ORACLE
E-BUSINESS SUITE

Oracleホワイト・ペーパー
2013年9月

# Oracle E-Business Suiteにおける統計収集のベスト・プラクティス

ORACLE

## 免責事項

以下の内容は弊社の全般的な製品の方向性を示すものです。情報提供のみを目的としており、なんらかの契約に組み込むことはできません。マテリアルやコード、機能の提供を確約するものではなく、購買を決定する際の判断材料になさらないでください。本書に記載されているオラクル社製品の機能の開発、リリースおよびタイミングは、オラクル社の裁量により決定されます。

## エグゼクティブ概要

このペーパーは統計収集のベスト・プラクティスについて説明したもので、Oracle E-Business Suite 11i (11.5.10以降)およびリリース12に適用可能なFND_STATSパッケージに実装されている新機能に注目します。

## はじめに

『オプティマイザ統計の収集のベスト・プラクティス』ホワイト・ペーパーではOracleデータベースの統計収集プロセスに関する広範な概要が示されており、このペーパーの前に見ておくと役立ちます。そのなかでは統計の収集および設定のためのいくつかの方法が説明されていますが、Oracle E-Business Suiteでサポートされているのは「統計の収集」コンカレント・プログラムまたはFND_STATSパッケージ(これは「統計の収集」コンカレント・プログラムにより呼び出されます)のいずれかのみです。FND_STATSはDBMS_STATSパッケージのラッパーで、表やスキーマ、データベース・レベルで統計を収集するためのいくつかのPL/SQLプロシージャを提供します。FND_STATSパッケージではヒストグラム、表の除外、および新しいDatabase 11Gの機能(拡張統計、パーティション表のための増分統計収集およびコンカレント統計収集など)もサポートされています。DBMS_STATSパッケージおよび廃止されたANALYZEコマンドのいずれもこれらの機能は認識せず、結果として最適化されていない実行計画となる可能性があります。

## FND_STATSの機能

このセクションでは、FND_STATSパッケージがDBMS_STATSとどのようにやりとりするかの概要を説明します。Oracle E-Business SuiteとDBMS_STATSのいずれも、統計が失効しているか空である(欠落している)オブジェクトの統計を収集しますが、FND_STATSパッケージはOracle E-Business Suiteに次の機能を実装します。

- 個々のスキーマ、またはOracle E-Business Suite全体の統計の収集をサポートします。すべてのE-Businessスキーマを処理するには、スキーマ名として'ALL'を渡します。DBMS_STATSにはこの機能はありません。
- FND_HISTOGRAM_COLSに指定された列のヒストグラム。これには、Oracle Developmentが必須として指定したヒストグラム列のリストが格納されます。
- 新しいGATHER_AUTOオプションでは、統計が失効または欠落している(空である)表の統計の収集を一括で行います。
- FND_STATSでは再実行機能がサポートされています。*「スキーマ統計の収集」*コンカレント・プログラムが使用されている場合、FND_STATSでは各発行の履歴が保持されます。実行が停止または中断した場合、次の発行は失敗または終了したところから再開されます。データベースのバージョン10gから、再実行機能のために表モニター機能を備えたDBMS_STATS 'GATHER AUTO'オプションを使用できます。
- FND_STATS_HISTには、統計収集パッケージFND_STATSが統計を収集するのにかかった時間に関する履歴情報が格納されます。

## 履歴モード

*「表統計の収集」*プログラムではオプションで、新しい統計を収集する前に既存の統計をバックアップできます。バックアップ・フラグ・パラメータの値がBACKUPの場合、FND_STATSでは新しい統計を収集する前にDBMS_STATS.EXPORT_TABLE_STATSを使用して古い統計をエクスポートし、FND_STATTABに格納します。

履歴モード・パラメータでは、作成される履歴レコードの量を制御できます。

- **LASTRUN**: 最後に実行された統計の収集の履歴レコードのみが保持されます。収集が実行されるたびに、オブジェクトのその前の履歴レコードが上書きされます。これはデフォルトの動作です。
- **FULL**: このモードではそれまでの履歴情報は上書きされません。各新規実行に対して履歴レコードが作成され、要求IDによって識別されます。この要求IDは、指定がなければ自動的に生成されます。このモードを使用する場合、「FND_STATS履歴レコードのパージ」コンカレント・プログラムを定期的に実行してFND_STATS_HIST表をパージする必要があります。
- **NONE**: このモードでは履歴情報は生成されず、実行は再開できません。

統計の収集後に問題が発生した場合、RESTORE_SCHEMA_STATSまたはRESTORE_TABLE_STATSプロシージャを使用して、FND_STATTAB表に以前にバックアップされた統計をディクショナリに復元できます。*statid*を使用して、それぞれの統計のセットを識別できます。

## 収集オプション

統計を増分収集するには、GATHER_AUTOオプションを使用します。新しいまたは変更されたオブジェクトの統計についての情報を収集するに際し、GATHER_AUTOはOracleの表モニター機能を利用します。

Oracle 9iでは表を個別に指定することが可能でしたが、このアプローチはその後のリリースで使用不可となり、Oracle 10gおよびOracle 11gではデフォルトで、スキーマ全体に対して自動的に有効化されるようになりました。これと同様の動作をOracle 9iで実現するには、次のプロシージャを使用します。

```
exec FND_STATS.ENABLE_SCHEMA_MONITORING (‘ALL’);
または、単一のスキーマに対してモニタリングを有効にする場合:
exec FND_STATS.ENABLE_SCHEMA_MONITORING (SCHEMA_NAME);
```

