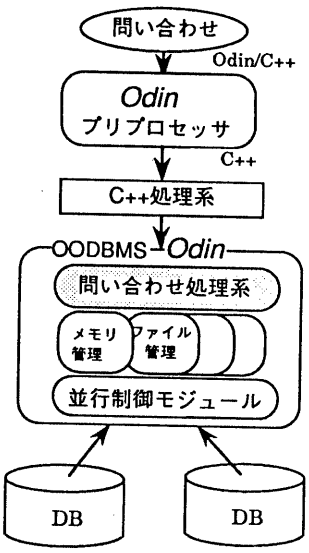

図 1: OODBMS Odin

3 問い合わせ処理系の概要

Odin における問い合わせ処理系は、問い合わせ変換器 (Parser)、問い合わせ最適化器 (Optimizer)、問い合わせ評価器 (Evaluator) からなる。問い合わせ変換器は、与えられた問い合わせを、問い合わせ最適化器において処理対象とする形態に変換する。次に、問い合わせ最適化器では、その時々のデータベースの物理情報と予め与えられた最適化手法をもとに、ユーザからの問い合わせに対する、最も処理効率の良い問い合わせ評価計画 (Query Evaluation Plan: QEP) を生成する。そして、問い合わせ評価器がこの QEP を実行することによって、問い合わせ処理が行われる。Odin の問い合わせ処理系における問い合わせ処理全体の流れを、図 2に示す。

以下では、Odin/C++ で書かれた次のような問い合わせが与えられた場合の処理の様子を例にとって、Odin における問い合わせ処理の説明を行なう。

≪ C++ 言語で書かれた 1000 行より小さいプログラムのうち、共同作業 “Odin 開発” の現在の版番

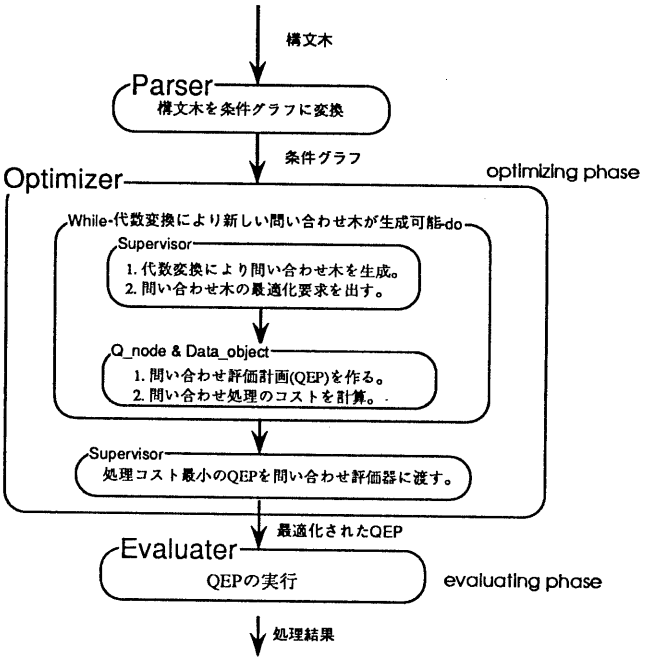

図 2: Odin における問い合わせ処理の流れ

号の前の版番号を持つものを選択せよ。≫

```
$プログラム.select([self: P; $共同作業: C; |
    where
      P.言語 == ''C++'' and P.行数 < 1000
    and
      P.版番号 == C.版番号 - 1
    and
      C.作業名 == ''Odin開発''];);    -- (*)
```

ここで、"$A" は class A のインスタンスを管理する集合オブジェクトを表す。また、クラス "プログラム" のインスタンス変数「言語」および「行数」には、インデックスが付加されているものとする。

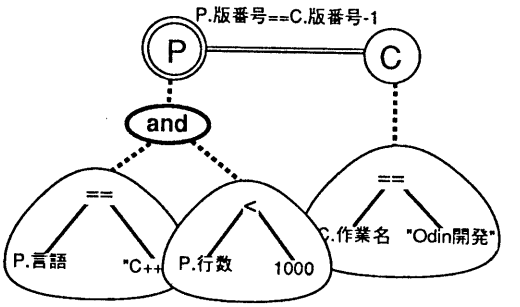

図 3: 条件グラフの例

3.1 問い合わせ変換器 - Parser -

Odin のプリプロセッサは、ユーザからの問い合わせを受け取ると、その検索条件を構文木に変換し、その中に含まれるクラス名などの情報を Odin 内部での表現 (ID など) に置換する。

