データベースシステム 116－4
（1998. 7. 8）

画像検索アプリケーションにおける問い合わせ機構

岡田 敏　鬼塚 真

NTT情報通信研究所

E-mail:{satoshi, onizuka}@dq.isl.ntt.co.jp

概要

近年、大量のディジタル画像を扱う機会が増加しており、これらの画像データを効率的に扱うため、画像データを格納するデータベースの利用が不可欠となっている。画像データベースに対する問合せでは、検索キーに指定された画像に対して、類似した部分を含む画像を検索する複雑な検索が要求される。この要求を実現するため、画像を細かい部分画像に分割し、画像と部分画像との間に親子関係を持たせてデータベースに格納する方法が必要である。

我々が研究を進めている拡張可能DBMSフレームワーク上で実現した、画像検索用ＤＢＭＳでは、画像検索アプリケーションに適した問合せ機構を実現している。まず、親子関係を持ったデータへの検索を実施した場合には親子関係を保ったまま検索結果をアプリケーションに返却する機能を提供しており、アプリケーション側の利便性を高めている。次に、類似検索時に使用するメソッドの実行に関しては一度実行したメソッドの結果をDBMS内に保存し、他のメソッド実行時に この結果を利用することにより、メソッドの実行回数を削減し、処理の高速化を実現している。そして、画像データおよび、データモデルの特性、画像データ検索で実行するメソッドの処理コストを解析し、画像データに適した最適化機構を実現している。

Query Processing Mechanism for Image Retrieval Application

Satoshi OKADA　Makoto ONIZUKA

NTT Information and Communication Systems Laboratories

abstract

As digital cameras have been becoming popular, database systems for still image data are becoming important. It is one of the promising technique for content based retrieval to extract sub-images from each image data, store them into database system, and search some still images in the database according to the similarity of sub-images to the key image.

In this paper, we propose query processing mechanism for above image retrieval application with three parts, and explain how we implement it on extensible DBMS framework. First, we extend query model for structured data processing to reduce the size of query result. Second, we propose method materialization during the query execution which reduce the cost of query . Third, we argue some query optimization issues for structured data and show some examples to prove it.

1 はじめに

近年のディジタルカメラやインターネットの急速な普及により、大量のディジタル画像を扱う機会が増えている。これら大量の画像を利用するアプリケーションでは効率良く画像データを管理するためデータベースの利用が不可欠となっている [1]。これらのアプリケーションでは画像を検索キーとし、「この画像と似た部分を含む画像が欲しい」といった複雑な検索が要求される。

"含む画像の検索"を実現するためには、画像から構成部品を抽出し、元の画像と部分画像を別々の管理対象としてデータベース内に格納する。そして、元画像と部分画像との間に包含関係を示す親子関係を持たせる方法が必要である。検索時には、子画像である部分画像と検索キーとの比較を実施し、検索条件に適合する子画像の親画像を検索することにより"含む画像の検索"を実現する。

また、"似た画像を検索"は類似検索を示しており、様々な類似検索用のアクセスメソッドが提案されている。DBMS はこのアクセスメソッドを提供し、SQL 等で示される問合せを高速に実行する必要がある。

親子関係のような構造を持った画像データへ検索要求を発行する際、RDBMS、OODBMS、OR-DBMS で代表される従来の DBMS では以下に示す、三点の問題がある。

- 構造を持ったデータにアクセスした場合、得られる結果には親子関係の構造が反映されない。そのため、どの部分画像にヒットしたのか、また、1 画像に何件の類似した部分画像が含まれているのか等のアプリケーション側での解析処理が複雑になる。
- 類似検索を実行する際に、検索キーとなる画像とデータベース内のデータがどの程度似ているかを示す、類似度の計算など、処理コストの高いメソッドの実行が必要となり、処理速度が劣化する。
- 画像データ用に特化された DBMS ではないため、SQL 等の問合せ処理の実行順序を決定するときに、類似検索用のユーザ定義インデックスの処理コストを反映できない等、問い合わせ処理の最適化が十分でない。

本稿では、まず 2 章で画像検索アプリケーションの概要ついて述べ、DBMS に要求される機能を明らかにする。続く各章では、上で述べた個々の問題について詳述し、画像検索エンジン *HyperMatch*[3] を我々が研究を進めている DBMS の拡張可能フレームワーク [2] を用いて設計・実装した DBMS 上で実現している解決方法について述べる。

