はじめに

■コース概要

Oracle Database をより効率的に使用するための SQL のチューニング方法について説明します。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理解します。

■コースのゴール

・適切な SQL 診断ツールを使用して、高負荷 SQL の情報を確認できる。
・実行計画を読み、SQL のどこに問題があるかを判別できる。
・索引作成のガイドラインを理解して、どこに索引を作成すべきか判断できる。
・索引を効率的に使用する SQL を記述できる。
・結合処理のチューニング・ポイントを理解して、効率的な結合処理を実行できる。

■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、同等の知識をお持ちの方

■テキスト内の記述について

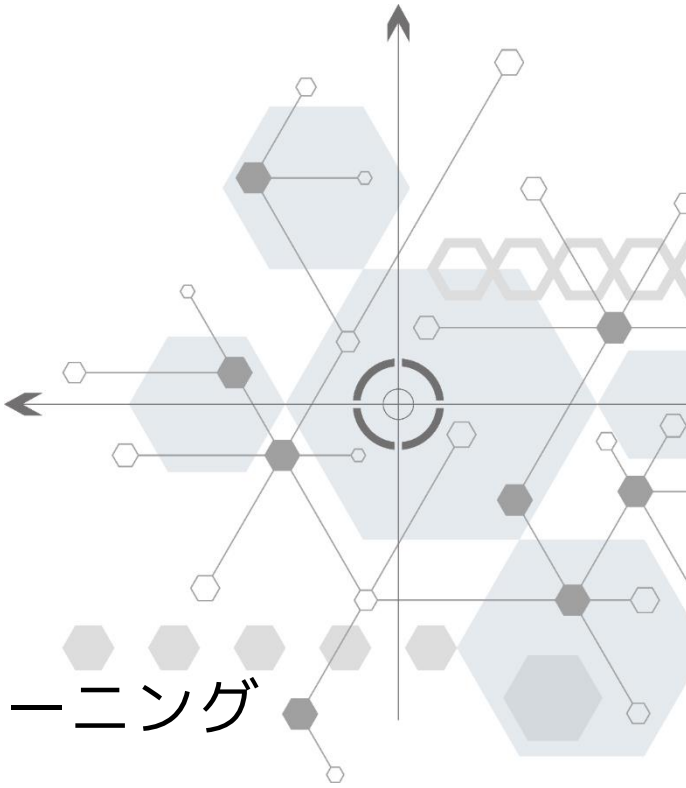
▼構文

[ ]	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

	指定バージョンからの新機能 （左記の場合、Oracle Database 23ai からの新機能）
	Enterprise Edition で使用できる機能
	注意事項
	参考情報
	データ・ディクショナリ・ビューまたは動的パフォーマンス・ビュー

本テキストは、Oracle Database 19c～23ai に対応しています。Oracle Database 23ai は、2025 年 10 月より、Oracle AI Database 26ai という名称に置き換わりました。



第 5 章
結合処理のチューニング

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。

本章のゴール

・結合の処理内容に応じて、適切な結合の種類と結合順序を判断できる。

01 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、問い合わせ文の中で結合する複数の表名と、結合条件を指定します。

■結合処理の指定方法

指定項目	ANSI 対応の結合	Oracle Database 独自の結合
結合する複数の表	FROM 句で JOIN 句を使って指定	FROM 句で「,」(カンマ) で区切って指定
結合条件	ON 句 (または USING 句) で条件を指定	WHERE 句で条件を指定

※本章では、標準 SQL である ANSI 対応の結合方法を例題に使用しています。

結合のパフォーマンスを向上させるには、最も適切な結合の種類と結合順序を選択することが重要です。

(1) 結合の種類

結合の種類には、主に以下の 3 つがあります。

・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	使用する条件
ハッシュ結合	結合結果が大量	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	大規模な表側に索引が必要

(2) 結合の順序

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。

02 結合の種類

結合には、主に 3 つの種類があり、有効なシーンはそれぞれ異なります。そのため、各結合の内部動作を踏まえて、状況に適した結合の種類を検討できることが、根拠をもってチューニングを行ううえでは重要になります。

