UNIX用メモリDBMSにおける高速インデックス方式*

6F－4

岡田 敏　梅田 昌義　黒岩 淳一†

NTT 情報通信研究所‡

1　はじめに

通信サービスではその制御のためにデータベースの使用が不可欠となってきている。従来の通信サービスでは要求される処理はユニークな一件検索という単純なものであるが、交換機等の通信機器がデータベースを利用するため、高レスポンス・高スループットが要求されている。この要求には、データベースをメモリ上で構築し、ハッシュ法を基にしたインデックスを使用して検索を行うデータベース管理システム [1](メモリ DBMS) を作成することで対応中である。

一方、WWW サーバ等でもデータベースの使用が盛んになっている。WWW の検索エンジンとしてデータベースを使用する場合、多数のユーザから検索要求を受付け、処理するため、交換機等に準じた高速レスポンス、高スループットが必要となる。処理内容は単純な検索であるが、範囲検索、重複キーを用いた複数件検索は必要である。我々はこれらの要求に応えるため、今まで開発を進めてきたメモリ DBMS を UNIX 上へ移植し (UNIX 用メモリ DBMS と呼ぶ)、ユニークな一件検索に加えて、範囲検索、複数件検索等が行えるインデックスをＴ木を基に構築し、ハッシュ法との共存を図った。本稿では、このインデックスの実現方式について述べる。

2　インデックス構成

UNIX 用メモリ DBMS におけるインデックス構成を図 1に示す。

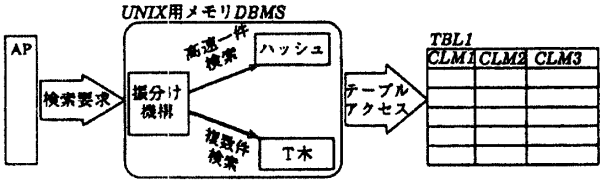

図 1: インデックス構成

この DBMS では１テーブルに対して高速一件検索用のハッシュと範囲検索用のＴ木インデックスが定義できる。AP からの検索の条件によって DBMS 内でインデックスの振り分けが行われる。ユニークキーに対する'=' 検索が要求されると、インデックスとしてハッシュが選択される。他の検索条件 (指定できる検索条件は=, <, >, <=, >=, like(但し前方一致のみ)) が指定されると、Ｔ木が選択される。

3　複数件検索用インデックス

3.1　Ｔ木

UNIX 用メモリ DBMS における複数件検索用インデックスはメモリデータベースに最適とされているＴ木 [2] を基にしている。図 2 にＴ木の形状およびＴノードと呼ばれるＴ木の１ノードを示す。

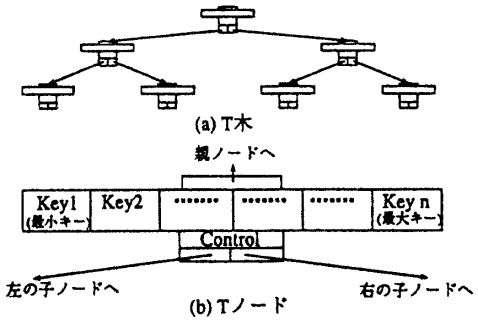

図 2: Ｔ木とそのノード

Ｔ木は以下のような特徴を持つ。

- １ノードに複数のキーを持つ、平衡二分木である。
- ノード内のキーは昇順に並び、範囲検索の実現が容易である。

Ｔ木におけるキー値の検索は AP から与えられた検索キーを使用してルートノードから始め、以下に示す順序で下位のノードへと検索を進めていく。

1. "検索キー ＜ ノード内最小キー" ならば、左の子ノードへ移動。
2. "検索キー ＞ ノード内最大キー" ならば、右の子ノードへ移動。
3. "最小キー ≦ 検索キー ≦ 最大キー" ならば、現在見ているノード内を検索し、目的のキーを見付ける。

