6X - 3　データベースラッピングにおける問合せ処理の実現手法

古舘 丈裕　　小室 睦
日立ソフトウェアエンジニアリング（株）

1. はじめに

早くから汎用機上に構築されてきたデータベースの中には、現在でも有益な情報資産として利用されているものも少なくない。最近ではこれらのDBを統合し、PCやモバイル機器からも利用したいというニーズが高まってきている。この時、汎用機上のデータを利用するアプリケーションが現役の場合は、それらのアプリケーションも含めて完全に移行するか、データの定期的な転送・変換で運用するか、あるいはラッピングするかの選択肢がある。更新内容の即時反映が要求される場合や、アプリケーション移行のコストが問題となる場合はラッピングが有効であると言える。そこで我々はラッピングによるレガシー統合システムの構築に取り組んできた[1]。

本稿では、小型ホスト上のネットワーク型DBをSQLでアクセスするラッパーを開発した経験から、ラッパー内部での問合せ変換処理を実現する際の問題点を挙げ、その解決手段について述べる。

2. データベースラッピングの概要と問題点

2.1 ラッパーの役割

データベースのラッピングでは、ラッパーはクライアントに対して共通の問合せ処理APIを提供し、問合せ実行時にラッピングの対象とするDB(以降、ターゲットDB)が持つネイティブAPIを呼び出すことによって問合せ処理を実現する。共通の問合せ言語をSQLとしてネットワーク型DBをラッピングする場合、SQL問合せに含まれる問合せ条件を引数にしてネイティブAPIを呼び出し、実行結果として得られたレコードからテーブルを生成する。ここで、ネイティブAPIだけではSQLによる問合せ条件を表現できない場合は、元の問合せ条件よりも緩い条件でネイティブAPIを実行し、その結果を元の条件を用いてフィルタリングすることで、要求通りの結果を返すことができる[2]。

2.2 ラッパー実装時の問題点

図1にDBラッピング環境のシステム構成を示す。DB統合エンジンによる汎用機上の他のDBとの統合利用の目的があったため、ラッパーはPC側や中間サーバではなく、ターゲットDBと同一マシン上で稼動させる必要があった。また、ラッパーの稼動により業務APに悪影響を及ぼしては本末転倒なので、ラッパーは消費するリソースを最小限に抑えるように設計した。しかし、実際の開発を通して以下の問題点が明らかになった。

(1) ターゲットDBの問合せ表現力が低いためラッパーの問合せ処理の負荷が高くなってしまう
(2) ネイティブAPIが、ターゲットDBの物理的なファイル編成法に依存する場合がある

そこで(1)に関し、与えられた問合せ条件の最適化によって問合せ処理全体の効率を上げる問合せ最適化機構を実現した。また、(2)については、レコードを抽象データ型として捉え、DBの物理構成に依存した部分を隠蔽することで、最適化処理の実装を簡略化した。

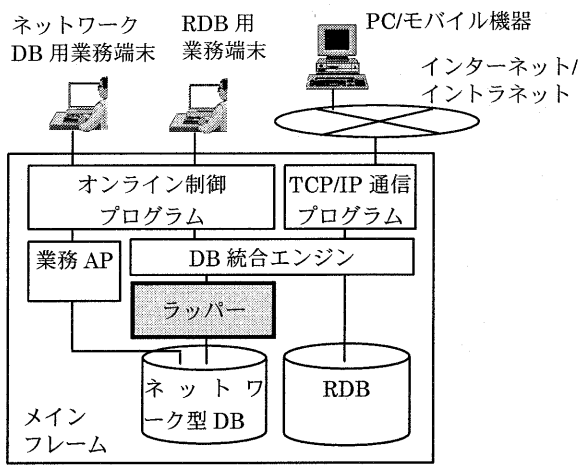

図1 ラッピングシステムの構成

3. 問合せ処理の実現手法

3.1 ターゲットDBの構造

ターゲットDBの構造を図2に示す。同一のフィールド定義を持つレコードの集まりをデータ集合と呼び、親集合と子集合の2種類がある。親集合は顧客情報など比較的不変な情報を、子集合は伝票やログなど親集合に従属した時系列的な情報を保持するのに利用される。親集合のレコードは子集合のレコードへのリンクを持ち、子集合のレコードは同一子集合内の他のレコードへのリンクを持つ。またフィ

Implementation of Query Processing for Database Wrapping
Takehiro Furudate and Mutsumi Komuro
Hitachi Software Engineering Co., Ltd.

