第2章 インデックス

ここでは、SQL Serverのインデックス機能について記述します。

インデックスの概要

SQL Serverのインデックスには、クラスタ化インデックスと非クラスタ化インデックスがあります。どちらもBツリー方式です。インデックスはインデックスページに格納され、非リーフレベルとリーフレベルに分けられます。リーフレベルのページのデータは、インデックスキーの順番（降順・昇順）に並んでおり、非リーフレベルはリーフレベルのデータをポイントするデータが階層的に作成されます。図2-1は、インデックスの基本的な概念図です。

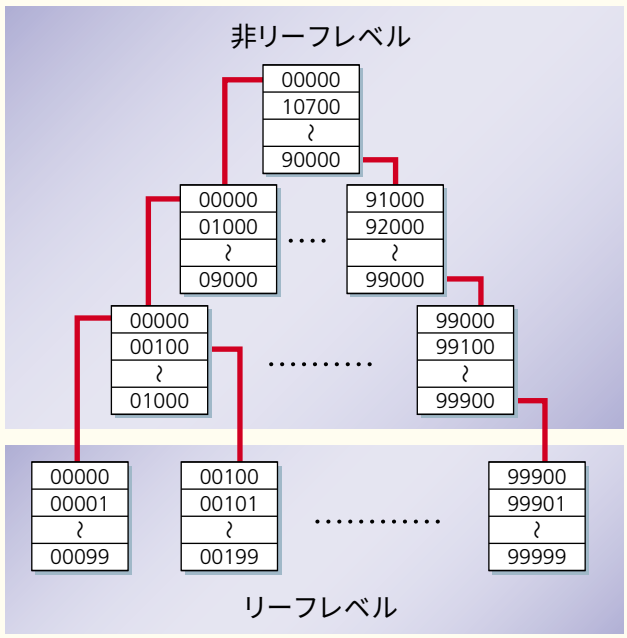

図2-1
インデックス

インデックスの作成には、大量の作業領域が必要になります。SQL Serverでは、SORT_IN_TEMPDBオプションを使用すると、作業領域としてtempdbを指定できます。tempdbが別ハードディスクにとられている場合などはインデックスの作成が効率的に行えます。

クラスタ化インデックス

クラスタ化インデックスでは、リーフレベルが実際のテーブルのデータページになります。つまり、テーブルのデータは物理的にインデックスキーの

順番に並んでいます。リーフレベルが実際のテーブルのため、インデックスは非常に小さくなります。テーブルには、1つだけクラスタ化インデックスを作成できます（図2-2）。これは、Oracleの索引構成表と同等です。

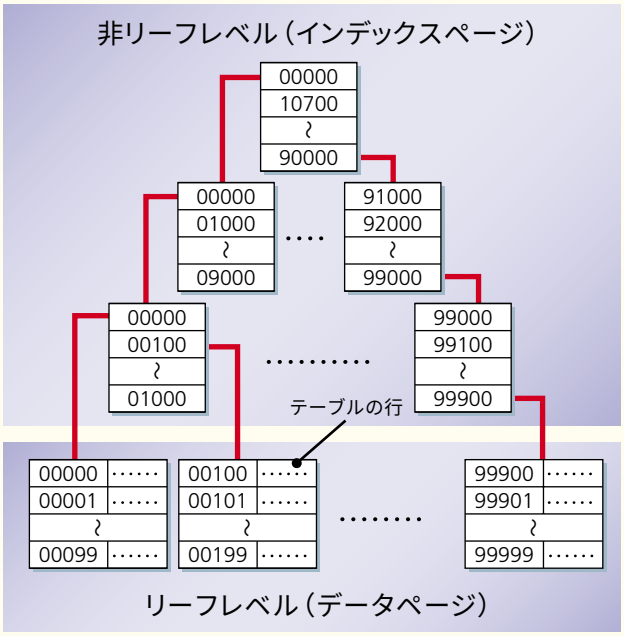

図2-2
クラスタ化インデックス

非クラスタ化インデックス

非クラスタ化インデックスのリーフレベルの各行には、データへのポインタが入っています。そのポインタは、クラスタ化インデックスが構築されていないテーブルでは行識別子（ファイルID＋ページ番号＋行ID）が入っており（図2-3）、クラスタ化インデックスが構築されている場合はクラスタ化インデックスのキーが入っています（図2-4）。非クラスタ化インデックスは、1つのテーブルに複数作成することができます。