「収集オプション」パラメータが"GATHER AUTO"に設定されている場合、FND_STATSでは、表が最後に分析されてから(統計が収集されてから)の変更量が変更しきい値%を超えている表についてのみ統計を取得します。変更しきい値はmodpercentに値を渡すことでユーザーが調整でき、デフォルトは10となっています。そのため、GATHER_AUTOでは10%を超える変更量がDBA_TAB_MODIFICATIONSに示されている表についての統計のみが収集されます。

```
FND_STATS.gather_schema_statistics ( .. ,options=>’ GATHER AUTO’);
```

## ヒストグラム

ヒストグラムは特別なタイプの列統計で、列内の値の頻度や分布についての追加情報を提供します。この情報はオプティマイザにより、たとえば索引の使用や全表走査の実行の決定、結合順序の決定の際に使用されます。ヒストグラムがない場合、オプティマイザでは列の個別値全体に対して均一に行が配分されると仮定します。これらは*where句*で参照されている列の値の数が不均衡で、索引を使用するよりも全表スキャンを実行するほうが通常は安価となる場合、言い換えると列に一意でない繰り返しキーがあり、異なる値がわずかしかない場合にOracle E-Business Suiteにおいて作成されます。これはデータ・スキューと呼ばれます。ヒストグラムは一部のjoin述語に対しても使用でき、これはカーディナリティの見積に役立ちます。Oracle 11gでは、適応カーソル共有を使用することにより、バインド・ピーキングで正しい計画が使用されるようにします。

ヒストグラムの統計を収集すると、プロセス全体にかかる時間が長くなります。ヒストグラムは問題を招く可能性があり、Oracle 10gではバインド・ピーキングおよびそれがカーディナリティの見積にどう影響するかについての懸念がありました。オプティマイザは、SQL文の最初のハード解析でバインド変数の値を調べ、その後のすべての実行においてその計画が使用されます。Oracle E-Business Suiteでの唯一の現実的な解決策はバインド・ピーキングを無効化することで、ヒストグラムを削除するという選択肢はありませんでした。

Oracle 11gでは、適応カーソル共有の導入によりこれが改善されました。ヒストグラムを使用しないことにより、SQLの実行計画がバインド値によって変わることはなくなります。

カスタム開発が可能であれば、FND_STATS.CHECK_HISTOGRAM_COLSを使用することにより、提示されたヒストグラムの実行可能性を判断できます。このプロシージャでは、一意でない索引の先頭の列で単一の値がサンプルの1/75以上を占めるかどうかをチェックします。行の数は3000以上あることが推奨されます。次の例は、このプロシージャを実行する方法を示します。

```
Set Serveroutput on
EXEC FND_STATS.CHECK_HISTOGRAM_COLS( tablelist => ‘owner.table_name1 , owner.table_name2, ....’ )
```

設計およびパフォーマンスをチェックしたら、LOAD_HISTOGRAM_COLSプロシージャを使用してFND_HISTOGRAM_COLS表に情報をシードします。これにより、次の構文を使用して統計を収集するときに確実に更新されます。

```
begin
 FND_STATS.LOAD_HISTOGRAM_COLS
   (action=>'INSERT', appl_id=>&custom_application_id, tabname=>&table_name, colname=>&column_name);
 FND_STATS.GATHER_TABLE_STATS( ownname=>&owner_name, tabname=>&table_name);
 end;
 /
```

## AUTOサンプリング・オプション

Oracle E-Business Suiteでは伝統的に、統計の大部分は、ESTIMATE_PERCENTパラメータで指定されているデフォルトのサンプリング・サイズである10%で収集されていました。経験によると、データ・スキューが発生しがちな一部の製品ではより高いパーセンテージで収集することが有効です。これにはトレードオフがあります。サンプル・サイズが大きくなれば結果がいっそう正確になりますが、収集にかかる時間が長くなります。サンプル・サイズが小さくなれば収集にかかる時間は短くなりますが、データの実際の分布を反映した統計を収集できるとは限らず、最適なランタイム・パフォーマンスが(とりわけデータ・スキューがある場合に)得られなくなります。

Oracle 10gのDBMS_STATSでは、ESTIMATE_PERCENTパラメータのデフォルト値がDBMS_STATS.AUTO_SAMPLE_SIZEに変更されました。これによってOracle側で適切なサンプル・サイズを決定できるようになりましたが、表に極端なデータ・スキューがあった場合に不適切なサンプルによる問題が発生していました。Oracle 11gでは、精度が100%に近く、かつ古い方法で10%を収集したときよりも速く実行される、ハッシュベースのサンプリングが使用されるようになりました。

新しいバージョンのFND_STATSではESTIMATE_PERCENTパラメータの'AUTO'オプションがサポートされるようになり、これによりDBMS_STATS.AUTO_SAMPLE_SIZEが使用されます。

**注意: パーセントを数字で指定した場合、たとえ100%に指定したとしても古い統計収集アルゴリズムが使用されるため、AUTOを使用することをお薦めします。**

サンプル・サイズにパーセントを具体的に指定するのではなくAUTOを使用することには、他にも次のようなメリットがあります。

- システムの成長やパージといったデータ量の変化に対応できます。一方、ある段階では適切だった固定のサンプリング・パーセントも、時間の経過に伴い定期的にテストやレビューが必要となります。
- これまでの経験上、ほとんどのオブジェクトについては10%のサンプルで十分ですが、一部の製品/表については、たいていはデータ・スキューにより、30% - 40%という高いサンプリング・パーセンテージが適切となります。どの製品に適切だったか、あるいはどのサンプル・サイズなら最短時間で最適な結果が得られたかを判断する必要がなくなりました。