本問い合わせ変換器は、このプリプロセッサの出力として与えられる検索条件を入力とする。そして、問い合わせの意味を解析するためのグラフ (以下これを、「条件グラフ」とよぶ) を構文木より生成する。

問い合わせ (*) が与えられた時に生成される条件グラフを、図 3に示す。破線が単一クラス内の条件、二重線がクラス間の条件を表す。クラス間およびクラス内の条件が複数存在した場合は、and,or で結合される。最終的に求めるクラス (target class) は、二重丸で表される。

3.2 問い合わせ最適化器 - Optimizer -

本問い合わせ最適化器での処理を概念的に表すと、図4のようになる。まず、QEP を問い合わせから生成する過程全体を管理するモジュール(管理者)がある。これによって、最終的に解として求めるデータに対して、その処理過程をトレースする木が生成される。この時、この木の各ステージで行なわれる処理の種類は決まっているが、実際の処理レベルで用いられる関数・手続きなどは決まっていない。

テキスト、画像などのマルチメディア情報を扱うデータベースでは、それ固有の操作手続きが存在する。また、一つの演算を実現するための処理アルゴリズムが複数存在する場合がある。最適化器は、データベース中のデータの種類やその物理情報(データ数、インスタンス変数に対するインデックスの有無など)、また、検索処理を行なう計算機の性能情報(命令実行速度、I/O 速度など)等をもとに、これらの手続き(DB 操作関数)を決定する。決定方法およびその処理の最適化手法は、これらの情報と共に、データのクラス毎に用意されるモジュール内で管理されている。そのステージでの処理内容が決定されると、それは木の各ノードに記憶される。

木を構成する各ノードでは、この処理と同時に、そのステージとそれ以前の処理コストが予測計算される。最上位のまとめ役のモジュールは、いくつかの QEP を生成し、それらのコストを比較することによって、コストが最小の QEP を選択する。

実行フェイズでは、この木を図の下から順に実行することによって、問い合わせ処理が行なわれる。

本最適化器では、上述の処理を、"Supervisor"、"Q_node"、"Data_object" の3つのクラス・オブジェクト(モジュール)を使って実行する(図 2)。Supervisor は、問い合わせの最適化および QEP 生成処理全体を管理する。Supervisor が生成する QEP 生成のための問い合わせ木は、Q_node による木で表される。QEP の生成において利用される DB 中のデータの物理情報および統計情報は、各データ・クラスに対応した Data_object で管理されている。

以下に、これらのモジュールの機能の詳細を示す。

3.2.1 Supervisor

Supervisor ではまず、条件グラフに大まかな処理順序を与え、その結果に代数変換を施すことによって、複数個の問い合わせ木の候補を生成する。この問い合わせ木は、後述の Q_node によって構成されている。この Q_node による問い合わせ木の例を、図5に示す。この時の問い合わせ木は、まだ下位の DB 操作関数が決定されてない。

この代数変換ルールは、Supervisor オブジェクトのメソッドとして定義される。したがって、ルールの追加、削除、変更は、メソッドに対する操作とその施行方法の記述変更だけでよい。

問い合わせ木を生成した後、Supervisor はこれに対して、問い合わせ処理の各部分処理レベルでの最適化および DB 操作関数決定を要求するメッセージを送る。それぞれの Q_node での処理の結果、その問い合わせ木(QEP)による処理のコストが返されるので、Supervisor は、この予測コストの値を基に、最終結果の QEP を選択する。

変数名	型	備考
parent	Q_node*	問い合わせ木における親ノード
child	Q_node*	問い合わせ木における子ノード
param	Q_node*	自ノードに対する演算のパラメータ
op_name	char*	そのノードでの演算名
lof_name	char*	最適化によって決定される論理操作関数名
cost	int	そのノード以下の subtree の全コスト
cond	OD_CTREE*	そのノードでの演算に対する条件
entity	Data_object*	演算を施した中間結果の集合オブジェクト
param_data	Data_object*	演算の対象であるデータ・オブジェクト

表 1: Q_node に格納される情報

3.2.2 Q_node

本方式では、複数の Q_node によって問い合わせ木を表現する(図 5)。個々の Q_node は問い合わせ木の各ノードに対応し、各ノードが問い合わせ処理を構成する一連の処理の一つ一つを表現する。そして、処理の順序に従ってそれらを関連付けることにより、問い合わせ木を構成している。