2 画像検索アプリケーション

2.1 データモデル

画像データベースに対する問い合わせ要求を効率良く実現するデータモデルを図 1 の例に従って説明する。

- 画像と画像から抽出した部分画像を別々のクラスに格納する。例では Image クラスに抽出元の画像を格納し、SubImage に部分画像を格納する。以下の説明では Image クラスの画像を親画像、SubImage クラスの部分画像を子画像とする。
- 親画像と子画像の間には親子関係の構造を付与する。例では Image クラスの children 属性が子を参照し、SubImage クラスの parent 属性が親を参照している。この参照関係により親子関係を形成している。

2.2 画像データベースへの問い合わせ要求

画像検索アプリケーションでは、

「1998 年に撮った、りんごとオレンジが写っている写真が検索したい」

というような検索要求が DBMS に対して発行される。この検索要求を分析すると以下に示す検索方法に分類できる。

- **書誌情報検索**：各画像に付けた書誌情報 (作者、製作日、タイトル、キーワード等) をキーに検索する。
- **画像検索**：キーとして画像を使用して検索を行う。

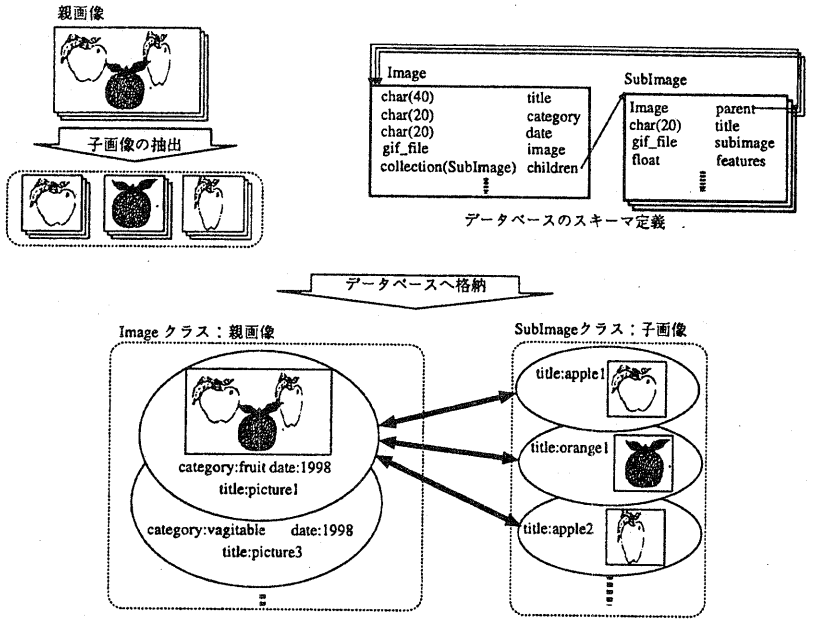

図 1: 画像データモデル

画像検索はさらに、以下のような検索方法を含んでいる。

- **類似検索**: キー画像と似ている画像を検索する。
- **部分画像検索**: 1 枚の画像を構成する部分画像 (例えば、食卓の写真では、その中の皿、パンなどの被写体) を検索の対象とする。

上で示した検索例では

- **1998 年撮影の写真**: 書誌情報検索
- **りんごとオレンジの写っている写真**: 部分画像による類似検索

となっており、分類した検索方法を組み合わせた検索要求が DBMS に発行されるのが一般的である。

3 検索結果の構造化

図 1 で示したように画像データは、親画像と子画像の間に親子関係という構造を持っている。本章では、このような構造を持ったデータを保持しているデータベースへ検索要求を発行したときに生じる従来技術の問題点、および我々が採用した解決法について述べる。

3.1 従来技術の問題点

従来の入れ子リレーショナルモデル [4], OODBMS における問合せモデル [5] では、返却される検索結果を構造化できないという問題があった。

検索例 1 を使用して問題点を明らかにする。なお、以下の説明で用いる SQL は ODMG2.0 に準拠した形式で記述する。

- 検索例 1

```
select x.gif_file, y.gif_file
from   Image x, x.children y
where  y.features similar() 'AppleKey.gif';
```