**注意: Oracle E-Business SuiteのAUTOサンプリング機能は、Oracle 11gにおいて記述された場合にのみ動作します。それより前のデータベース・バージョンで使用すると、*estimate_percent*パラメータに対して10%が使用されます。**

**注意: 「統計の収集」コンカレント・プログラムを起動する際には、この見積パーセント・パラメータは空白のままにしておいてください。このプログラムでは(データベース・バージョンに基づいて) estimate_percentパラメータに対して自動的にデフォルト値が選択されます。値を指定すると、その指定されたパーセンテージで統計が収集されます。データベース・バージョンが11g以上の場合、このパラメータのデフォルト値はdbms_stats.auto_sample_sizeとなり、それより前のリリースの場合は10%に設定されます。**

**注意: Oracle Database 11gのAUTOサンプリング統計収集機能は、FND_STATSの最新のコードに実装されています。「スタンドアロン・パッチ」セクションを参照してください。Oracle 11gのスタンドアロン・パッチが表1に示されています。**

## カスタム・スクリプト

「統計の収集」コンカレント・プログラムの使用に代わる方法として、FND_STATSパッケージを直接呼び出すカスタム・スクリプトを開発できます。構文は次のとおりです。

スキーマ統計の場合: EXEC FND_STATS.gather_schema_statistics('schema_name');

例:

EXEC FND_STATS.gather_schema_statistics('ALL');

EXEC FND_STATS.gather_schema_statistics('XLA');

表統計の場合: EXEC FND_STATS.gather_table_stats(‘owner’,’table_name’);

例:

EXEC FND_STATS.gather_table_stats('AR','RA_CUSTOMER_TRX_ALL');

## 統計を収集するタイミング

一部のOracle E-Business Suiteの顧客は、あまりにも頻繁に統計を収集して無駄にシステム・リソースを浪費しています。代表的な統計を得るとは、頻繁に(たとえば毎日)「統計の収集」を実行するという意味ではありません。そうではなく、収集頻度によりSQL実行計画が目まぐるしく変わることがないように戦略を立てるとともに、その戦略が陳腐化してパフォーマンスに影響が出ることがないようにしてください。大規模なデータ・インポートの直後に特定のオブジェクトのセットの統計を収集するなどの例外もあります。これは、通常はOracle E-Business Suiteインタフェース・プログラム内で自動化されています。

Oracle E-Business Suiteの顧客は、統計を収集する頻度をデータまたはデータ・スキューの実際の変更に基づいて、言い換えると時間ではなく量に基づいて決定するように薦められています。統計は実際のデータに基づいて、隔週、毎月、あるいはさらに長い頻度で収集することをお薦めします。

1000万行が含まれる表に毎月100万件のレコードが追加される例を考えてみましょう。これは、失効した統計を収集するときに使用される10%のしきい値に対応し、したがって統計はだいたい月に1回収集されます。10%の失効しきい値をトリガーする量の変動により、統計収集プロセスの更新の頻度が上がったり下がったりする場合があります。

この収集間隔をトランザクション・スループットのモニタリングによって決定することもできます。クリティカルなビジネス・プロセスおよびトランザクションの反復可能な基本編成を作成します。この基本編成を、キー・トランザクション・パフォーマンスに変化が見られるようになるまで定期的に(毎週や毎月など)レビューします。ビジネス・サイクルおよび大規模なデータ変更(転記、月末処理、データ・インポート、パージなど)をマップします。この情報を、収集頻度の決定に役立てます。

**注意: クリティカルなビジネス・プロセスおよびトランザクションの基本編成は基準となり、これはパッチ適用やシステムの更新の際のパフォーマンス・モニタリングにとりわけ役立ちます。**

時間に制約があり、統計の収集にかかる時間が収集ウィンドウを超過する場合、スキーマごと、または場合によっては個々の表ごとに時間帯で収集プロセスを分割する必要があります。後者のアプローチでは、一般にサイズが小さい、変動が少ない表および参照表は失効する可能性が低いため、より変動の激しい高トランザクションの表に注力することがいっそう重要になります。

**注意: 収集の際に、Oracleでは統計が必要なデータベース・オブジェクトの優先順位付けを内部的に行っています。**

管理者が適切な戦略を判別するうえで、USER_TAB_STATISTICSのSTALE_STATS列をモニターするのが有効となる場合があります。しかし、この情報は1日に1回更新されるのみであるため、パージやデータ・インポートといった主要なアクティビティについてのビジネス・ナレッジがさらに必要となるか、または挿入、更新および削除の数がリストされているUSER_TAB_MODIFICATIONSからの情報を追加で参照する必要があります。

総じて、ビジネス・サイクル中の特定の曜日や日付、時間に統計を収集することにより、管理者は統計イベントの収集を予期しない動作と関連付けることができます。すべてのスキーマについて毎晩や毎週のように過度に統計を収集することはしないでください。

## カーソルの無効化

オブジェクトの統計がリフレッシュされると、共有プールやライブラリ・キャッシュ内のオブジェクトへのすべての参照が無効となり、そのオブジェクトを参照するすべての依存カーソルも無効となり、そのたびに新しくハード解析が必要となります。Oracleは無効化が必要な依存オブジェクトに対して無効化タイムスタンプを配布するため、同時に大量のハード解析が行われる可能性は高くありませんが、それでも、特に大量のオブジェクトに帯する解析を収集する際にライブラリ・キャッシュ待ちが増えることがあります。

