はじめに

■コース概要と目的

Oracle をより効率的に使用するための SQL チューニング方法を説明します。また、索引の有無、SQL の記述方法がパフォーマンスにどのように影響するのかを実習を通して習得します。

■受講対象者

アプリケーション開発者/データベース管理者の方。

■前提条件

「SQL トレーニング」「データベース・アーキテクチャ」コースを受講された方、もしくは同等の知識をお持ちの方。

■テキスト内の記述について

▼構文

[　]	省略可能
{ A | B }	A または B のどちらかを選択
n	数値の指定
_	デフォルト値

▼マーク

12.1	指定バージョンからの新機能 （左記の場合、Oracle 12cR1 からの新機能）
EE	Enterprise Edition で使用できる機能
NOTE	知っておいたほうが良いテクニック、もしくは注意事項
	参照ページ
	データ・ディクショナリ・ビュー

第 5 章

結合

結合のパフォーマンスに影響を与える結合の種類と、表の結合順序について内部動作を交えて説明します。

1．結合処理のチューニング概要
2．結合の種類
3．結合順序
4．結合処理のチューニングポイント
5．結合関連のヒント

1.結合処理のチューニング概要

結合とは複数の表からデータを取り出す操作です。結合するには、問合せ文のFROM句に複数の表を指定し、結合する各表の関係をWHERE句内に結合条件として設定します。
※ANSIの場合、JOIN句に複数の表を指定し、ON句（またはUSING句）に結合条件を指定します。

結合のパフォーマンスを向上するには、最も適切な結合方法と結合順序を選択することが重要です。

（1）結合の種類

結合方法には、主に以下の3つの種類があります。

・ハッシュ結合
　内部的にハッシュ関数を使用して結合します。

・ソート/マージ結合
　結合条件に指定した各列をソートし、その結果をマージします。

・ネステッド・ループ結合
　大規模な表と小規模な表（またはWHERE句で行ソースを絞っている）を、索引を使用して結合します。

結合の種類	有効なシーン	結合する表の特徴	使用する条件
ハッシュ結合	結合結果が大量	大規模な表と小規模な表	結合条件が等価のみ使用可能
ソート/マージ結合	結合結果が大量	結合する表が同規模	結合条件が非等価でも使用可能
ネステッド・ループ結合	結合結果が少量	大規模な表と小規模な表	大規模な表側に索引が必要

（２）結合順序

結合では２つの表ずつ処理されます。そのため、３つ以上の表の結合では、まず２つの表を結合し、その結果作成された行ソースと３つ目の表を結合、さらにその結果と４つ目の表を結合、というように処理されます。このため、３つ以上の表の結合では何通りかの結合順序が考えられます。

例）A表、B表、C表を結合する場合の順序の候補

・A ⇒ B → C
・A ⇒ C → B
・B ⇒ A → C
・B ⇒ C → A
・C ⇒ A → B
・C ⇒ B → A

各結合順序によって実行負荷は異なります。そのため、実行負荷が低くなるよう、結合結果が小さな行ソースとなるものから結合するように調整します。

２．結合の種類

各結合の内部的な動作と、どのようなときに利用されるかを解説します。

（１）ハッシュ結合

ハッシュ結合は、小規模な表と大規模な表の結合で、表の大部分の行が結合対象である場合に向いています。多くの場合、ソート/マージ結合よりも効率的に実行できます。

１）内部動作

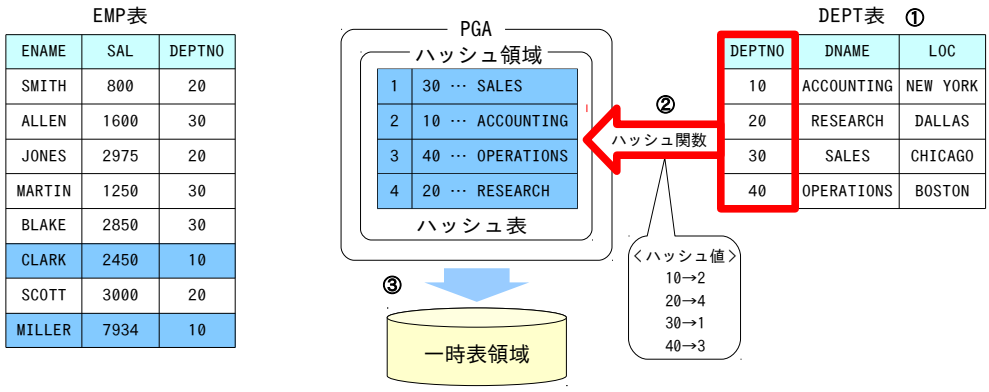
ハッシュ結合は、結合する一方の表にハッシュ関数を適用してメモリー上（ハッシュ領域）に展開した後、もう一方の表にもハッシュ関数を適用してハッシュ値が等しいデータを結合する方法です。
ハッシュ結合は、以下のステップによって行われます。