（1）ハッシュ結合

ハッシュ結合は、大量のデータを結合する必要がある場合（または小規模な表の大きな割合のデータを結合する必要がある場合）に向いています。通常、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

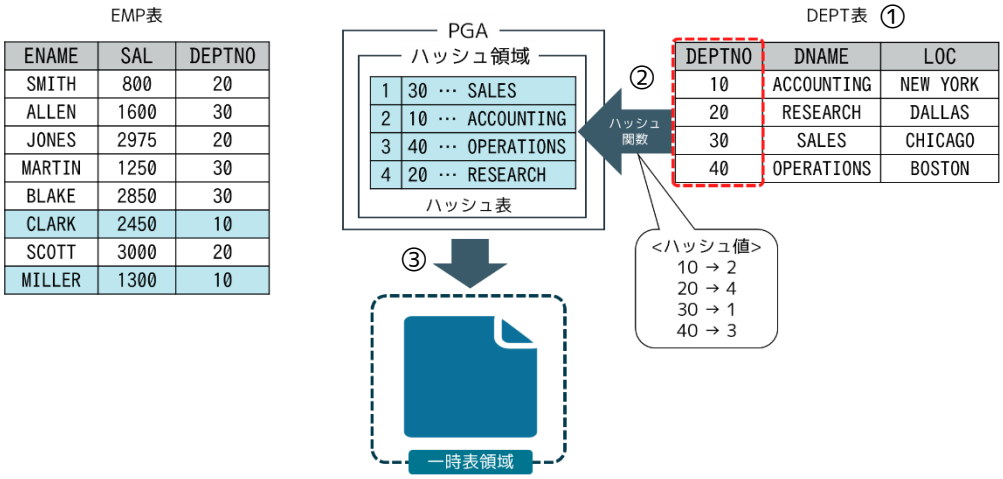
ハッシュ結合は、内部的に以下のステップで行われます。

1. 結合する 2 つの表を比較し、小さな表を選択する。
2. 選択した小さな表の結合条件列にハッシュ関数を適用し、PGA 内にハッシュ表を作成する。
   ※ハッシュ表が PGA 内に収まらない場合は、ハッシュ表の一部がディスク側の一時表領域に書き出され、パフォーマンスが低下します。そのため、小さな表をもとにハッシュ表が作成される必要があります。
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。

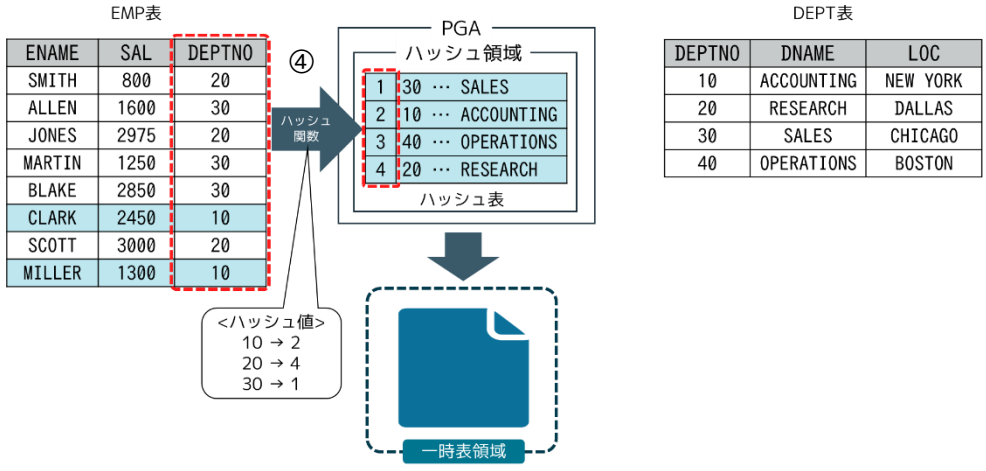
＜実行 SQL＞

```
SELECT ename,dname
FROM   emp JOIN dept
ON     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き出す。



④もう一方の EMP 表の DEPTNO 列にハッシュ関数を適用する。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、ソート/マージ結合よりハッシュ結合が選択されやすいと言えます。

・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 JOIN dept
  2  ON     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（上記例では DEPT 表）をもとにハッシュ表が作成されます。

（2）ソート/マージ結合

ソート/マージ結合は、結合条件に指定された各列をソートし、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。
※通常、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

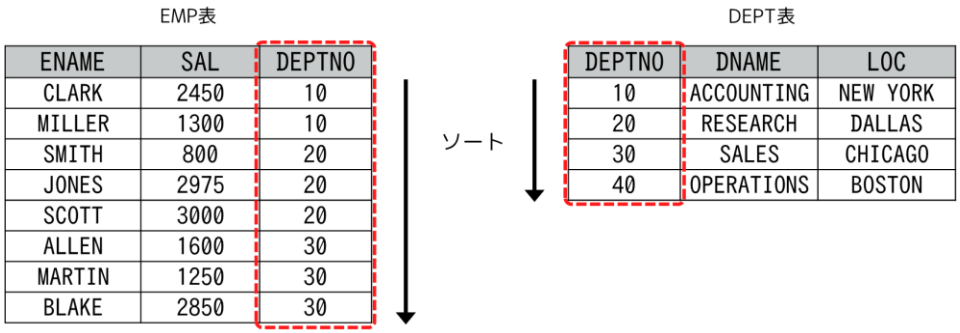
ソート/マージ結合は、内部的に以下のステップで行われます。

1. ソート操作　　：両方の表の行ソースが、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 JOIN dept
ON     emp.deptno = dept.deptno;
```

