データベースS 講義資料　第12回 物理的データ格納方式

九州工業大学 情報工学部 システム創成情報工学科　講義担当：尾下真樹

1. 物理的データ格納方式

リレーショナルデータベースでは、データベースに格納されるデータは、リレーション（表形式のデータ構造）としてモデル化される。リレーションはあくまで概念的なモデルであり、実際には、リレーションは何らかの方法でハードディスク上に配置されて格納されることになる。一般にデータベースシステムでは大量のデータを扱う必要があるため、全てのデータはメモリ上には入りきらず、ハードディスク上に格納される必要がある。しかし、ハードディスクへの読み書きは、メモリへの読み書きに比べると、ハードウェアの制約上、多くの処理時間がかかる。従って、高速な処理を実現するためには、ハードディスクに対して効率的にデータを読み書きできるような、データの格納方法や検索方法の工夫が必要となる。リレーショナルデータモデルは概念的なモデルであり、このような格納・検索方法は規定されていないため、各データベースシステムで工夫されることになる。

1.1. データ格納方法

ハードディスク上にリレーションを格納する際には、リレーション中の各データ（インスタンス、タプル）を一つのレコードとし、レコードを連続的に配置することになる。このとき、ハードディスク上での配置順序を特に考慮せず、追加されたデータを順番に追加していく**順序なしの格納**と、キー属性値にもとづいて大小関係を保つように整列して格納する**順序付きの格納**の大きく２通りの格納方法がある。順序なしの格納の場合、挿入の処理が単純である代わりに、検索に時間がかかる。一方、順序付の格納の場合、データの追加を行う際にはソートを行った上で適切な位置に挿入する必要があるため、挿入の処理が複雑になる代わりに、主キー属性にもとづく検索を効率的に行える。これら以外に、順序なしの格納の特殊な例として、ハッシュを使った格納方法がある（詳細は後述）。

1.2. データ検索方法

データベースを利用するときには、指定された条件にもとづいてリレーションからデータを検索する処理が必要である。データの格納方法に応じて、利用可能な検索方法は異なる。データが順序なしで格納されている場合は、基本的に全てのデータを確認して、条件に合うかどうかを検索する**スキャン法**を用いることになる。一方、順序付きで格納されている場合は、探索範囲の中央のデータの属性値と検索条件を比較することで探索範囲を半分に狭める処理を繰り返していく、**探索法**を用いることができる。

検索を効率的に実現するための方法として、リレーションに対して**インデックス**を付加することができる。インデックスを付加することによって、追加の格納領域が必要になる代わりに、リンデックスを付加した属性での検索を効率的に実現することができる。インデックスは、順序付き格納・順序なし格納どちらのリレーションに対しても付加することができる。順序付き格納されたリレーションについても、インデックスを付加することで、より効率的に検索を行ったり、キー属性以外の属性でも検索を行ったりすることができる。

インデックスの更新や検索を効率的に行う方法として、**ツリー（多段インデックス）**がある。ツリーの例として、B+-ツリーがある。B+-ツリーは、下位ノードへの参照を格納する中間ノードと、データ（もしくはデータへの参照）を格納する葉ノードから構成され、どの部分でもツリーの高さが同じ平衡木となる（常に平衡木となるように更新処理が行われる）。ツリーの最下段のみが葉ノード、それ以外の段は全て中間ノードとなる。中間ノードに p 個、葉ノードに q 個の参照・データが格納であるとすると、中間ノードに $p/2$～p 個、葉ノードに $(q-1)/2$～q 個の参照・データが格納されるように、ツリー更新時にノードが分割・統合される。データを検索するときには、一番上のルートノードから、検索条件のキー属性値（ツリーの作成に用いられている属性値）にもとづいて、中間ノード・葉ノードを辿っていくことで、データを探索できる。データを挿入するときには、検索と同様の方法でデータを格納するべき葉ノードを探索し、葉ノードに領域が空いていれば、その葉ノードにデータを追加する。葉ノードに領域が空いていなければ、葉ノードを 2 つに分割する。このとき、ノードを分割したことによって、上の中間ノードに領域も足りなくなったら、上の中間ノードも分割する。さらに上の中間ノードにも、同様の処理を繰り返し行う。同様に、データを削除した場合には、葉ノード・中間ノードに格納される最少個数のデータを下回ると、2 つのノードを 1 つのノードに統合する処理を、下から上に繰り返し行っていく。

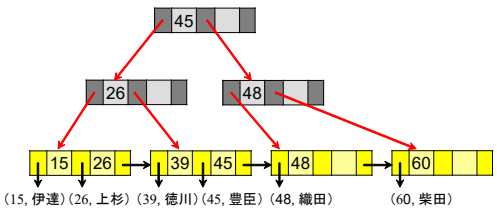

図 1　B+-ツリーの例

ハッシュ法は、順序なしのデータ領域をいくつかの領域（バケット）に分け、何らかのハッシュ関数を用いて、各データをどのバケットに格納・検索するかを決定するものである。ハッシュ関数は、キー属性値を入力としてハッシュ値（バケット番号）を返す関数であれば、どのような関数でも構わない。検索処理では、検索条件の属性値に対応するバケット内からスキャン法を用いてデータを検索することになる。データ数に対してバケット数が十分にあり、データが全バケットに均等に分布していれば、効率的に検索を行える。ただし、ハッシュ法は、等号検索にしか用いることができず、範囲検索には用いることはできない。