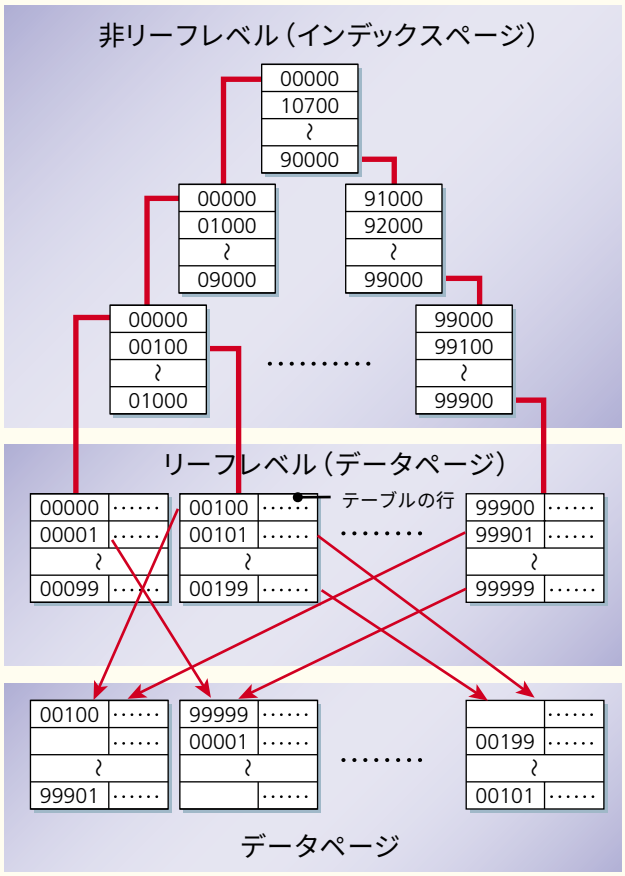

図2-3
非クラスタ化インデックス
（クラスタ化インデックスなし）

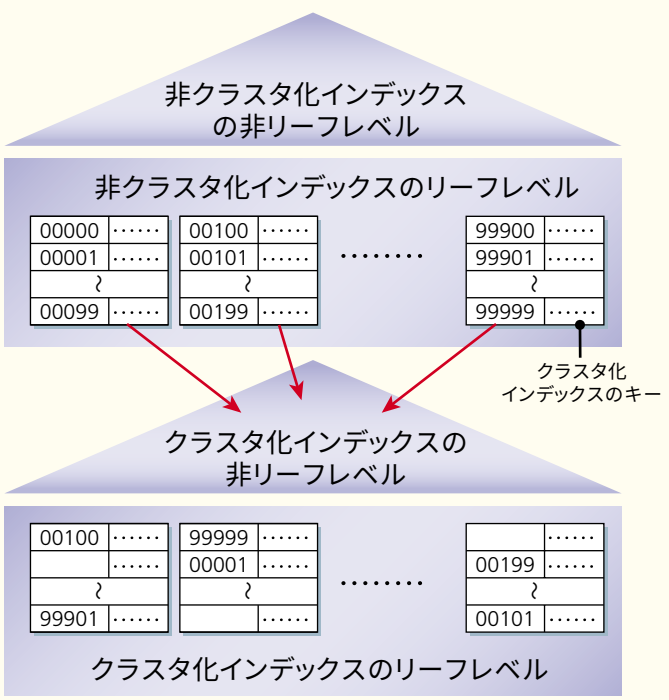

図2-4
非クラスタ化インデックス
（クラスタ化インデックスあり）

クラスタ化インデックスを構築しているテーブルに非クラスタ化インデックスを構築した場合、検索パフォーマンスは若干悪くなりますが、データページの物理位置を非クラスタ化インデックスが持たないため、データの修正あるいは保守時に高いパフォーマンスが得られます。

データのフラグメンテーション（断片化）

SQL Serverでは、データの修正によりページ分割が発生し、データのフラグメンテーションが発生します。フラグメンテーションの状況を調べるには、DBCC SHOWCONTIGステートメントを使います。以下に、実際の実行結果を示します。論理スキャンフラグメンテーションが0%のため論理的には順番に並んでいますが、スキャン密度が100%でないため、物理的にフラグメントが発生しているのがわかります。

```
DBCC SHOWCONTIG ([order details])
```

【実行結果】

- スキャンされたページ数 …… 9
- スキャンされたエクステント数 …… 6
- 切り替えられたエクステント数 …… 5
- エクステントごとの平均ページ数 …… 1.5
- スキャン密度［最善:実際］…… 33.33% [2:6]
- 論理スキャンフラグメンテーション …… 0.00%
- エクステントスキャンフラグメンテーション…… 16.67%
- ページごとの平均空きバイト数 …… 673.2
- 平均ページ密度（全体） …… 91.68%