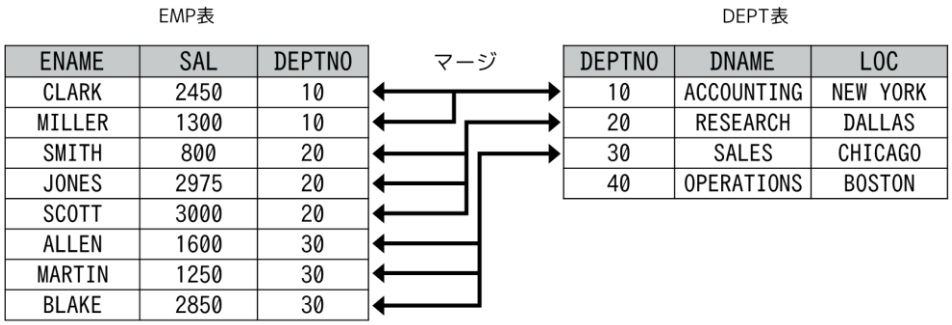
①それぞれの表の結合条件列（DEPTNO 列）をもとに行ソースをソートする。



ENAME	SAL	DEPTNO
CLARK	2450	10
MILLER	1300	10
SMITH	800	20
JONES	2975	20
SCOTT	3000	20
ALLEN	1600	30
MARTIN	1250	30
BLAKE	2850	30

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

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。



ENAME	SAL	DEPTNO
CLARK	2450	10
MILLER	1300	10
SMITH	800	20
JONES	2975	20
SCOTT	3000	20
ALLEN	1600	30
MARTIN	1250	30
BLAKE	2850	30

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

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（<、>、<=、>=）である場合
⇒ハッシュ結合は、等価条件（=）でなければ使用されません。

・行ソースがすでに他の操作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しているため、ハッシュ結合が選択された */
SQL> SELECT ename,dname FROM emp JOIN dept
  2  ON     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 JOIN dept
  2  ON     emp.deptno = dept.deptno;

実行計画
-------------------------------------------------
| Id  | Operation                    | Name       |
-------------------------------------------------
|   0 | SELECT STATEMENT             |            |
|   1 |  MERGE JOIN                  |            |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP        |
|   3 |    INDEX FULL SCAN           | IDX_DEPTNO |
|*  4 |   SORT JOIN                  |            |
|   5 |    TABLE ACCESS FULL         | DEPT       |
-------------------------------------------------
```

※結合条件列に索引が作成されたことでソート処理を省略できます。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/マージ結合が選択されています。

＜SORT JOIN＞

結合条件列をもとにソートされたことを示します。

＜MERGE JOIN＞

ソートされたそれぞれの行ソースが、結合条件列をもとにマージ結合されたことを示します。