ールド毎にインデックスの作成が可能である。

ネイティブAPIは指定した1種類のフィールド値を用いてレコードを特定する。インデックスを利用した直接・順次アクセス、リンクを辿る順次アクセス、ファイル編成法の物理的特徴を利用した順次アクセスが用意されている。例えば索引順編成ファイルで構築した親集合では、インデックスを用いなくてもプライマリキーの値順にレコードにアクセスできる。インデックスの無いフィールドの場合は、エントリ順に全件アクセスしなければならない。

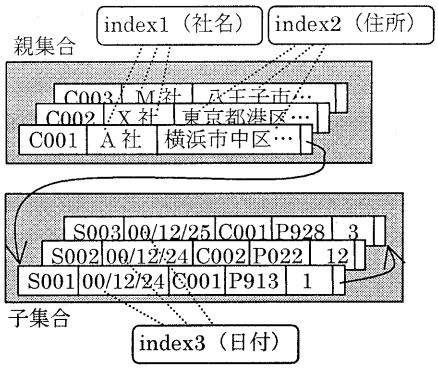

図 2　ターゲットDBの構造

3.2 問合せの最適化

今回のターゲット DB で利用できるネイティブAPIは基本的に値指定による直接アクセスと順次アクセスしかサポートしていない。すなわち、問合せ条件中で指定した値 a, b と、取り出したレコードのフィールド値 x に対して、x=a、x≧a、a≦x≦b(インデックス付きフィールドの場合のみ)のいずれかだけが実行可能な条件となる。そのため、SQL 中に与えられた問合せ条件と実行時に利用できる条件とのギャップが大きくなり、フィルタを通らない不要なレコードを取得する回数が多くなってしまう。また、問合せ条件に OR が含まれている場合は全件検索になり易い。DB アクセスは一連の問合せ処理の中で最も負荷の高い部分なので、検索範囲の絞り込みを中心とする最適化を行った。

問合せの最適化は以下の手順で行う。

(1) 問合せ条件中で指定されているフィールドごとに検索範囲を限定する。
(2) 複数のフィールドによる検索条件が与えられた場合は検索用フィールドを選択する。まずインデックスが無くプライマリキーも利用できないものは除外する。
(3) 残ったフィールドのうち、相対的に検索範囲の最も狭いものを検索用フィールドとする。

次の問合せ条件を例として考える。

(10<x<30) & (20<y<40) & (z=10) |
(20<x<40) & (10<y<30) & (z=20) |
(50<x<60) & (30<y<60)

OR による結合が含まれているので各フィールドでの検索範囲はそれぞれ、10<x<40,50<x<60、10<y<60、-∞<z<∞となる。図 3 は x,y に関する問合せ条件の範囲とレコードのアクセス状況を示している。レコードの配置に偏りが無いと仮定すると、範囲の狭い x フィールドで実行すれば、y フィールドで実行した場合と比べて不要なレコードの取得回数を少なくすることができる。

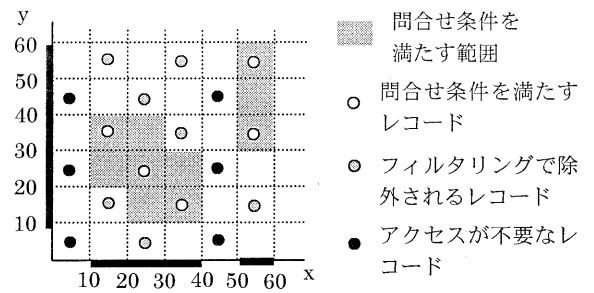

図 3 問合せ条件を満たす範囲

3.3 レコードの抽象化による設計の簡略化

指定された検索範囲のレコードを取得するために使用すべきネイティブAPIは、ターゲットDBを実現しているファイル編成法によって異なる。SQLの内容に応じてネイティブ API を選択する時点でこの場合分けを行うとラッパーの実装が複雑になってしまう。そこで、レコードを表す抽象データ型を定義し、まずはＳＱＬ問合せを抽象的な操作の並びとフィルタの組にマップする。そして、DB アクセス時にネイティブAPIを決定することで、ターゲットDBの実現形態を隠蔽しつつ問合せ最適化処理の実装を簡明にすることができた。

4. まとめ

汎用機上のネットワーク型DBのラッピングにおいて、検索フィールドの選択と検索範囲の絞り込みによりラッパーの問合せ処理効率を向上させた。また 2 段階の問合せ変換により、DB の物理的特性を隠蔽しつつ最適化処理の実装を簡略化することができた。今後はより効果的な最適化の実現を目指す。

参考文献

[1] 中重, 他：ラッピングによるネットワーク型データベースのシステム統合, 第 58 回情報処理学会全国大会講演論文集(3), pp. 239-240 (1999).
[2] Y. Papakonstantinou, et al. : A Query Translation Scheme for Rapid Implementation of Wrappers: DOOD95 (1995)