本例はキーとして画像'AppleKey.gif' が指定されたとき、AppleKey.gif と子画像の特徴量を示す'features' との比較をユーザ定義オペレータ'similar()' により実行する。AppleKey.gif と似ていると評価された子画像：y.gif_file とその画像の親画像：x.gif_file をアプリケーションへ返却するものである。

従来の DBMS に対して、検索例 1 の検索要求を発行した場合、検索結果は図 2 で示される形式で返

却される。

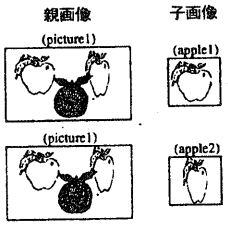

図 2: 検索例 1 の実行結果

データベース内では apple1 画像と apple2 画像は picture1 画像の子画像であるという構造を持っていた。しかし、検索結果では"picture1 画像 - apple1 画像", "picure1 画像 - apple2 画像"というように独立な関係に展開されている。

このように、従来の DBMS ではデータベース内で構築していたデータ間の構造関係を検索結果に反映してアプリケーションへ返却することができない。そのため、親画像として何件ヒットしたのか、また、一つの親画像に対して類似した子画像が何件含まれているのか等の情報を検索結果から直接得ることができない。

これらの情報を得るためには、検索結果をアプリケーション側で親画像についてソートし、同一の親画像で集約するなどの処理が必要である。多くのクラス間で参照されているような場合では、展開された検索結果からデータベース内の参照関係を再構築することはソート、集約処理が多発し、アプリケーション側での処理コストが高くなってしまうという問題があった。

3.2 検索結果の構造化

HyperMatch では、構造化されたクラスへの検索について、検索結果も構造化してアプリケーションへ返却する。

検索例 1 は以下のような順序で実行される。実行イメージと検索結果の例を図 3 に示す。

1. where 句の実行により、各クラスについて検索条件に合致するインスタンス集合を得る。
2. where 句の実行で求められた親画像のインスタンスについて、子画像への参照を where 句の実行で求められた子画像のみとし、データベース中にあった親子関係を再構成する。
3. 構造化されたインスタンス集合からアプリケーションが要求するデータを構造を持つ形式で返却する。

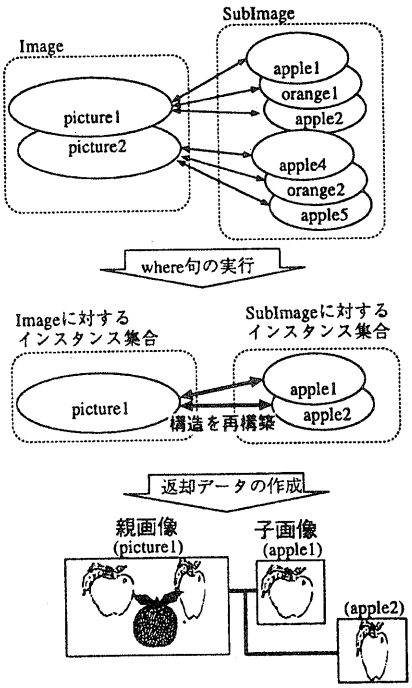

図 3: 検索結果の構造化

アプリケーションは構造化された結果を受け取るため、結果として得られる画像のどの部分 (どの子画像) が検索対象となったかが明確となり、アプリケーション側での処理コストが削減できる。また、親画像に対する結果が重複して返却されないため、転送コストが軽減され、検索処理の高速化が実現できる。

4 メソッドの実行

画像の類似検索ではキー画像とデータベース内の画像がどの程度似ているかを計算するユーザ定義メソッドが多用される。ユーザ定義メソッドは画像の色、形状など多くの要素を用いて計算する。ここではユーザ定義メソッド実行時に生じる従来技術

の問題点および、我々が採用した解決法について述べる。

4.1 従来技術の問題点

問題点を明らかにするため、例として検索例2を以下に示す。

- 検索例2

```
select x.gif_file,
x.similarity(AppleKey.gif)
from SubImage x
where x.similarity(AppleKey.gif) >= 0.8;
```