**注意: 「統計の収集」コンカレント・プログラムを使用する際に、「依存カーソルの無効化」パラメータはデフォルトで「Y」に設定されており、これは標準のベスト・プラクティスです。カーソルが消去されて再ロードされると必ず、新しい実行計画が生成されます。**

このアプローチはほとんどのOracle E-Business Suiteインストレーションに適していますが、このパラメータでカーソルを無効化しないように指定することもできます。これによりCPU利用率が下がることがありますが、これは、依存カーソルが消去されるまで、最適化されていない可能性がある既存の計画が引き続き使用されることを意味します。

これら2つの無効化オプションのどちらを選択するかは収集の頻度によって決まります。統計をほぼ最新に保っている場合、新しく収集された統計からの計画が大きく異なる可能性は低いため、カーソルを無効にする必要はないでしょう。言い換えると、パフォーマンスは時間が経過しても変わらず安定します。しかし、統計を収集する頻度が十分でなかったり、統計の失効が原因でパフォーマンスが低下するようになった場合は、できるだけ早くこれをデフォルトに設定してターゲット・オブジェクトに依存するカーソルを無効化することで、計画がより早くリフレッシュされます。

**注意: 統計の収集は、システム・アクティビティがほとんどないとき、または定義済のバッチ期間中にのみ行ってください。**

## 統計のロック

ロックにより、収集プロセスの実行時にオブジェクト統計が上書きされるのを防ぐことができます。これは、収集プロセスの実行時に表が空である場合や、例外的に大きな表を除外してスキーマの残りについて統計を収集するという場合に役立ちます。

次の場合に統計をロックすることを検討してください。

- 代表的な統計がすでにあり、ほとんど更新されない**非常に大きく、変動が少ない表**。
- 時間の経過とともにデータが劇的に変更される**変動の激しい表**。そのよい例が、バッチ・プロセスにおいて使用される中間表や、Oracle E-business Suiteで幅広く配置される一部のインタフェース表です。アプリケーション表の例としては、AP_SELECTED_INVOICESやWSH_PR_WORKERSがあります。
- 実行時にのみデータが含まれるものの、収集プロセスの実行時には通常は空の**一時表または暫定表**。このケースでは、統計をロックしないと統計がゼロになるかまたは代表的でない統計となります。これにはグローバル一時表は含まれません。グローバル一時表はFND_STATSスキーマ統計の収集からは除外されます。

  **注意: 表の統計がゼロ(num_rows = 0)の場合、CBOではこれらの統計を使用して実行計画が決定され、実行時に表に多数の行がある場合、この実行計画は非常に非効率になります。表に空の統計(null)がある場合は動的サンプリングが使用され、これは正しくないまたはゼロの統計でCBOを使用するよりも望ましいです。**

表統計を変更から保護するには2種類の方法があります。

1. DBMS_STATS.LOCK_TABLE_STATSを使用して統計を物理的にロックし、これによりFND_STATSまたはDBMS_STATSの呼び出しによる統計の変更を防ぎます。

2. FND_STATS.LOAD_XCLUD_TABを使用して除外表のリストに表を追加することもできます。これにより、スキーマ統計の収集を使用したときに表の統計が変更されるのを防ぐことができます。ただし、スキーマ統計の収集で考慮されるのはDBMSロックのみで、FND除外は考慮されません。これは意図的な動作です。手動で統計を収集する必要があることもあるためです。

Oracle E-Business Suite開発チームは、LOAD_XCLUD_TABを使用して例外をFND_EXCLUDE_TABLE_STATS表(これはメタデータ・コンテナです)にシードしています。EBSの変動の激しい表および一時表の多くは、参照用にのみこの表にシードされています。これらには負のapplication_idがシードされているため、FND_STATSによるスキーマ統計の収集からは除外されません。これらの表はロック対象とすることを検討してください。

**注意: FND_EXCLUDE_TABLE_STATSにシード表を使用するか、統計を変更しない場合はDBMS_STATS.LOCK_TABLE_STATSを使用することをお薦めします。**
**ロックした変動の激しい表または一時表についてはそれぞれ、最初に空の(nullの)統計を設定して動的サンプリングに任せる必要があります。すでに値が含まれている場合は統計を削除してください。パフォーマンスに問題がある場合のみ、表に代表的なロードがある場合は統計を収集するか、またはFND_STATS.SET_TABLE_STATSを使用して移入することを考慮します。**

DBMS_STATS.LOCK_TABLE_STATSを使用して統計がロックされると、「スキーマ統計の収集」コンカレント・プログラムにより、当該の表において統計がロックされていることを明示的に示すメッセージが要求ログ・ファイルに表示されます。

次を使用してFND_EXCLUDE_TABLE_STATSに表をシードします。

```
FND_STATS.LOAD_XCLUD_TAB(action=>'INSERT', appl_id=>&application_id, tabname=>&table_name)
例
        FND_STATS.LOAD_XCLUD_TAB(action=>'INSERT',
        appl_id=>222,tabname=>'RA_CUSTOMER_TRX_ALL)
```