---

*High-speed Index in Memory Resident DBMS for UNIX

†Satoshi OKADA, Masayoshi UMEDA, Jun'ichi KUROIWA

‡NTT Information and Communication Systems Laboratories

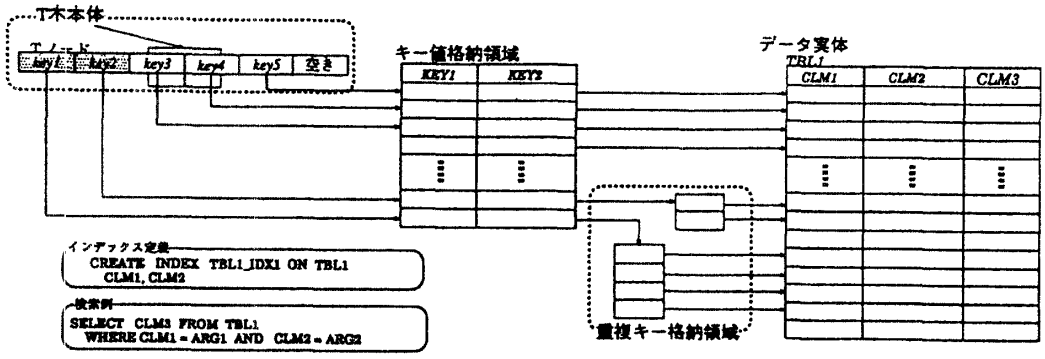

図 3: インデックスの構造

3.2 T木インデックスの実現

T木をマルチカラムおよび重複キー対応とするためにUNIX用メモリDBMSで実現したインデックスの概要を図3に示す。

3.2.1 格納構造

このインデックスは以下の3領域に分けて格納されている。

- **T木本体：** 3.1で説明したT木を実現する。しかし、各ノードには実際のキー値は保持せず、実際のキー値を格納しているキー値格納領域へのポインタのみを保持する。
- **キー値格納領域：** 検索時に使用する実際のキー値を保持している。各キー値はデータ実体からインデックス生成時に定義された順に値を取り出し、一つのキー値とする。各キー値はT木本体の各要素からポイントされ、データ実体へのポインタを保持している。
- **重複キー格納領域：** 重複キーを扱う場合に重複している各レコードへのポインタを保持している。

このような領域管理とすることで、以下のような特徴が生じる。

- **T木の作成、再構成が容易である：** T木本体はキー値を持たないため、T木本体の構成はキー値の長さ、キーを構成するカラム数に依存しない。また、重複キーも別領域で管理されるため、1キーに対するレコードの重複数も影響しない。
- **メモリ使用効率が良い：** Tノードで、実際には使用されていない領域(図3の'空き')はポインタ領域だけで済むため、メモリの使用効率が良い。

3.2.2 アクセス方法

このインデックスを使用した、図3の'検索例'に沿った検索は以下の順序で行われる。

1. Tノード内のポインタをたどり、実際のキー値を取り出す。
2. *KEY1* と *ARG1* を比較し、右の子ノードへ移動するか、左の子ノードへ移動するかを特定する。
3. *KEY1=ARG1* の場合には *KEY2* と *ARG2* との比較を行い移動すべき子ノードを特定する。
4. *KEY1=ARG1,KEY2=ARG2* であれば、検索条件に合致するため、キー値格納領域からポインタをたどり、データ実体へアクセスする。

このように、マルチカラムインデックスが定義されている場合には、インデックス生成時に定義した順に検索キーとの比較を行い、検索対象のレコードを特定する。

また、重複キーの場合には、まずキー値を特定し、レコードが重複しているかチェックする。レコードが重複している場合には、重複キー格納領域内のリストを使用し、データ実体へアクセスし、目的のデータを取り出す。

以上のように、UNIX用メモリDBMSのインデックスでは複数のポインタを経由し、データ実体へアクセスする。ディスクアクセスの生じないメモリ常駐インデックスであるため、ポインタ経由のデータアクセスでも高速性が維持できるものと考えている。

4 まとめ

本稿では、UNIX用メモリDBMSで使用する範囲検索、重複キーの扱いが可能なインデックスの実現方式について述べた。今後は実際にWWWサーバ等で使用し、参照および更新時のリバランス等の性能評価を実施する予定である。

参考文献

[1] T.Honishi, N.Kobayashi and J.Nakamura: Design and Implementation of an Enhanced Relational Database Management System for Telecommunication and Network Applications, in *Proceedings of Pacific Telecommunications Council Eighteenth Annual Conference*,pp.698-703, 1996.

[2] T.J.Lehman and M.J.Carey: A study of Index Structure for Main Memory Database Management System , *Proceeding of 12th Vary Large DataBase*, pp.294-303, 1986.