各 Q_node 内では、表1のような情報を管理している。ここで、child は自分より先に演算を行なう子ノードの Q_node、parent は自分より後に演算を行なう親ノードの Q_node を示す。param は演算のパラメータを計算するための Q_node、op_name はそのノードで行なわれる処理、そして cond はそのノードでの演算の条件をそれぞれ表す。Data_object によって DB 操作関数が決定されると、その関数名を lof_name に、演算の結果生成される中間データの情報を entity に、処理コストを cost にそれぞれ格納する。この Q_node の構造を図6に示す。

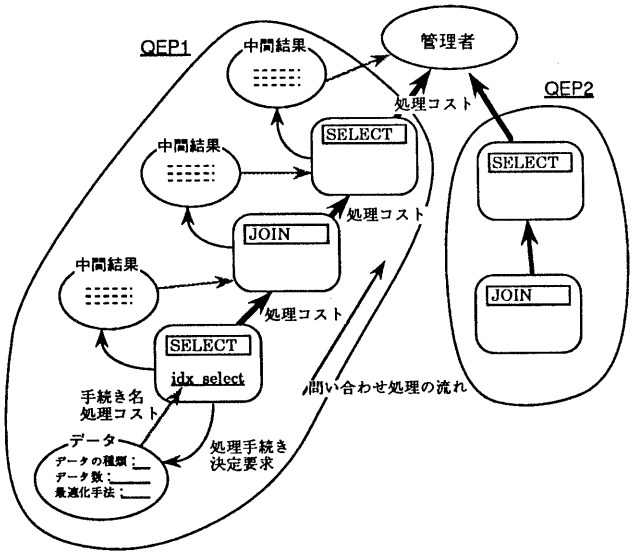

図 4: 問い合わせ最適化器の処理

Q_node ではまず、その親ノードから処理要求のメッセージを受ける。そして、そのノードが子ノードを持つ場合には、それに対して処理要求のメッセージを送り、その結果として処理コストの予測値を受けとる。さらに、その演算のパラメータとなるノードが存在する場合には、それに対して処理要求のメッセージを送り、予測処理コストを受けとる。

各 Q_node では、そのノードで行なう演算に対する DB 操作関数の決定を、対応する Data_object に求める。Data_object から演算のパラメータとなるデータの物理／統計情報を要求された場合には、param で管理されているパラメータ・ノードからその値を得、それを渡す。Data_object から得られた関数名、処理コストは、表 1に示す項目に格納する。

Q_node ではさらに、中間結果として新しく生じるオブジェクトのデータ数、選択率などを予測し、中間データ格納用のインスタンス変数 entity に設定する。親ノードでは、この中間結果の情報を使って、最適化を行なう。

Q_node は、Data_object とのメッセージ交換の結果、与えられた演算を複数の演算に分割して処理した方が処理コストが小さい可能性があることが判明した場合には、それらの演算に対応する Q_node を生成して問い合わせ木を拡張する。反対に、問い合わせ中で分割して書かれていた処理の統合も行なう。

処理の分割を、図 5の問い合わせ木に対して、Supervisor から最適化要求のメッセージが送られてきた場合を例に考える。この様子を図 7に示す。

クラス“プログラム”のインスタンス変数「言語」および「行数」にはインデックスが付けられているので、左の枝の P(クラス“プログラム”) に対する検索条件の処理方法としては、「言語」「行数」それぞれに対する条件を別々にインデックスを利用して検索し、その結果の共通項を求める、という手法が考えられる。この P に対する検索を行なう Q_node は、二つのインスタンス変数に関するインデックスの有無の情報を、P に対応する Data_object から得、処理方法の一つの可能性として図 7の左下のような木を生成する。そして、その場合の処理コストを計算し、一度に逐次的に条件検索を行なう場合 (図 7右上) の処理コストと比較して、コストの小さい方を結果の QEP とする。