次を使用してfnd_exclude_table_statsから表エントリを削除します。

```
FND_STATS.LOAD_XCLUD_TAB(action=>'DELETE', appl_id=>&application_id, tabname=>&table_name)
例
        FND_STATS.LOAD_XCLUD_TAB(action=>'DELETE',
        appl_id=>222, tabname=>'RA_CUSTOMER_TRX_ALL')
```

## Database 11g拡張統計

Oracle Optimizerでは個別の列の統計を収集しますが、同じ表内のそれぞれの列に保存されているデータどうしの関連や相関関係については認識しません。

列のグループについて拡張統計を作成すると、SQL文のwhere句で列が一緒に使用されるときの述語の組合せ選択、ひいてはカーディナリティの見積がいっそう正確になります。これらは通常、列のグループのカーディナリティの見積が不正確である場合に使用されます。

列グループはDBMS_STATS.CREATE_EXTENDED_STATSを使用して定義されます。作成後は、統計が表に収集されると、Oracleにより自動的に列グループの統計が保守されます。

**注意: Oracle E-Business Suiteスタンドアロン・パッチで提供される新しいFND_STATSではOracle 11gの拡張統計がサポートされるようになりました。「スタンドアロン・パッチ」セクションを参照してください。Oracle 11gのスタンドアロン・パッチが表1に示されています。**

Oracle E-Business Suiteでは、複数列のグループをシードするのにFND_STATS.LOAD_EXTNSTATS_COLSが使用されます。統計は「スキーマ統計の収集」または「表統計の収集」コンカレント・プログラムによって、またはFND_STATSを使用するときに自動的に保守されます。次の例は、このプロシージャの構文を示します。

```
begin
 FND_STATS.LOAD_EXTNSTATS_COLS
        (action =>'INSERT', appl_id=>&custom_application_id,  tabname => &table_name,        owner=>&owner, colname1
=>&column_name1,  colname2 =>&column_name2, colname3.... );
 FND_STATS.GATHER_TABLE_STATS( ownname=>&owner_name, tabname=>&table_name);
end;
/
```

**注意: 列のグループを定義した後は統計を収集する必要があります。**

## ディクショナリおよび固定オブジェクトの統計の収集

コスト・ベースのオプティマイザが内部(再帰)SQLや、データ・ディクショナリ・ビューや固定オブジェクトを使用するアプリケーションSQLに対して効果的に動作するためには、('SYS'、'SYSTEM'および'SYSAUX'などによって所有される)すべてのディクショナリ表および動的v$パフォーマンス・ビューによって使用されるx$表の統計を収集する必要があります。

10g以降では、FND_STATSはディクショナリや固定オブジェクトの統計は収集せず、かわりにDBMS_STATSを使用する必要があります。ディクショナリや固定オブジェクトの統計の収集頻度に関するOracle E-Business Suite固有のガイドラインはありません。

### ディクショナリ統計

ディクショナリ統計は、データベースに構造上の大きな変更があった場合にのみ収集する必要があります。
例を次に示します。

- EBS全体のアップグレードの一環としてなされた関連プラットフォームやDBのアップグレード後。
- R12アップグレードの後。
- OATMへの移行後。

問題は、いつ、どのくらいの頻度で収集する必要があるかです。典型的な例としては、アップグレードや新規モジュール、データ定義言語(DDL)操作を使用して複数のオブジェクトが作成または再構築されるような大規模なパッチ適用があります。

AWRレポートや他のモニタリング・ツールにおいてデータ・ディクショナリ・オブジェクトに対するSQLの負荷が高い(通常は経過時間が長い)ことが示されている場合や、trace/tkprofファイルでデータ・ディクショナリ・オブジェクトの再帰的SQLに対する時間が長くなっている場合、統計を収集する必要があります。すべてのシステム・スキーマの統計を収集するには、次のコマンドを使用できます。

```
EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS( estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE options
=> 'GATHER AUTO' ); -- 10G以降。
```

### 固定オブジェクトの統計

固定オブジェクトにはx$表とその索引が含まれており、一般的にこの索引にはインスタンスおよびメモリー構造に関する情報が格納されています。SYS.DBA_DATA_FILESのような、ディスク上にあるディクショナリ表とは異なり、V$SQLやV$SQL_PLANといったv$ビューの大部分はx$表に基づいており、これらはインメモリー構造となっています。そのようなわけで、固定表には関連付けられるIOコストがないため、実行計画の生成時にCBOではCPUコストのみが考慮されます。

自動統計収集ジョブでは固定オブジェクトの統計は収集されず、その場合、統計がないとx$表が関係するSQL文で動的サンプリングが自動的には使用されません。その代わり、オプティマイザでは事前定義済のデフォルト値が使用されます。ただし、これらのデフォルトは代表的ではないことが考えられ、結果として最適化されていない実行計画となる可能性があります。

#### 固定オブジェクトの統計を収集する方法とタイミング

固定オブジェクトの統計を収集するには、(Oracle 10g以降で提供されている)DBMS_STATS.GATHER_FIXED_OBJECT_STATSを使用します。パフォーマンスの最適化のため、および一時的というx$表の性質により、システムで代表的なワークロードが発生したときに統計を収集することが重要となります。収集プロセスはリソースを大量に消費する傾向があり、ピーク負荷時のキャパシティが十分でない場合は、データベースが実行開始されてしばらくたち、v$ビューに十分な量のデータが得られた後の時間を選択してください。ただし、負荷が大きいときに固定オブジェクトの統計を収集すると競合の問題が発生し、パフォーマンスの低下やハングアップにつながる可能性があります。詳細は、MyOracleナレッジ・ドキュメント『固定オブジェクトの統計(GATHER_FIXED_OBJECTS_STATS)に関する考慮事項(文書ID 798257.1)』を参照してください。