本例はキーとして画像'AppleKey.gif'が指定され、SubImageクラスの各インスタンスとキーとの類似度をユーザ定義メソッド'similarity()' の実行により求める。求められた類似度が0.8以上のインスタンスについて、'子画像：x.gif_file' と'類似度：similarity() の実行結果'をアプリケーションに返却するものである。

検索例2を実行する場合、検索の高速化を実現するため、SubImageクラスには画像検索用のインデックスを付与し、このインデックスを利用して検索が実施される。この時の処理の流れは以下のようになる。

1. where句を実行する。SubImageに付与されているインデックスを利用して類似度を求め、検索条件に合致するインスタンス集合を得る。
2. select句の実行。where句の実行で求めたインスタンス集合の個々に対してメソッドを実行することにより類似度を求め、アプリケーションに返却するデータを得る。

このように、インデックスを利用した類似検索では、インデックスをたどる処理により、類似度の計算がなされいる。しかし、その後のselect句の実行でもwhere句と同様のメソッドを実行しなくてはならないため、処理速度の劣化が生じる。

メソッドの実行に関しては検索結果を利用しないメソッドについては、あらかじめメソッドを実行しておき、その結果を格納しておくことにより、検索処理時のメソッドの実行を避ける方法(メソッドマテリアライゼーション)が提唱されている[6]。

しかし、本例のように、メソッドの処理結果が検索処理実行時にしか得られないような検索要求では、あらかじめメソッドの処理結果を保存しておくことは困難である。

4.2 実行時のメソッドマテリアライゼーション

*HyperMatch*では図4に示すように、メソッドが定義されているクラスの各インスタンスに一時的な属性(テンポラル属性)を付与する。そして、類似検索実行時にインデックス経由で求められるメソッドの結果をテンポラル属性領域に保存することにより、実行時のメソッドマテリアライゼーションに対応する。

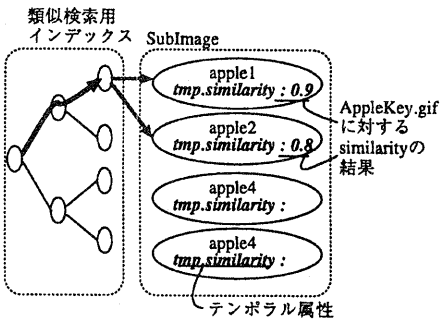

図4: テンポラル属性の利用

テンポラル属性を使用し、検索例2を実行する処理の流れを以下に示す。

1. メソッドの実行結果の再利用が可能かチェックする。具体的には、where句のメソッドとselect句のメソッドが同一メソッドであり、パラメータの一致など、同一条件で実行されるものかどうかチェックする。
2. where句の実行。インデックスを利用し、検索条件に合致するインスタンスを特定する。上記チェックに適合する場合にはインデックスを利用したデータアクセス時に計算される類似度をテンポラル属性に保存する。
3. select句の実行。上記チェックに適合している場合にはテンポラル属性に保存されている類似

度を使用することにより、select 句に指定されたメソッドの実行結果とする。

テンポラル属性を使用することにより、メソッドの実行回数を削減することが可能となり、問合せ処理の高速化が実現できる。

5 問い合わせ処理の最適化

画像データベースでは2章で示したように、画像に対して複数の子画像が対応する形式でデータが格納されている。そして、親画像と子画像間の参照関係を利用した検索、あるいは従来の書誌情報検索よりも多くの処理コストを要する類似検索など、画像データベースに特徴的な処理が要求される。

アプリケーションから発行された検索要求は、このような画像データベースに特有の検索方法に適合する最適化処理を施し、検索を実行しなくてはならない。

5.1 従来技術の問題点

以下に示す検索例を使用して、従来技術の問題点を明らかにする。

- 検索例3

```
select x.gif_file, y.gif_file, z.gif_file
from Image x, x.children y, x.children z
where y.features similar() 'AppleKey.gif'
and z.features similar() 'OrangeKey.gif';
```

'AppleKey.gif' と'OrangeKey.gif' がキーとして指定されている。SubImage クラスの各インスタンスとキーとの類似検索をオペレータ similar() を使用して実行し、2個のキー画像に似ている親画像：x.gif_file と'AppleKey.gif に類似している子画像：y.gif_file そして、'OrangeKey.gif に類似している子画像：z.gif_file を返却するものである。