3.2.3 Data_object

Data_object は、問い合わせの中で使われるデータのクラスに対応して生成される。ここでは、各ク

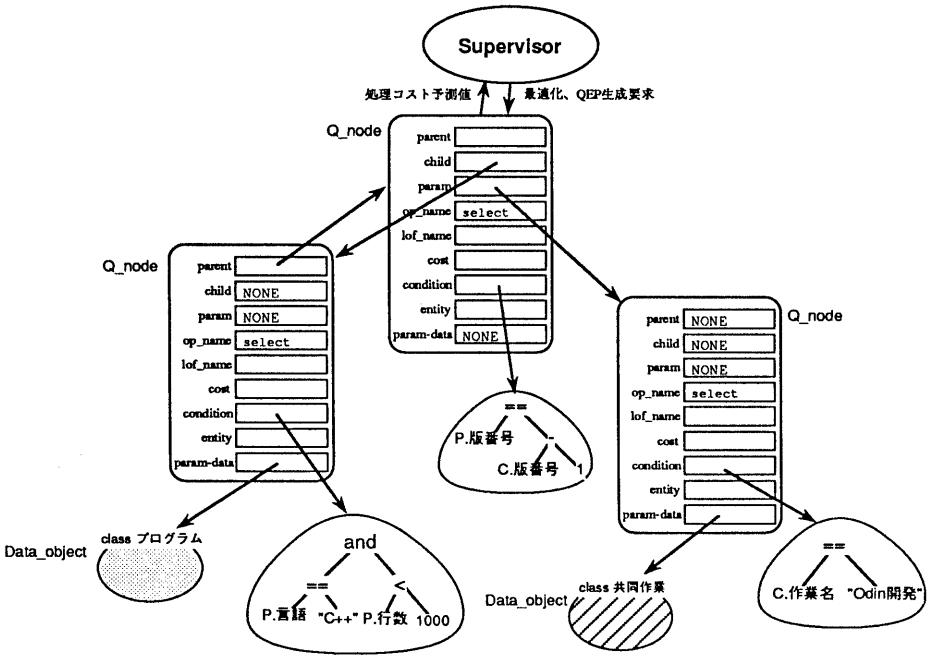

図 5: Q_node による問い合わせ木

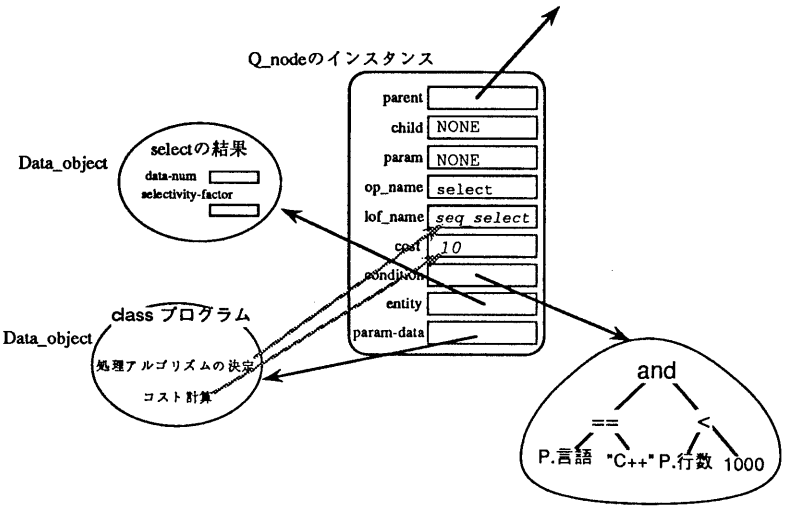

図 6: Q_node の概要

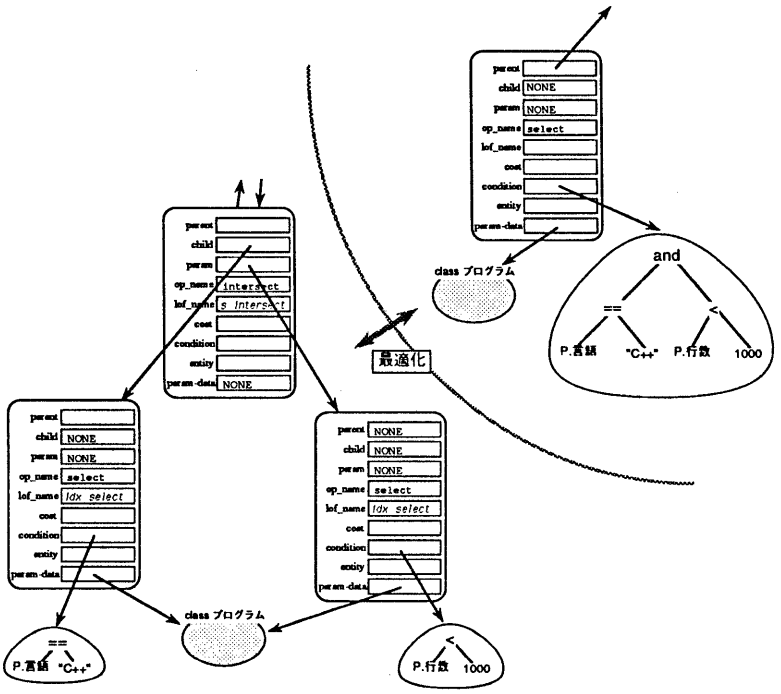

図 7: Q_node の分割

ラスに対して次の情報を管理する。

- そのクラスのインスタンス数
- インデックスを有するインスタンス変数のリスト
- 各インスタンス変数の選択率 (selectivity factor)
- そのクラスに定義されたメソッドの最適化手法

各 Data_object では、その物理／統計情報を求めるメッセージを受けとると、その値を返す。また、最適化および DB 操作関数決定要求のメッセージを受けとると、そのデータの物理情報や、自分のメソッドとして管理している固有の最適化手法より DB 操作関数を決定する。これは、処理コストの計算を行い、最もコストの小さいもの求めることによって行なう。決定したデータベース操作関数、および処理コストは、対応する Q_node に返す。

あるクラスに対応する Data_object に定義された最適化手法は、そのクラスを継承するサブクラスにも継承され、適用することが出来る。

複数のインスタンス変数に対する検索条件が、一つあるいは複数の Q_node から与えられた時には、それらのインスタンス変数におけるインデックスの有無を調べ、Q_node における処理の分割あるいは統合を行なうための情報を提供する。

3.3 問い合わせ評価器

問い合わせ評価器では、問い合わせ最適化器において生成された QEP をもとに、Odin の DB 操作関数を実行する。

与えられた問い合わせに対する QEP は、Q_node による木によってその操作列が与えられている。評価器は、これを順に実行していくことによって QEP を評価し、問い合わせの解を得る。

4 まとめ

本稿では、オブジェクト指向 DBMS Odin における問い合わせ処理系の実現方式について述べた。

本方式は、オブジェクト指向パラダイムの枠組の中で問い合わせ処理系を実現することにより、問い合わせ処理全体を構成する個々の処理を部品化した。これにより、ユーザ定義のメソッドに関しても、そのメソッド特有の最適化方式を Data_object に追加することによって、最適化を行なうことが出来る。