固定オブジェクトの統計を収集するには、次のコマンドを使用できます。

```
EXECUTE DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

#### 収集のプランニング

固定オブジェクトの統計は、データベース構成に著しい変更があった場合やv$ビューへのアクセスが問題となっている場合に収集する必要があります。Oracle E-Business Suiteには次のような例があります。

- 主要なデータベースまたはアプリケーションのアップグレード*****
- プラットフォームのアップグレード(とりわけExadata)
- 新しいアプリケーション・モジュールの配置
- V$ビューの問合せ時のパフォーマンスの問題
- セッションのワークロードや数の著しい変更
- データベース構造(例: SGA/PGA)の大きな変更の後

たとえば、v$buffer_poolやv$shared_pool_adviceで使用されるx$表など、バッファ・キャッシュおよび共有プールに関する情報が格納されるすべてのx$表は、SGAが増えると大きく変更されることがあります。

**注意: Oracle E-Business Suiteでは統計を収集するDatabase 11g自動DBMSジョブを無効化することは必須ですが、ディクショナリ統計は引き続き収集されるようにしてください。**

***** Oracle E-Business Suiteをアップグレードする際に、$APPL_TOP/admin/adstats.sqlスクリプトでは自動的に統計を収集するOracle DBMSジョブを無効にします。それから、ディクショナリおよび固定オブジェクトの統計を収集します。しかし、データベースはアップグレード中も制限モードで実行されます。そのため、アップグレード後のシステムに代表的なロードがあるときにDBMS_STATS.GATHER_FIXED_OBJECT_STATSを再実行することをお薦めします。

## グローバル一時表

一部のOracle E-Business Suite製品チームはプロセス内でグローバル一時表(GTT)を作成および使用しています。これはプロセスが完了すると削除されます。これらの表は本質的に完全に動的です。標準のパフォーマンス推奨としては、グローバル一時表の統計は収集しないでください。

適切な実行計画を保証するために、次のアプローチが使用可能です。

- 統計がない場合に発生する動的サンプリングに任せる。動的サンプリングは統計が空の場合にのみ機能するため、グローバル一時表に対してDBMS_STATS.delete_table_statsを使用してすべての統計を削除する必要があります。
- ただし、動的サンプリング統計は最適な実行計画とならない可能性があり、たいていは問合せに特定の実行計画を強制するためのオプティマイザ・ヒントを含めることも必要となります。

- FND_STATSによるスキーマ統計の収集ではいずれにしてもグローバル一時表は除外されるため、除外またはロックの必要はありません。

## 一時表

Oracle E-Businessの一時表は通常表であり、これをGTTと混同しないでください。これらは一般に変動が激しく、一部の処理サイクルにおいては通常は空であり、その後再移入されます。例としては、インタフェース表、中間一時表、支払などの特定のプロセスによって選択されたトランザクション、削除対象のレコードなどがあります。

これらの表についてはそれぞれ、最初に空の(nullの)統計を設定して動的サンプリングに任せる必要があります。すでに値が含まれている場合は統計を削除してください。

パフォーマンスに問題がある場合のみ、表に代表的なロードがある場合は統計を収集するか、またはFND_STATS.SET_TABLE_STATSを使用して移入することを考慮します。

いずれの場合でも、統計は(収集されないように)ロックする必要があります。

## パーティション表の増分統計

増分統計収集はOracle 11gR2のDBMS_STATSの新機能で、FND_STATSで全面的にサポートされています。Oracleでは表全体ではなく新しいまたは変更されたパーティションのみをスキャンすることによりグローバル統計を提供します。これは大容量のOracle E-Business Suite表がある場合にとりわけ有効です。

増分グローバル統計は、各パーティションについて統計を収集してシノプシス(統計メタデータ)を保存することによって機能します。シノプシスは各パーティションの統計メタデータおよび列によって構成されます。このシノプシスは通常は数KBにすぎず、SYSAUX表領域に保存されます。グローバル統計は表全体を読み取るのではなく、各パーティションのシノプシスを集計することによって作成されます。新しいパーティションにシノプシスが作成されると、すべてのシノプシスが集計されて表レベルの統計が更新されます。

このアプローチのためには、次の前提条件が満たされている必要があります。

- パーティション表のINCREMENTAL設定がTRUEに設定されている
- DBMS_STATS.GATHER_*_STATSのパラメータGRANULARITYに"ALL"または"AUTO"が設定されている
- ESTIMATE_PERCENTがAUTO_SAMPLE_SIZE (Oracle 11gのデフォルト)に設定されている

増分統計を新しいOracle 11g DBMS_STATS.AUTO_SAMPLE_SIZEと組み合せることで、正確性の高い統計を収集するのにかかる時間を大幅に削減できます。

#### Oracle E-Business Suite

増分統計はどのような大規模なパーティション表にも適用可能です。表プリファレンスが設定されると、Oracle E-Business Suiteの統計の収集(表またはスキーマ)を使用して自動的に統計を収集してシノプシスを作成できます。

#### 増分プリファレンス

増分プリファレンスがXLA_AE_LINES表(これはOracle E-Business Suite12 Subledger Accountingの一部です)に対して設定されているかどうかをチェックするには、次のコマンドを使用します。