１．結合する２つの表を比較し、小さな表を選択する。
２．選択した表の結合条件列にハッシュ関数を適用し、ハッシュ領域に展開する（ハッシュ表の作成）。
　※ハッシュ領域はPGA内に獲得されます。また、表をハッシュ領域に格納できなかった場合は、一時セグメントとしてディスクに書出されます。
３．もう一方の大きな表の結合条件列にハッシュ関数を適用しハッシュ表と比較する。同じハッシュ値の行があった場合は結合する。この作業を行ごとに繰り返す。

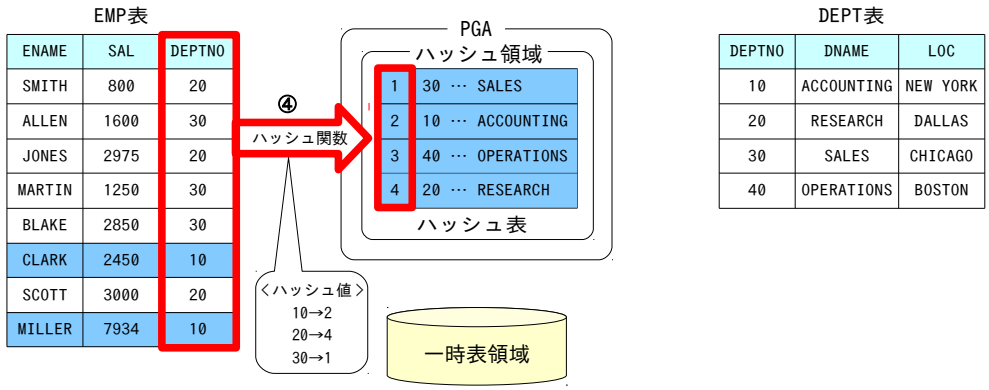
<実行 SQL>

```
SELECT ename,dname
FROM   emp,dept
WHERE  emp.deptno = dept.deptno;
```

①小さな表として DEPT 表を選択する。
②DEPT 表の DEPTNO 列にハッシュ関数を適用し、ハッシュ領域に格納する（ハッシュ表の作成）。
③表がハッシュ領域に収まり切らない場合は、一時セグメントとしてディスクに書出す。



④もう一方の EMP 表の DEPTNO 列にハッシュ関数を適用。その結果とハッシュ表を比較し、同じハッシュ値の行があれば結合する。



2）ハッシュ結合が選択されるケース

オプティマイザは以下のようなケースでハッシュ結合を選択する可能性があります。

・表の大部分の行を結合する場合。
⇒通常、ソート処理よりもハッシュ処理のコストの方が低いため、ハッシュ結合が選択されやすいといえます。

・結合条件が等価（=）である場合。
⇒非等価演算子（　<、>、<=、>=　）では、ハッシュ結合は行われません。

例）EMP表とDEPT表を結合したときの実行計画を確認する。

```
SQL> SELECT ename,dname FROM emp,dept
  2  WHERE  emp.deptno = dept.deptno;

実行計画
----------------------------------
| Id  | Operation          | Name |
----------------------------------
|   0 | SELECT STATEMENT   |      |
|*  1 |  HASH JOIN         |      |
|   2 |   TABLE ACCESS FULL| DEPT |
|   3 |   TABLE ACCESS FULL| EMP  |
----------------------------------
```

<HASH JOIN>

ハッシュ結合が行われたことを示します。

先に全表スキャンされている表がハッシュ表になります。上記例ではDEPT表がハッシュ表です。

（2）ソート/マージ結合

ソート/マージ結合は、結合条件の各列をソートし、その結果をマージします。
データ量の多い表同士を結合し、表の大部分の行が結合対象である場合に向いています。
※一般的にソート/マージ結合よりもハッシュ結合の方がパフォーマンスが優れています。

