はじめに

■コースの概要と目的

本研修では、Oracle データベースのパフォーマンス問題への対処方法として、以下の内容を説明します。

・パフォーマンス情報の収集、分析方法

・データベースの内部構造（メモリー領域、ディスク領域）のチューニング方法

■受講対象者

データベース管理者

■前提条件

Oracle の内部構造や管理操作に関する知識を有する方。

「データベース・アーキテクチャ」コース、および「データベース・マネジメント」コースを受講された方。

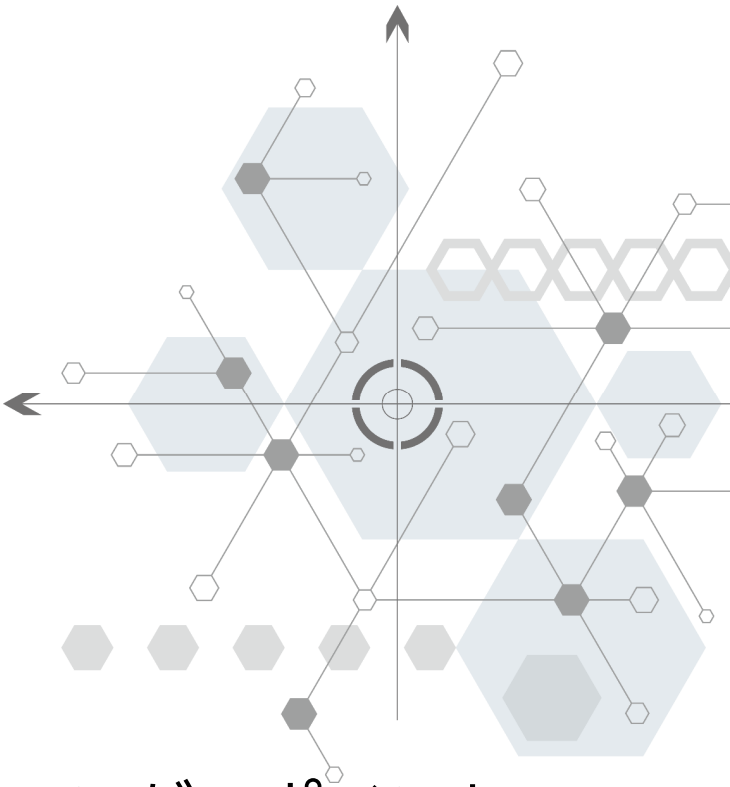
■テキスト内の記述について

▼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 19c からの新機能）
	Enterprise Edition で使用できる機能
	注意事項
	参考情報
	知っておくと便利なテクニック
	参照ページ
	データ・ディクショナリ・ビュー



第 3 章

代表的なチューニング・ポイント

Oracle データベースにおける代表的なパフォーマンス問題と、その解決方法について説明します。

01 代表的なチューニング・ポイント

Oracle データベースでパフォーマンス問題が発生している場合、主に次のような箇所に原因があります。

（1）SQL

非効率な SQL によって大量のディスク I/O やメモリー消費が引き起こされます。そのため、高負荷 SQL を特定し、適切な実行計画（アクセス・パス）が選択されるようにチューニングを行います。

例）索引を定義し、表の一部にアクセスする SQL の実行計画を全表スキャンから索引スキャンへ切り替える。

※SQL チューニングの詳細については、「SQL パフォーマンス・チューニング」コースで説明しています。

（2）メモリー

メモリーサイズが不適切な場合、ディスク I/O が多発するなどのパフォーマンス低下が起こる可能性があります。
Oracle ではシステム・グローバル領域（SGA）とプログラム・グローバル領域（PGA）を適切に管理します。

1）SGA

SGA では、主にメモリー上に保持されているデータの共有率を高めることが重要です。

・共有プール
SQL 解析結果の共有率の向上が重要です。そのため、まずは SQL の記述を統一化し、共有プールを適切なサイズに調整します。

・データベース・バッファ・キャッシュ
データベース・バッファ・キャッシュに保持されているデータの共有率の向上が重要です。そのため、SQL を改善した後に、データベース・バッファ・キャッシュを適切なサイズに調整します。