```
select dbms_stats.get_prefs('INCREMENTAL', 'XLA','XLA_AE_LINES')
from dual;
```

TRUEが返される場合、プリファレンスが設定されています。

このプリファレンスは、次のコマンドを使用して設定できます。

```
exec dbms_stats.set_table_prefs('XLA','XLA_AE_LINES','INCREMENTAL','TRUE');
```

テストの際に、次を使用してこのプリファレンスをオフにできることを確認しておくとよいでしょう。

```
exec dbms_stats.set_table_prefs('XLA', 'XLA_AE_LINES, 'INCREMENTAL', 'FALSE');
```

#### 粒度パラメータ

統計の収集をスキーマまたは表レベルで実行する場合:

- 粒度パラメータを"ALL"または"AUTO"に設定します。このいずれも実際上は同じであり、すべての粒度レベルが呼び出されます。
- *「評価率」*パラメータを"DBMS_STATS.AUTO_SAMPLE_SIZE"に設定します。これはOracle 11gのデフォルトです。FND_STATSでカスタム・スクリプトを使用する場合は、粒度を"ALL"または"AUTO"に設定します。例:

```
exec fnd_stats.gather_table_stats
     ('OWNER', 'TABLE_NAME', granularity =>'ALL');
```

## 収集時間の改善

統計収集プロセスの速度を向上させる方法は2通りあります。

#### 並列度

FND_STATSのDEGREE (並列度)は、大規模なオブジェクトの統計収集の速度向上に役立ちます。並列度が指定されていない場合、FND_STATSにより自動的に並列度が決定されますが、デフォルトではparallel_max_serversおよびcpu_countのいずれか小さい方になります。このアプローチはオブジェクト内並列度として定義されています。

#### コンカレント統計収集

コンカレント統計収集は、オブジェクト間の並列を実現するOracle 11gR2のDBMS_STATSの新しい機能です。CONCURRENTがTRUEに設定され(デフォルトはFALSE)、*「評価率」*がDBMS_STATS.AUTO_SAMPLE_SIZEに設定されると、OracleではOracle Job Scheduler (*job_queue_processes*は4より大きい数に設定する必要があります)およびAdvanced Queuingを使用して、複数のオブジェクトに対して複数の統計収集ジョブが同時に実行されます。

このプリファレンスは、次のコマンドを使用して設定できます。

```
exec DBMS_STATS.SET_GLOBAL_PREFS('CONCURRENT','TRUE');
```

これは、sysユーザーとしてログインして設定する必要があります。

DEGREEを使用している場合、パラレル・サーバーは一度に1つの表パーティションのみを処理します。この制限は、Oracle 11gR2のコンカレント機能と組み合わせることで克服できます。

この組合せによるアプローチは使用可能なCPUが潤沢にある場合に最も有効です。これは、統計収集プロセスの実行時にキャパシティに余裕がある複数のCPUがあるシステム向けです。(大規模なパーティションがないような)小規模なサーバーや、従来の(非コンカレント)統計収集時に最大CPUに達する大規模なサーバーではほとんど差は出ません。

これらの機能がサポートされてはいるものの、11gR2のFND_STATSではAUTOサンプル・サイズおよび統計収集時の並列化がサポートされているため、著しい改善が見られるとはかぎりません。さらに、FND_STATSは

gather_schema_statisticsではなくDBMS_STATS.gather_table_statsを呼び出します。そのため、この機能は同じ表の異なるパーティションの統計を同時に収集する場合にのみ使用できます。それで、これはデータがパーティションにおいて適度に分散されているパーティション表においてのみ有効です。

## 統計の検証

FND_STATS.VERIFY_STATSプロシージャは統計が最新であるかをチェックするのに使用できます。レポートが生成され、次の構文を使用して実行できます。

```
set serveroutput on
set long 10000
exec apps.fnd_stats.verify_stats (‘OWNER’, ‘TABLE_NAME’);
```

例:

```
exec apps.fnd_stats.verify_stats ('ONT', 'ONT.OE_ORDER_HOLDS_ALL');
```

別の方法として、My Oracle Supportナレッジ・ドキュメント163208.1の次のスクリプトを使用して統計を検証できます。*bde_last_analyzed.sql - CBO 統計の検証。*

## スタンドアロン・パッチ

FND_STATコードに実装された次の新機能を提供するOracle 11gのスタンドアロン・パッチが「表1: Oracle E-Business Suiteスタンドアロン・パッチ」に示されています。

1. Database 11gのAUTO統計サンプリング生成オプション

   Autoサンプリング・スタンドアロン・パッチが適用できない場合は、My Oracleナレッジ・ドキュメント156968.1で提供されている自動化スクリプトを参照してください: coe_stats.sql - FND_STATSおよび表サイズの使用によるCBO統計の収集の自動化。この暫定的なソリューションは、範囲、サンプル・サイズおよび収集頻度に基づいて推奨を提供します。

2. Database 11gの拡張統計収集機能

**表1. ORACLE E-BUSINESS SUITEスタンドアロン・パッチ (ORACLE 11G)**

| E-BUSINESS SUITEのバージョン | パッチ |
|---|---|
| 12.1.x | 16410424:R12.FND.B |
| 12.0.x | 16410424:R12.FND.A |
| 11.5.10 | 14707975 |

# パフォーマンスのテスト・ケースおよび例

1. Oracle 11g拡張オプティマイザ統計
2. 増分統計収集
3. コンカレント統計収集

## 1. Oracle 11g拡張オプティマイザ統計

この簡単な例は、Oracle E-Business SuiteでOracle 11g拡張(複数列)オプティマイザ統計機能を配置する方法をデモします。思い出してほしい点ですが、個別の列統計では特定の列を選択できますが、where句に同じ表からの複数の列が含まれている場合、個別の列統計には列どうしの関係を示すものは何もありません。拡張統計により個別の述語の組合せ選択、ひいてはカーディナリティの見積がいっそう正確になり、実行計画がより優れたものとなります。

このFND_LOOKUP_VALUES表はこの例の基礎となります。サイズが適度に小さく、わかりやすく、いくつかのモジュールで使用されているためになじみがあると思われるためです。

これにはOracle Application Object Libraryのクイックコード値が格納されます。原則的に、各行にはクイックコード参照タイプ、クイックコードそのもの、その意味、および追加の説明が格納されます(この例に直接関係のない他の列もあります)。これらの値は、Oracle E-Business Suiteフォームのいくつかにおいて値リストで使用されています。

値のいくつかは特定のモジュールでのみ使用され、そしてこれらの値は通常、いずれもデータの問合せ時に参照されます。これは拡張統計を使用する際の主要条件の1つです。そのようなわけで、これは列の間に相関関係があることを暗示しており、LOOKUP_TYPEが実際にはVIEW_APPLICATION_IDに依存しています。

#### 1. 拡張統計のない統計

このテスト・ケースにおいて複数列の拡張統計のない表の統計を収集するために、FND_LOOKUP_VALUES表を使用します。

- FND_LOOKUP_VALUES表を表示します。