本 SQL を実行する場合、図5に示すように、2種の実行順序がある。

- マージ法
  個々の検索条件に合致するインスタンスを求め、その結果をマージし、最終的な検索結果とする方法である。
  1. 'AppleKey.gif' との類似検索
     (a) SubImage について、'AppleKey.gif' との類似検索を実行し、似ている子画像を絞り込む。
     (b) 上の (a) の処理で絞り込まれた子画像について、Image と SubImage 間の参照関係を利用して、各子画像から親画像を絞り込み (トラバース処理)、親画像に対するインスタンス群を得る。
  2. 'OrangeKey.gif' との類似検索
     AppleKey.gif との類似検索と同様に子画像が OrangeKey.gif に似ている親画像のインスタンス群を得る。
  3. 各インスタンス群のマージ
     各類似検索の結果、得られたインスタンス群を統合 (ここでは AND) し、where 句の実行結果とする。

- 入れ子法
  検索条件を順番に実行することにより、検索範囲を絞り込み、目的とするインスタンスを得る方法である。
  1. 'AppleKey.gif との類似検索
     一方の検索条件 (ここでは AppleKey.gif との類似検索とする) について検索を実行し、似ている子画像を絞り込む。
  2. 親画像の絞り込み
     AppleKey.gif との類似検索で絞り込まれた子画像について、親画像へトラバースし、絞り込まれた親画像のインスタンス群を得る。
  3. 'OrangeKey.gif' との類似検索
     上の 2 の処理で絞り込まれた親画像からトラバース可能な子画像について OrangeKey.gif' との類似検索を実施する。この類似検索によって絞り込まれたインスタンス群から検索結果を生成する。

この2種類の検索方法のどちらを使用するか決定する場合には、検索にインデックスが使用できるか、また、インデックスを使用する場合とインデックスを使用しない場合の処理コストの比較が必要である。

また、格納しているデータの特性、そして、検索を高速化するためにユーザが作成したユーザ定義イ

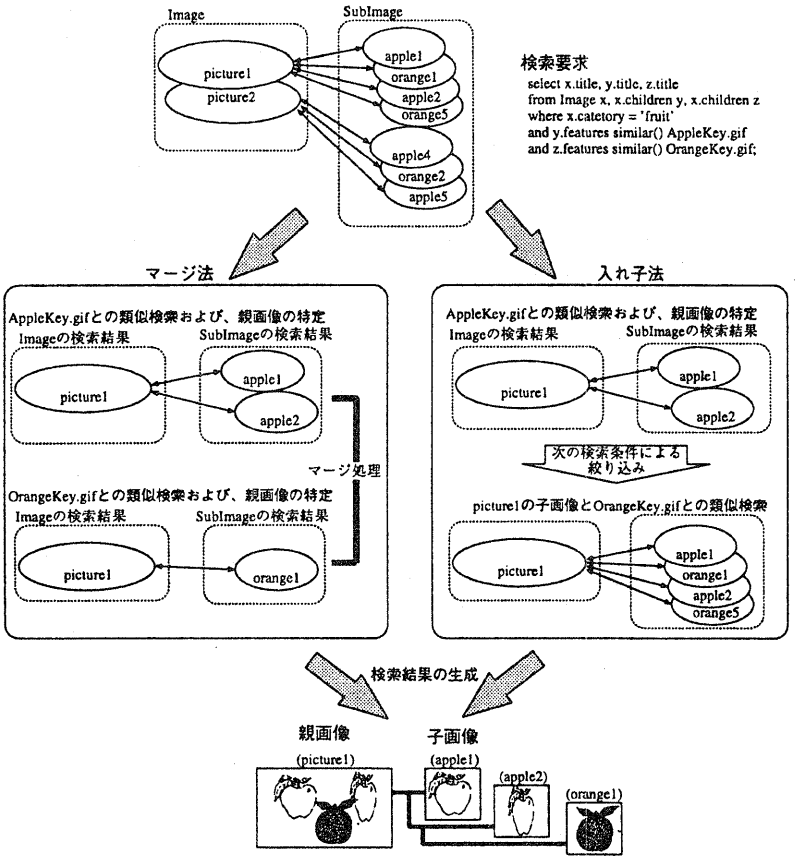

図 5: SQL の実行順序

ンデックスの処理コストなどを事前に見積り、ＳＱＬの最適な処理順序を決定しなくてはならない。従来の画像データベースでの最適化[7]では、入れ子法のみであり、トラバース処理が考慮されていないなど、不十分なものであった。