・REDO ログ・バッファ
サーバー・プロセスによる REDO レコードの書き込み待機が発生しないようにすることが重要です。
そのため、REDO ログ・バッファを適切なサイズに調整します。

Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.
株式会社 アシスト 無断転載・複製禁止

2）PGA

PGA では、ソート処理やハッシュ処理によるディスク書き込みが発生しないようにすることが重要です。
そのため、具体的にどのような操作で PGA が使用されるかを把握したうえで、大量データを操作する際に
オーバーフローが発生しないように、適切なサイズに調整します。
※PGA のチューニングの詳細については第 4 章で説明します。

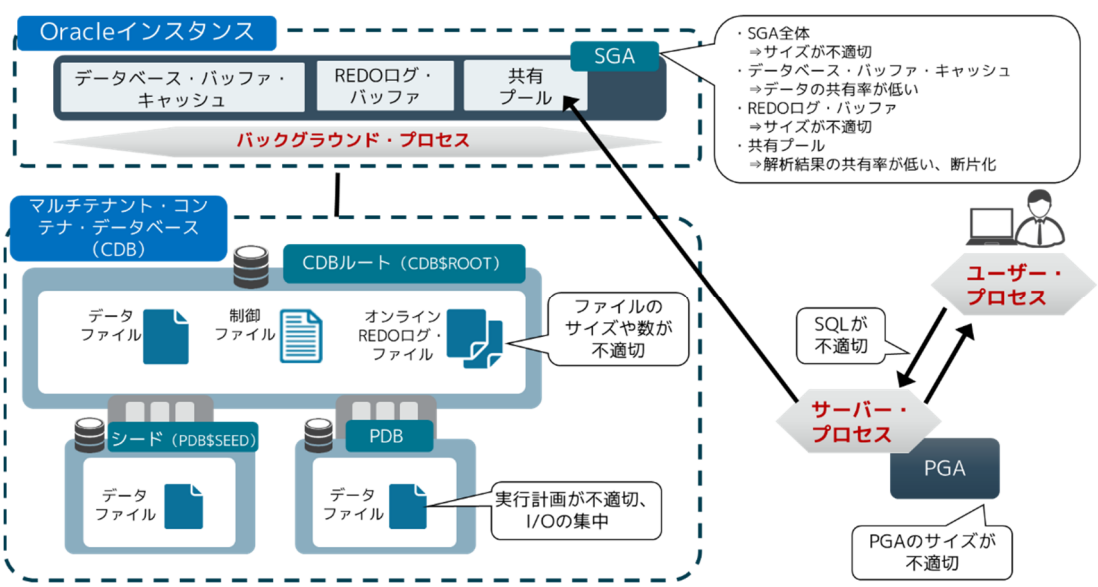
（3）ディスク I/O

ディスク I/O は Oracle の処理の中で最も時間がかかる処理です。ディスク I/O を効率化するためには、RAID を使用し I/O が 1 つのディスクに集中しないように複数のディスクに分散させます。
また、RAID を使用している場合でも、アクセス頻度の高いファイルを別のディスクに分散することで I/O 負荷を低減できます。Oracle では、データファイル、オンライン REDO ログ・ファイルを中心に管理します。

・データファイル
特定のデータファイルに対する I/O 集中が発生しないようにすることが重要です。そのため、全表スキャンを行っている SQL を索引スキャンに変更し I/O 量を削減したり、複数ディスクにアクセス頻度の高いデータを持つファイルを分散配置したりするなどの対応を行います。

・オンライン REDO ログ・ファイル
LGWR による REDO レコードの書き込み待機が発生しないようにすることが重要です。そのため、複数ディスクにオンライン REDO ログ・ファイルを分散配置したり、オンライン REDO ログ・ファイルのサイズや数を調整したりするなどの対応を行います。

＜代表的なチューニング・ポイント＞



Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.

