2.8 インデックス

2.8.1 インデックスとは

インデックス(索引)は、データベース上の特定のデータを、効率よく検索する手段として用いられます。例えば図書館で大量の本の中から目的の本を速く見つけたいときに、タイトルや著者名を索引として使うのと同じ概念です。

データ項目の値を条件とし特定データを検索する場合、当該データ項目に対しインデックスが作成されていると、検索速度が向上します。

インデックスは、実行する処理内容に応じて、ひとつのテーブルに対し複数作成することができます。

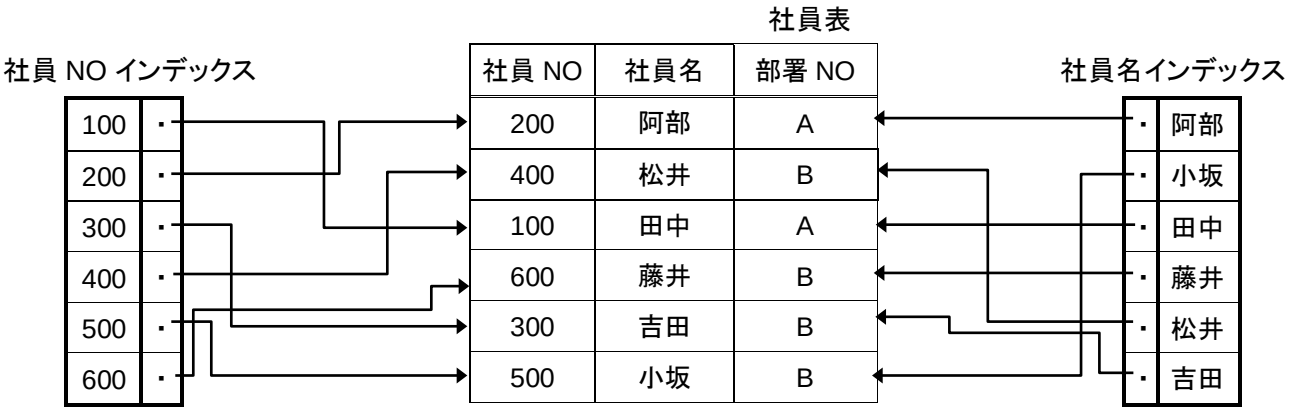

図 2-18 インデックスとは

2.8.2 B+treeインデックス

インデックスの構造には、B+tree、ハッシュ、Bitmap 等が存在しますが、ここでは最も広く使われている B+tree インデックスについて述べます。B+tree インデックスを用いたデータベース上のデータに対するアクセスの仕組みを以下に示します。

