SELECT 文の記述の変更が出来ない処理に対しての実行計画の変更方法

通常の SQL 文のチューニングでは、SQL 文の Where 条件などの記述を変更したり、ヒント句を加えて、オプティマイザが作成する実行計画に変更が行われるような方法を取ります

しかし、パッケージ製品などのようにソース・コードが公開されていないものに対しては、SQL 文の記述が変更できません

このような時に行うのが、『SQL 文の記述を変更せずに処理を効率化する』という対処で、その必要な背景が前文です

Select 文の記述を変更せずに処理を効率化するとは、

Oracle の内部動作を変更するようなシステム環境の変更やパラメータ指定を行い、実行処理の方法やオプティマイザによる実行計画作成結果を変更させるということである

このようにすれば、Select 文の記述を変更せずに、Oracle 処理の内容が内部で変更させて、現在の通常状態とは異なった動作で処理が行われることになる

Select 文の記述を変更せずに処理を効率化するチューニング方法の種類

- 初期化パラメータの変更
- 索引の追加、削除
- 統計情報取得時のサンプリング・サイズの変更
- 列統計の取得
- 動的サンプリングの取得
- プラン・スタビリティの使用
- 問合せのパラレル化
- ヒント付ビュー
- マテリアライズド・ビューの使用

・初期化パラメータの変更

sga_target 初期化パラメータ

自動共有メモリ管理機能を使用した SGA メモリ全体へのメモリ割当て量

キャッシュ・ヒット率の向上によるディスク・アクセス I/O 処理を減少させる

pga_aggregate_target 初期化パラメータ

自動 PGA 管理機能使用した PGA メモリ全体へのメモリ割当て量

テーブル結合やソート処理をメモリ上で完結させて、一時表領域ディスク・アクセスを減少させる

・索引の追加、削除

索引追加

索引を使った Select 処理によるディスク I/O のアクセスブロック数の減少

索引削除

索引効果が低いテーブルに対して、全件レコードのマルチブロック Read を使ってのディスク I/O 処理時間の短時間化

・統計情報取得時のサンプリング・サイズの変更

オブジェクトの統計情報を事前に収集しておくことにより、実際のデータに基づいたディスク I/O **アクセス数が少なくなるような最適な**実行計画を作成することになる

注 意）

サンプリング数が少ないと、実際のデータとは懸け離れてしまい、また単純に 100%を指定しまうと、データ件数が多いオブジェクトの場合には、統計情報の取得処理自体の負荷が重くなり、悪影響を及ぼすことがある

よって、精度が上がるようなサンプリング件数割合を指定してから、実行計画を作成させることが、理想的なチューニング作業です

例）　【列の値の種類数の収集】

analyze table テーブル名 estimate statistics sample <n> percent ;

↑ 列の値の種類数の統計情報を収集させる

↑ サンプリング件数割合%

これにより、オプティマイザが、列値の種類数の統計情報を考慮した実行計画を作成することになる

(使用例)

analyze table kozue.emp estimate statistics sample 20 percent ;

確認処理

select column_name , num_distinct from dba_tab_columns
where owner = 'KOZUE' and table_name = 'EMP' ;

COLUMN_NAME	NUM_DISTINCT
EMPNO	28
ENAME	9
GROUP_NAME	5

誤差が生じるが、どこまで追求するかが重要

実際の列値の種類数

select count(distinct (EMPNO)) from kozue.emp ;

COUNT(DISTINCT(EMPNO))
22

・列統計の取得

列の値の分布であるヒストグラムを収集しておくと、Where 条件中の列で、レコード数が少ない値での検索では、インデックスを使用したテーブルアクセスを使用し、レコード件数が多い値での検索では、テーブルフルアクセスを行うような実行計画を、オプティマイザが作成する

例) 【列の値の種類数の収集】

analyze table テーブル名 compute statistics for columns <列名>
size <n> ;