02 共有プールのチューニング

共有プールをチューニングすると、CPU などのリソース使用量を低減できます。

（1）概要

共有プール内にはさまざまな領域が確保されますが、中でも重要な領域はライブラリ・キャッシュです。
ライブラリ・キャッシュには、SQL の解析結果やコンパイル済みの PL/SQL プログラムが格納されます。
これは、同じ SQL や PL/SQL プログラムが発行されたときに解析結果を再利用し、解析の負荷を低減することが目的です。そのため、解析結果の再利用率が高まるほど CPU リソースが節約され、パフォーマンスが向上します。
なお、ライブラリ・キャッシュのデータは使用頻度に応じて管理されているため、新しい解析結果を保存できない場合は、使用頻度の低い解析結果から削除されます。

■解析結果の利用状態
- ライブラリ・キャッシュ・ヒット（ソフト解析）
  ライブラリ・キャッシュに存在する解析結果を再利用すること。
- ライブラリ・キャッシュ・ミス（ハード解析）
  ライブラリ・キャッシュに必要な解析結果が存在せず、新しく解析を行うこと。

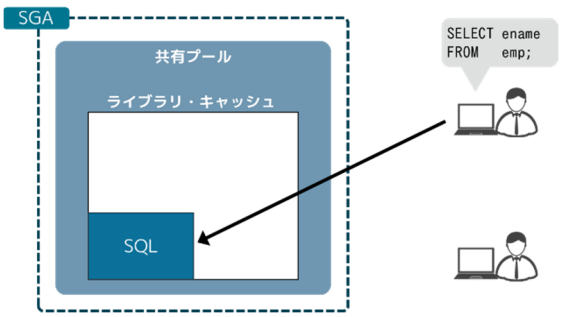
（2）発生する問題

ライブラリ・キャッシュでは、ヒット率の低下により、ハード解析によるリソース消費が増大し、CPU 負荷の高騰やラッチ競合の増加が発生します。これによりパフォーマンスが低下する可能性があります。

Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.
株式会社 アシスト 無断転載・複製禁止

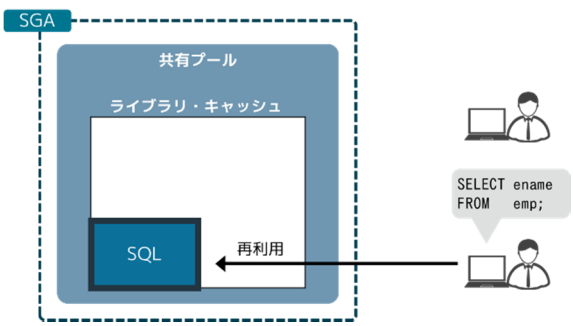
＜共有プールの内部動作＞

~ハード解析と解析結果の保持~

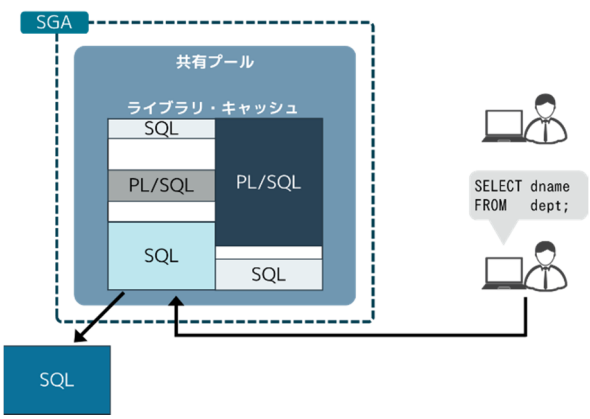


ライブラリ・キャッシュ内に共有できる解析結果が存在しなかった（ライブラリ・キャッシュ・ミスの発生）ため、ハード解析が行われる。解析結果はライブラリ・キャッシュに保持される。

~ソフト解析~



ライブラリ・キャッシュ内に共有できる解析結果が存在した（ライブラリ・キャッシュ・ヒットの発生）ため、ソフト解析が行われ、解析結果が再利用される。



ライブラリ・キャッシュがいっぱいである場合は、以下のように動作する。

①使用頻度の低い解析結果を削除する。

②新しいメモリー領域を確保し、解析結果を保持する。

株式会社 アシスト 無断転載・複製禁止
Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.