1）内部動作

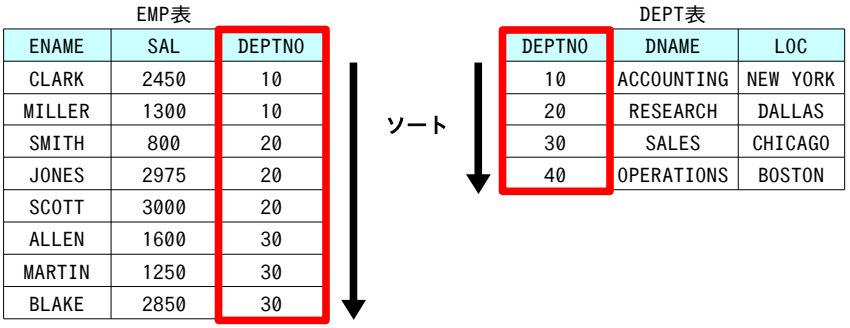
ソート/マージ結合は、内部的に以下ステップで行われます。

1．ソート操作　　：両方の表が、結合条件列をもとにソートされる。
2．マージ結合処理：ソートされた各表の行ソースがマージされる。

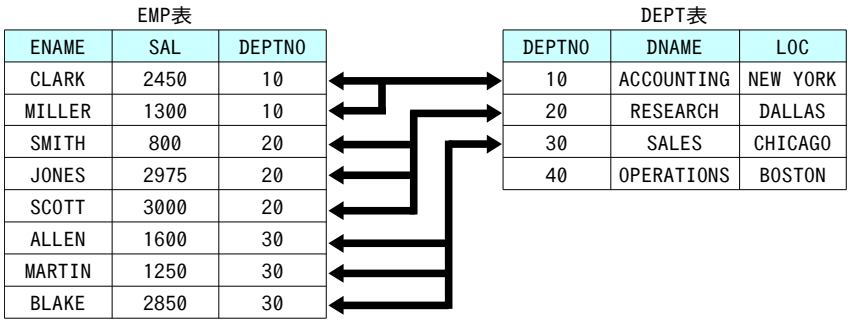
<実行 SQL>

```
SELECT ename,dname
FROM   emp,dept
WHERE  emp.deptno = dept.deptno;
```

①それぞれの表の結合条件列（DEPTNO 列）をもとにソートする。



②ソートされたそれぞれの行ソースを、結合条件をもとにマージする。



2）ソート/マージ結合が選択されるケース

オプティマイザは、以下のようなケースでソート/マージ結合を選択する可能性があります。

・表の大部分の行を結合する場合。

・結合条件が非等価演算子（ <、>、<=、>= など ）である場合。
⇒ハッシュ結合は、等価条件（=）でないと使用されません。

・行ソースが既に他の操作などでソートされており、ソート処理を避けることができる場合。
⇒例えば、結合条件の列に索引が作成されている場合、索引を利用することでソート処理を省略でき、ハッシュ結合よりも低いコストで結合できる場合があります。

例）EMP表とDEPT表を結合したときの実行計画を確認する。

```
/* 表の大部分の行を結合する場合、ハッシュ結合が選択される可能性が高い */
SQL> SELECT ename,dname FROM emp,dept
  2  WHERE  emp.deptno = dept.deptno;

実行計画
----------------------------------
| Id  | Operation          | Name |
----------------------------------
|   0 | SELECT STATEMENT   |      |
|*  1 |  HASH JOIN         |      |
|   2 |   TABLE ACCESS FULL| DEPT |
|   3 |   TABLE ACCESS FULL| EMP  |
----------------------------------

/* EMP表の結合条件列であるDEPTNO列に索引を作成 */
SQL> CREATE INDEX idx_deptno ON emp(deptno);

索引が作成されました。

/* 索引を作成したことにより、ソート/マージ結合が選択された */
SQL> SELECT ename,dname FROM emp,dept
  2  WHERE  emp.deptno = dept.deptno;

実行計画
-------------------------------------------------
| Id  | Operation                    | Name       |
-------------------------------------------------
|   0 | SELECT STATEMENT             |            |
|   1 |  MERGE JOIN                  |            |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP        |   } EMP表へのアクセスでは、索引をフルスキャンし、索引からソート済みデータを取得
|   3 |    INDEX FULL SCAN           | IDX_DEPTNO |   }
|*  4 |   SORT JOIN                  |            |   } DEPT表のDEPTNO列には索引がないため、全表スキャン後にソートが発生
|   5 |    TABLE ACCESS FULL         | DEPT       |   }
-------------------------------------------------
```

※結合条件列に索引が作成されたため、ソート処理をスキップできるようになった。これにより、ソート/マージ結合のコストがハッシュ結合を下回り、ソート/マージ結合が使用されるようになった。

<MERGE JOIN>

ソート/マージ結合が使用されたことを示します。