また、問い合わせの中間表現 (条件グラフ) を導

入したことにより、他の言語による問い合わせを処理に対する場合でも、その言語に対するプリプロセッサと条件グラフへの変換手続きを用意することによって、対応することが出来る。例えば、SQL で書かれた問い合わせに対して、それを構文解析するプリプロセッサを用意し、構文木を条件グラフに変換する手続きを Parser に組み込むことによって、それも実行することが出来る。

現在、本問い合わせ処理系の機能を制限した版を、C++ を使って試作中である。これをベースとして、高度な最適化手法を組み込んだ処理系を完成させ、評価・検討を行なう予定である。

謝辞

本研究に関して、貴重な御意見を戴いた日本電気(株)C&C システム研究所鶴岡邦敏、木村裕両氏に深謝いたします。また、有益な助言を与えて下さった同社情報処理システム技術本部川西淳氏に感謝いたします。

参考文献

[1] M. Jarke and J.Koch, "Query Optimization in Database Systems," Computing surveys, Vol. 16, No. 2, pp.111-152, June 1984.

[2] W. Kim, "Introduction to Object-Oriented Databases," MIT Press, 1990.

[3] 木村、鶴岡、「オブジェクト指向データベース操作言語 Odin/C++ について」、電子情報通信学会データ工学研究会、DE90-26、pp.17-24、1990.

[4] Y. Kimura and K. Tsuruoka, "A View Class Mechanism for Object-Oriented Database Systems," Proc. of 2nd Conf. on DASFAA'91, pp. 269-273, April 1991.

[5] G.M. Shaw and S.B. Zdonik, "Object-Oriented Queries: Equivalence and Optimization," Proc. of Conf. on DOOD'89, pp. 264-278, 1989.

[6] 鶴岡、木村、「オブジェクト指向データベース管理システムの物理構造仮想化方式」、情報処理学会データベースシステム研究会、DBS73-8、1989.

[7] J.D. Ullman, "Principles of Database and Knowledge-Base Systems," Volume II, Computer Science Press, 1989.