（3）調査

ライブラリ・キャッシュのヒット率を調査し、チューニングの必要性を検討します。
通常時のパフォーマンスと診断期間のパフォーマンスを比較し、ヒット率が低下している場合にはチューニングを検討します。

1）ヒット率

Statspack レポートの「Instance Efficiency Indicators」セクションの以下項目を確認します。

項目	内容	参考:目標
Library Hit %	ライブラリ・キャッシュのヒット率	95%以上
Execute to Parse %	解析なしで SQL を実行した割合	－
Parse CPU to Parse Elapsed %	解析そのものにかかった時間と解析の経過時間の割合	－
Soft Parse %	ソフト解析で処理を行えた割合	90%以上
%Non-parse CPU	解析処理以外にかかった時間とインスタンス全体の処理時間の割合	80%以上

例）Statspack レポートでライブラリ・キャッシュのヒット率を確認する。

```
Instance Efficiency Indicators
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

            Buffer Nowait %:  100.00       Redo NoWait %:  100.00
            Buffer  Hit   %:   99.60  Optimal W/A Exec %:   99.94
            Library Hit   %:   86.25        Soft Parse %:   86.34
        Execute to Parse %:   40.71         Latch Hit %:   99.82
Parse CPU to Parse Elapsd %:   92.31     % Non-Parse CPU:   99.65
```

※上記では、「Library Hit %」が 86.25%と一般目標と比較すると低いことが確認できます。

Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.
株式会社 アシスト 無断転載・複製禁止

■ライブラリ・キャッシュに関連する待機イベント

通常、ライブラリ・キャッシュのヒット率をもとに調査を行います。

あわせて、以下のようなライブラリ・キャッシュに関する主な待機イベントが上位に示される可能性があります。

待機イベント名	説明
library cache load lock	SQL の解析結果をライブラリ・キャッシュにロードするときに必要なラッチの獲得を待機。
latch: shared pool	共有プールの割り当てや解放を行うときに必要なラッチを獲得できなかった場合に発生。
cursor: mutex X cursor: mutex S	ライブラリ・キャッシュ・オブジェクト（SQL 解析結果や PL/SQL プログラムなど）にアクセスする際にラッチを獲得できなかった場合に発生。
library cache: mutex X library cache: mutex S	ライブラリ・キャッシュ内の検索や管理を行う際にラッチが獲得できなかった場合に発生。

株式会社 アシスト 無断転載・複製禁止
Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.

（4）対処

ライブラリ・キャッシュのヒット率を向上させるために、次のことを検討します。

1）SQL 記述の統一

解析結果が共有されるためには、発行される SQL が以下の条件を満たしている必要があります。
・SQL テキストの記述が空白、大文字/小文字、コメント、改行位置を含め、すべて同一であること
・参照するオブジェクトのスキーマが同一であること
・各セッションの環境が同一であること（初期化パラメータの設定がセッションで異ならない）

そのため、以下の対応を検討します。
・アプリケーション開発者間で SQL の記述ルールを標準化する
・多数のユーザーが非共有の SQL を発行しないように設計する

■バインド変数の使用
ライブラリ・キャッシュ・ミスの原因として最も多い例が、適切にバインド変数を使用していない場合です。バインド変数を使用すると、リテラルのみ異なる SQL の解析結果が共有できるようになり、ヒット率を高めることができます。

＜リテラルのみ異なる SQL＞
```
SELECT ename FROM emp WHERE empno = 7369;
SELECT ename FROM emp WHERE empno = 7499;
```
＜バインド変数を使用した SQL＞
```
SELECT ename FROM emp WHERE empno = :emp_num;
```

■CURSOR_SHARING パラメータの設定
「アプリケーションを変更できない」、「バインド変数への変更量が膨大でアプリケーション修正のコストが大きい」という場合は、CURSOR_SHARING パラメータの設定を検討します。このパラメータを設定すると、発行された SQL のリテラル部分を、Oracle がバインド変数へ自動変換して実行します。