フラグメンテーションを解消するには、クラスタ化インデックスが付いている場合はクラスタ化インデックスの再構築を行い、クラスタ化インデックスがない場合は、データの再ロードを行います。前記のテーブルのクラスタ化インデックスの再構築を行うSQLステートメントを以下に示します。また、実行後のDBCC SHOWCONTIGの結果を見ると、フラグメンテーションが解消されたことがわかります。

```
CREATE UNIQUE CLUSTERED
INDEX [PK_Order_Details] ON [dbo].[Order Details]
   ([OrderID], [ProductID])
WITH  DROP_EXISTING
```

【インデックス再構築後の実行結果】

- スキャンされたページ数 …… 9
- スキャンされたエクステント数 …… 2
- 切り替えられたエクステント数 …… 1
- エクステントごとの平均ページ数 …… 4.5
- スキャン密度［最善:実際］…… 100.00% [2:2]
- 論理スキャンフラグメンテーション …… 0.00%
- エクステントスキャンフラグメンテーション …… 0.00%
- ページごとの平均空きバイト数 …… 673.2
- 平均ページ密度（全体） …… 91.68%

フラグメンテーションをできる限り抑えるには、FILLFACTORオプションをインデックスに指定します。FILLFACTORオプションは、リーフレベルのページにデータをどれだけ満たすかをパーセンテージで指定します。ただし、0%は100%と同じ意味になるので注意してください。以下のSQLステートメントの場合は80%なので、データ修正のために20%の空き領域を確保したことになります。

```
CREATE UNIQUE CLUSTERED
INDEX PK_Order_Details ON dbo.[Order Details] (OrderID,
  ProductID)
WITH FILLFACTOR = 80
```

FILLFACTORのデフォルトは0です。この値は、サーバーのプロパティで変更できます（図2-5）。

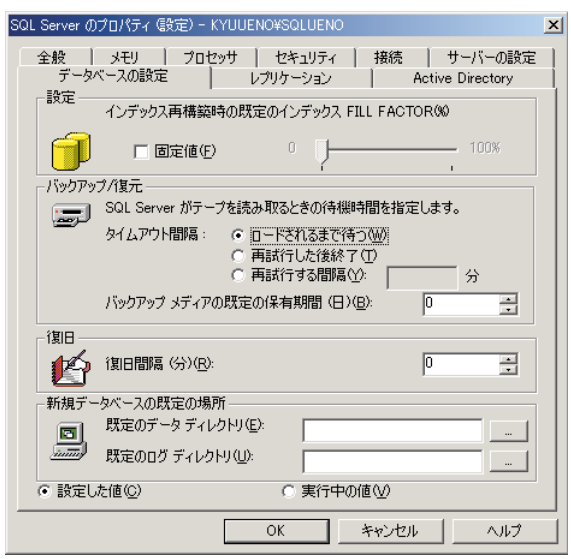

図2-5
［SQL Serverのプロパティ］で
FILL FACTORの値は変更可能
(SQL Server Enterprise Manager)

さらに、非リーフレベルのデータに空き領域を設定したい場合は、PAD_INDEXオプションを使用します。これはFILLFACTORと同等のパーセントを割り当てるため、FILLFACTORオプションと一緒に使います。PAD_INDEXを省略するとインデックスページに最低1行格納できるだけの空き領域を確保します。以下のSQLステートメントは、20%の空き領域を確保したことになります。

```
CREATE UNIQUE CLUSTERED
INDEX PK_Order_Details ON [Order Details] (OrderID,
   ProductID)
WITH PAD_INDEX, FILLFACTOR = 80
```

統計情報

SQL Serverでは、SQLステートメントの実行時にどのインデックスを使うかはクエリオプティマイザに任されており、クエリオプティマイザが最適なものを選択します。最適なインデックスを選択するには、テーブル内にどのようなデータがどれだけ入っているかを知る必要があります。インデックスを作成すると、インデックス列内の値の分布に関する統計情報をSQL Serverが自動的に格納します。また、データの修正による統計情報の変更はデフォルトでサンプリング自動更新になっています（サンプリングとは、すべてのデータページの統計情報ではなくランダムに情報を収集するということです）。たとえば、サンプリングの方法の設定や統計情報の更新のスケジュールリングを行いたい場合は、自動化せずにUPDATE STATISTICSステートメントを使うことも可能です。

AUTO_CREATE_STATISTICSデータベースオプションがON（デフォルト）の場合は、インデックスを持たない列についても自動的に統計情報が作成されます。統計情報が使用されない場合は、自動的に削除されます。

インデックスの指定

SQL Serverでは、Transact-SQLステートメントにオプティマイザヒントを指定することにより、インデックスを明示的に選択することも可能です。以下の例では、name_indという名前のインデックスを明示的に指定しています。

```
SELECT * FROM employee WITH (INDEX(name_ind))
WHERE name = '鈴木'
```

INDEX(0)を指定するとテーブルスキャン（全表検索）となり、INDEX(1)を指定するとクラスタ化インデックスとなります。