5.2 検索順序の最適化

画像検索では画像という、扱うメディアの特性を把握することにより、データへの最適なアクセス順序を決定する必要がある。

前節で述べた処理手順の中からコストのかかる処理をまとめると表１のようになる。ここで、類似検索については検索処理を高速化するために画像検索用のインデックスがSubImageクラスに付与されているものとする。

トラバース処理については、各インスタンスの参照関係により、ポインタをたどる操作のみであるため、処理コストは小さい。また、マージ処理についても、個々の類似検索によって絞り込まれた親画像に対するマージ処理であるため、マージする対象数が少なく処理コストは小さい。

類似検索については、インデックスの探索、または個々のインスタンスについて類似度を計算する処理が必要であるため、トラバース処理、マージ処理よりも処理コストは大きい。

入れ子法では、一方の類似検索にインデックスが使用できず、最初のキーによる類似検索の結果として、絞り込まれた親画像からトラバース可能な全子画像について、類似度の計算が必要となる。

表 1: 最適化プランで考慮すべき処理

処理	マージ法	入れ子法
AppleKey 類似検索	インデックス経由の検索 コスト：中	インデックス経由の検索 コスト：中
OrangeKey 類似検索	インデックス経由の検索 コスト：中	インデックスを経由しない検索 コスト：大
トラバース処理	子から親へが 2 回 コスト：小	子から親へ 親から子へ コスト：小
その他	マージ処理 コスト：小	

HyperMatch で管理している画像データでは、1 親画像につき約 100 件の子画像を保持しているため、インデックスを経由しない類似検索は処理コストが非常に大きくなる。

HyperMatch では、検索例 3 に対して、類似検索にインデックスが利用でき、類似検索の処理コストを抑えることが可能なマージ法を採用している。

以上のように、画像検索における最適化ではクラス間のトラバース処理、1 対多のトラバースの場合には "多" 側の数、類似検索における処理コスト等を見積もることが必要となる。

6 まとめ

本稿では、従来のデータベースに対し、画像検索アプリケーションを適用した場合の問題点を 3 点示した。そして、その問題点に対する解決方法を提案し、*HyperMatch* での実装方法を示した。その方法とは、

- 検索結果の構造化
- 実行時のメソッドマテリアライゼーション
- 画像検索に適した検索の最適化

である。

以上の実現により、画像検索アプリケーションが求める検索機能、および、検索の高速化が実現できた。

今後は、さらに、画像検索アプリケーション検索要求を解析し、DBMS の機能に反映していく予定である。

参考文献

[1] 赤間 浩樹, 紺谷 精一, 三井 一能, 串間 和彦, "画像内オブジェクトの自動抽出を使った画像検索システム - ExSight -," 信学会, 第 8 回データ工学ワークショップ,1997.

[2] 岡田 他 , "高速 ORDBMS LiteObject の設計と実装", "信学会, 第 9 回データ工学ワークショップ, 1998.

[3] K.Curtis, N.Taniguchi, J.Nakagawa and M.Yamamuro, "A Comprehensive Image Similarity Retrieval System that utilizes Multiple Feature Vectors in High Dimensional Space," 情処学会 DBS-113, 1997.

[4] Marc Gyssens and Dirk Van Gucht, "The Powerset Algebra as a Result of Adding Programming Constructs to the Nested Relational Algebra," ACM SIGMOD Conference 1988, pp.225-232.

[5] Won Kim, "A Model of Queries for Object-Oriented Databases," IEEE Conference on Very Large Data Bases, 1989, pp.423-432 .

[6] A.kemper, C.Kilger, and G.Moerkotte. Function Materialization in Object Base: Design, Realization, and Evaluation. IEEE Transactions on Knowledge and Data Engineering, August 1994, pp.587-608.

[7] Surajit Chaudhuri and Luis Gravano "Optimizing Queries over Multimedia Repositories," ACM SIGMOD, 1996,pp.91-102.