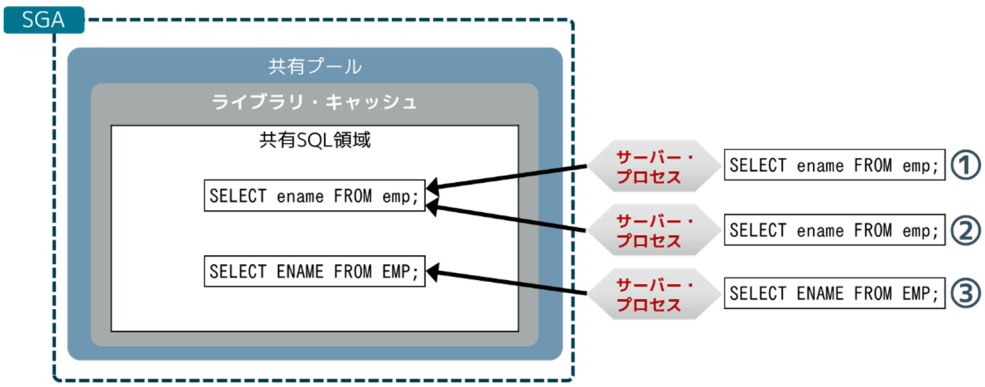
⚠ 注意事項
自動変換のオーバーヘッドが加わるため、CURSOR_SHARING パラメータはバインド変数への書き換えが困難な場合のみ、利用を検討してください。

☞「CURSOR_SHARING パラメータ」（A-9）

Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.　株式会社 アシスト 無断転載・複製禁止

＜ライブラリ・キャッシュのチューニング（ヒット率の向上）＞

～SQL の記述～

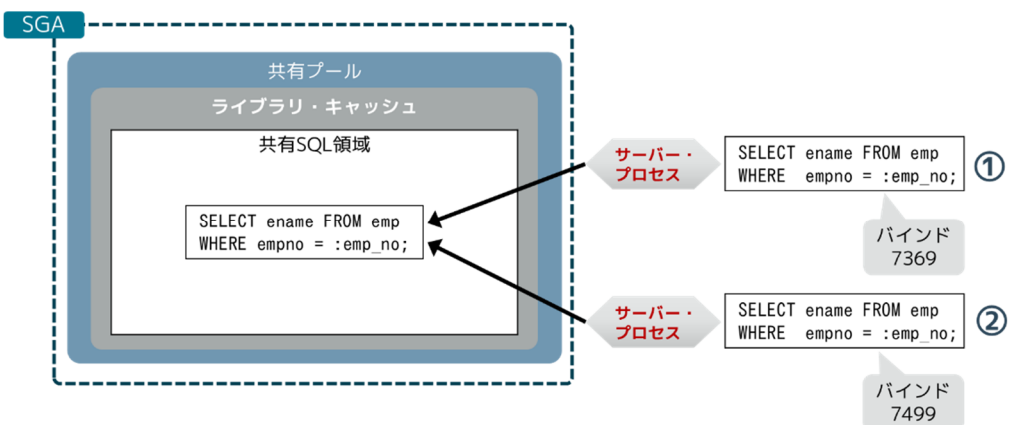


①初めて実行する SQL はハード解析が行われ、解析結果が保持される。

②2 番目に実行する SQL は事前に同一 SQL が実行されているため、ソフト解析が行われる。

③3 番目に実行する SQL は、SQL の記述が異なるため、同一 SQL と見なされず、ハード解析が行われる。

～バインド変数の使用～



①初めて実行する SQL はハード解析が行われた後に、バインド変数に「7369」を代入し、実行される。

②2 番目に実行する SQL も、バインド変数に値を代入する前に解析が行われる。そのため、ソフト解析が行われる。

株式会社 アシスト 無断転載・複製禁止
Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.