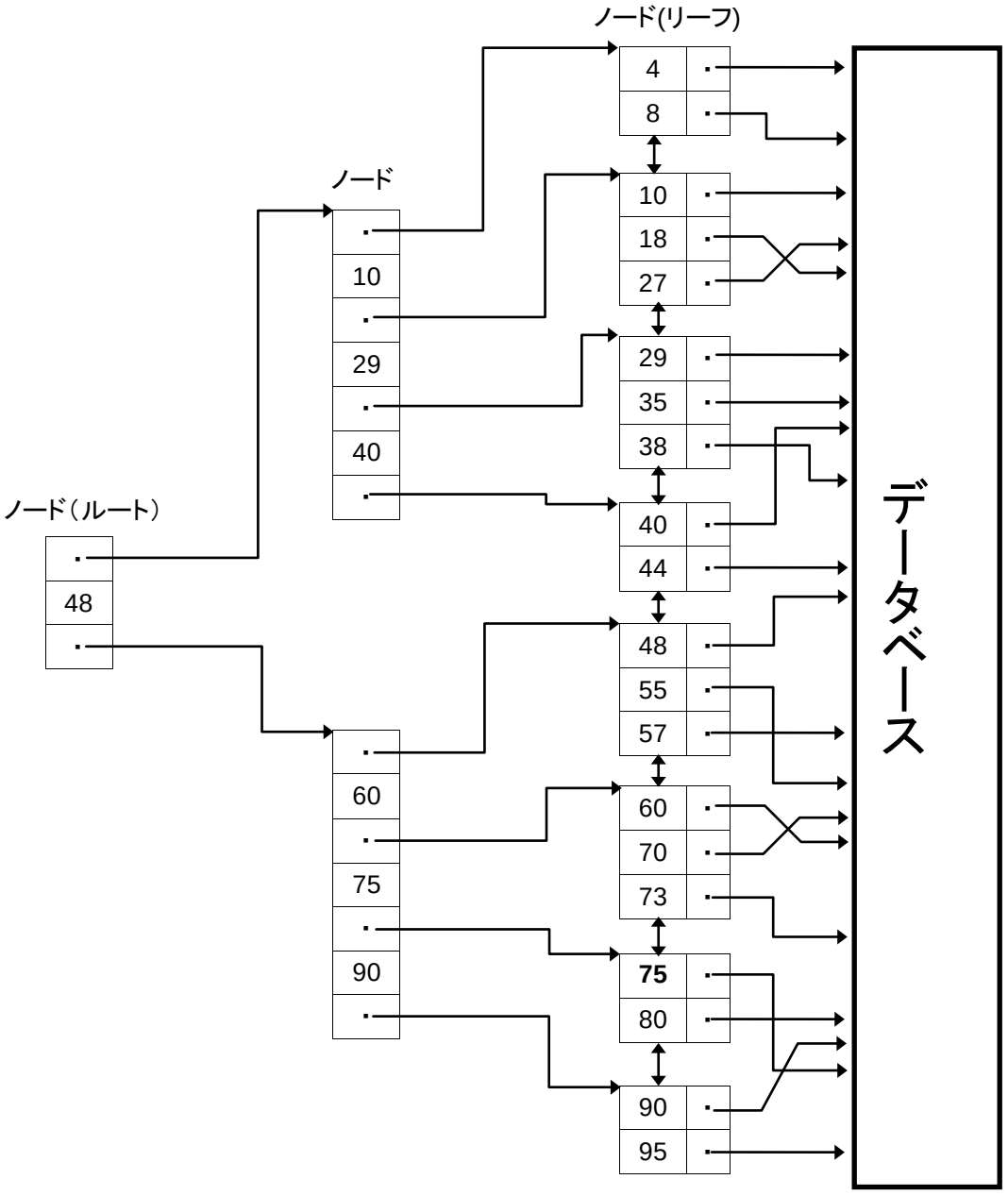

図 2-19 インデックスの構造

B+tree インデックスのデータ構造は、複数のノード（節）から構成されている木構造です。1 ノードは 1 ページで実現されています。また、ノードごとにキー値がソートされているので、目的のデータがノード中のキー値よりも小さければその前のポインタをたどり、そうでなければ次のキー値と比較します。ノード内の最後のキー値よりも大きければ最後のポインタをたどります。これをリーフレベルのノードまで繰り返せば目的のデータを探すことができます。

B+tree インデックスはどのデータをアクセスするときでも、たどっていくノードの数は同じです。つまりインデックスの木構造がどこをとっても高さが一定になるように保たれています。

2.8.3 B+treeインデックス作成における留意点

インデックスは検索のパフォーマンスを向上させるものですが、むやみにインデックスを付ければよいというわけではありません。インデックスは、以下に述べることに十分留意し作成すべきです。特にリレーショナルデータベースの場合、インデックス設計がパフォーマンスチューニング上、最も重要な要素のひとつとなります。

(1) 利用目的に合わせたインデックスの作成

インデックスはデータ検索のパフォーマンスは高めますが、逆にデータ更新のパフォーマンスは下げてしまう場合があります。なぜなら、データが更新されるとそれに伴って、そのデータに対するインデックスも DBMS によって自動的に更新されるからです。

したがって更新のパフォーマンスが検索のパフォーマンスよりも重要な場合には、インデックスの作成は必要最小限とします。逆に検索のパフォーマンスが更新のパフォーマンスよりも重要な場合には、積極的に、適切なデータ項目に対しインデックスを作成すべきです。

(2) インデックスを付けるべきデータ項目

以下の項目にインデックスを作成することにより、検索のパフォーマンス向上が見込めます。

- 検索条件に使用することの多い項目
- ソート、グループ化に使用されることの多い項目
- 表結合に使用されることの多い項目

基本的にB+treeインデックスは、選択性係数*[11](検索条件にヒットする率)が小さい場合に効果を発揮します。つまり10000件のデータの中から1、2件のデータを取得するような場合は、B+treeインデックスを使用すると検索速度が向上します。

また B+tree インデックスはキー値がソートされているため、アプリケーションプログラムの中で特別なロジックを組まなくてもデータを順番に得ることができます。つまり、ソートキーに対してインデックスがあるとソートのオーバーヘッドを削減できます。

さらにリレーショナルデータベースの場合、表結合に使用する列に対してインデックスを付けるのがセオリーです。したがって主キー、外部キーに対してはインデックスを付けるべきです。

*[11] 選択性係数とはアクセスすべきデータ件数を全体のデータ件数で割ったものです。

(3) インデックスを付けても効果の期待できないデータ項目／テーブル

以下の項目（またはテーブル）には、インデックスを付けてもパフォーマンスの向上は見込めません。さらにインデックスは、データとは異なる領域を必要とするので、使用されない様なインデックスを定義すると、領域を無駄に使用することになります。

- **検索の条件に使用することがあまりない項目**
- **データ件数の少ないテーブル**
- **重複したデータの多い項目**

選択性係数(検索条件にヒットする率)が大きい場合、インデックスを使用するよりシーケンシャルアクセス*[12]を行う方が、速い場合があります。例えば、10000 件のデータの中から 3000 件を検索するような場合は、インデックスを使用せずシーケンシャルアクセスを行う方が速いといえます。

例えば性別や商品分類など重複したデータの多い項目（値の種類の少ない項目）を検索条件とする場合、必然的に選択性係数が大きくなるためインデックスは使用されません。したがって、重複したデータの多い項目にインデックスを付けても意味がありません。

*[12] シーケンシャルアクセスとはインデックスを使わずに、格納順にデータを最初から最後までアクセスすることです。