サンプリング件数割合% ヒストグラムの作成の統計情報を収集させる

これにより、オプティマイザが、列値の種類数の統計情報を考慮した実行計画を作成することになる

(使用例)

analyze table kozue.emp compute statistics for columns
group_name size 10 ;

・動的サンプリングの取得

optimizer_dynamic_sampling 初期化パラメータの変更

SQL 文の処理が発生したときに、常にオブジェクトに対してサンプルを行ってから、オプティマイザが実行計画を作成するような動作にする
これによって、実行計画は実データの値に即した最適化実行計画が作成される
ただし、サンプリング処理のための負荷が、Oracle にとってのデメリットになる

```
show  parameter  optimizer_dynamic_sampling
```

NAME	TYPE	VALUE
optimizer_dynamic_sampling	integer	2

（補足情報）
SQL 文の処理がパッケージ・プロシージャを使ったもので、プロシージャの暗号化等で、SQL 文の変更が出来ないようなものであれば、Alter Session 文を使ったセッション限定での初期化パラメータの変更も可能である

```
alter  session  set  optimizer_dynamic_sampling = 1;
```

・プラン・スタビリティの使用

事前に、『過去に作成されていた最適な SQL 文の実行計画』や『ヒント句を加えた SQL 文の実行計画』をプラン・スタビリティとして保存しておく

そして、それをパッケージ中の SQL 文の実行の時に使用する実行計画として指定する

こうすることによって、パッケージ中の変更できない SQL 文に対して、意図したテーブルへのアクセス方法（アクセス・パス[Access Path]、結合順序[Join Order]、結合方法[Join Method]）を使用する実行計画に変更することが出来ます

・問合せのパラレル化

CPU リソースに余裕がある環境では、問い合わせ処理を**パラレル化して並列処理**で行うと時間短縮ができる
しかし、CPU リソースに余裕がない場合には、オーバーヘッド負荷が加わってかえってパフォーマンスが低下するので使用しないこと

パラレル化実行の方法としては、「セッションに対して行う方法」と「テーブルの定義を変更して行う方法」の２種類がある

SQL 文の処理がパッケージ・プロシージャを使ったもので、プロシージャの暗号化等で、SQL 文の変更が出来ないようなものであれば、処理を行うセッションに対して Alter　Session 文を使って、セッション限定での初期化パラメータを使用したパラレル化の平行処理が可能ある

これに対して、パッケージ・アプリケーションのようにセッションに対しても何も変更が出来ないものに対しては、テーブル定義に対してのパラレル化指定が有効である
ただし、テーブル定義に対してのパラレル化指定は、全処理に対して反映してしまうので、全体の負荷も上がってしまう欠点がある

【セッションに対して行う方法】
alter　session　force　parallel　dml　parallel　2；

【テーブルの定義を変更して行う方法】
alter　table　テーブル名　parallel　<n>　；

・ヒント付ビュー

パッケージの中でビューを使って、レコードにアクセスしている SQL 文処理においては、**同一のビュー名**でヒント付きビュー、もしくはマテリアライズド・ビューを作成して**置き換え**てしまえば、効率的な SQL 処理に変更することが出来る

注）元のビューと名前を同一にするためには、元のビューを削除しておく必要がある

作成方法

```
create　view　ビュー名　as
    select　/*+　ヒント句　*/　列名 1,　列名 2,　・・・
      from　テーブル名 1,　テーブル名 2
      where　テーブル名 1.列名 1 = テーブル名 2.列名 1　；
```

作成例）

ソートマージ結合の使用を明示的に指定したヒント指示ビュー

```
create　view　ビュー名　as
    select　/*+　ordered　use_merge( d )　*/　列名 1,　列名 2,　・・・
      from　テーブル名 1,　テーブル名 2
      where　テーブル名 1.列名 1 = テーブル名 2.列名 1　；
```

・マテリアライズド・ビュー

同　　上