2）共有プールのサイズ変更

SQL を適切に記述してもヒット率が向上しない場合は、共有プールのサイズが小さすぎる可能性があります。そのため、共有プールのサイズ増加を検討します。

■サイズの設定

自動共有メモリー管理（SGA 全体）や自動メモリー管理（Oracle 全体）を使用し、共有プールのサイズを設定します。

※SHARED_POOL_SIZE パラメータで共有プールのサイズを個別に設定することも可能です。なお、ライブラリ・キャッシュなどの共有プール内の領域は、Oracle により自動調整されるため、個別に調整することはできません

「自動共有メモリー管理」（4-3）
「自動メモリー管理」（4-13）

■適切なサイズ調整

メモリー・アドバイザ機能を使用すると、共有プールや SGA サイズを変更した際の動作に関する予測値を確認できるため、サイズ調整が容易になります。

「適切なメモリーサイズに調整」（4-8）

3）オブジェクトの固定

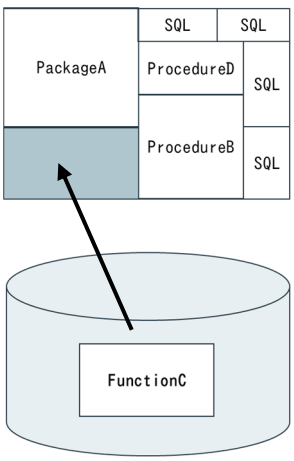
共有メモリーには、SQL の解析結果だけでなく PL/SQL プログラムもロードされます。
一度ロードされたプログラムは、再利用に備えて実行後もメモリー上に保存されますが、メモリー領域が足りない場合は、保存されたプログラムを消去してその領域を使用します。これが繰り返されると、サイズの大きな PL/SQL プログラムが頻繁にロードされ、パフォーマンスが低下することがあります。
このような事態を防止するため、DBMS_SHARED_POOL パッケージを使用して、使用頻度が高く、サイズが大きな PL/SQL プログラムをメモリー上に固定すると効果的です。

「DBMS_SHARED_POOL パッケージ」（A-13）

Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.
株式会社 アシスト 無断転載・複製禁止

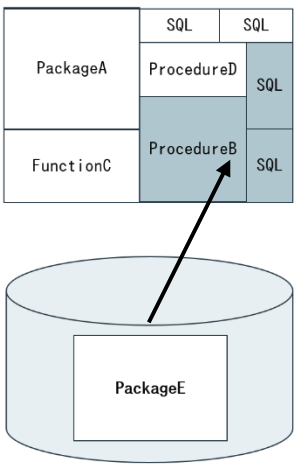
＜オブジェクトの固定＞

共有メモリー



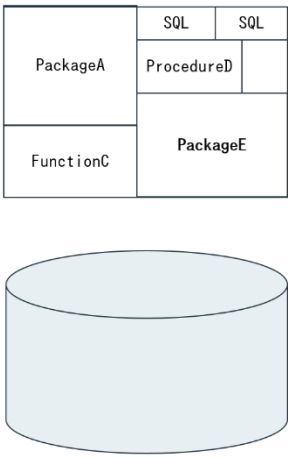
メモリー上にまだ空きがある場合、実行するプログラムは空いている領域にロードされる。

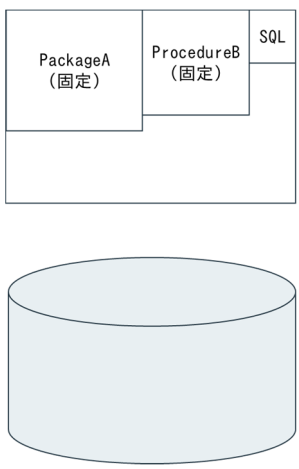
共有メモリー



メモリーに空きがない場合、空きを作るために使用頻度の低いものをメモリー上から消去し、用意した空き領域にプログラムをロードする。

共有メモリー





使用頻度が高く、サイズが大きな PL/SQL プログラムをメモリー上に固定する。

共有プールにはライブラリ・キャッシュとディクショナリ・キャッシュという領域があります。
通常はライブラリ・キャッシュへの対処を行うことで、ディクショナリ・キャッシュも間接的にチューニングされます。

株式会社 アシスト 無断転載・複製禁止
Copyright (C) 2020 K.K.Ashisuto All Rights Reserved.