```
desc FND_LOOKUP_VALUES

Datatype               Length        Mandatory Comments
---------------------- ------------- --------- -------------------------
LOOKUP_TYPE            VARCHAR2 (30)       Yes QuickCode lookup type
LANGUAGE               VARCHAR2 (30)       Yes Language
LOOKUP_CODE            VARCHAR2 (30)       Yes QuickCode code
MEANING                VARCHAR2 (80)       Yes QuickCode meaning
DESCRIPTION            VARCHAR2 (240)          Description
ENABLED_FLAG           VARCHAR2 (1)        Yes Enabled flag
SECURITY_GROUP_ID      NUMBER (15)         Yes Security group identifier
VIEW_APPLICATION_ID    NUMBER (15)         Yes Identifies which application's view will include the
                                               lookup values
TERRITORY_CODE         VARCHAR2 (2)            Territory code of territory using the language
....
....
```

- FND_LOOKUP_VALUES表の合計カーディナリティ:

```
select count(1) from FND_LOOKUP_VALUES;

  COUNT(1)
----------
    532559
```

- FND_LOOKUP_VALUES表の統計の収集の実行:

```
exec fnd_stats.gather_table_stats('APPLSYS','FND_LOOKUP_VALUES');
```

- dba_tablesの更新済統計を確認します。

```
select table_name, num_rows from dba_tables where table_name='FND_LOOKUP_VALUES';

TABLE_NAME                     NUM_ROWS
------------------------------ --------
FND_LOOKUP_VALUES                532559
```

- 次のヒストグラムは、FNDデータ・ディクショナリにすでに存在しています:

```
select table_name,column_name from fnd_histogram_cols where table_name='FND_LOOKUP_VALUES';

TABLE_NAME                     COLUMN_NAME
------------------------------ ------------------------------
FND_LOOKUP_VALUES              LANGUAGE
FND_LOOKUP_VALUES              VIEW_APPLICATION_ID
```

- 表データの配分を確認します。

```
select VIEW_APPLICATION_ID,lookup_type, count(*)
from FND_LOOKUP_VALUES
group by VIEW_APPLICATION_ID,lookup_type
order by 3 desc;

VIEW_APPLICATION_ID  LOOKUP_TYPE                    COUNT(*)
-------------------  ------------------------------ --------
                  3  NO_POSTAL_CODE                   13698
                  3  FI_POSTAL_CODE                   11496
                  3  GHR_US_ACADEMIC_INSTITUTION      11043
                  3  AE_OCCUPATION                     8760
                  3  PL_COMMUNITY                      7437
                222  NAICS_2002                        7035
               8901  FV_PSC_TYPE                       6906
                  3  NAIC                              6306
                222  NAICS_1997                        5430
                  3  GHR_US_ACADEMIC_DISCIPLINE        5127
                222  NACE                              4599
                222  1987 SIC                          4512
                222  1972 SIC                          4338
               8405  PE_US_COURSE_SUBJECT              4299
                  3  DK_POSTCODE_TOWN                  4002
               8901  FV_NAICS_TYPE                     3537
                222  1977 SIC                          3111
               8901  FV_SIC_TYPE                       3012
                200  NUMBERS                           3000
                  3  GHR_US_OCC_SERIES                 2889
                  3  NO_TAX_MUNICIPALITY               2619
                  3  GHR_US_LANG_IDENTIFIER            2547
                  3  GHR_US_FOR_ALL_LOC                2238
                  3  GHR_US_AGENCY_CODE                2208
                  0  BUSINESS_ENTITY                   2199
                222  NAF                               2145
                  3  GHR_US_LEGAL_AUTHORITY            2031
                  3  GHR_US_HEALTH_PLAN                2001
               8405  IMPORT_ERROR_CODE                 1890
                  3  HU_JOB_FEOR_CODES                 1884
                222  HZ_RELATIONSHIP_ROLE              1521
               8901  FV_FSC_TYPE                       1365
                  3  AE_AREA_CODES                     1350
```

- fnd_lookup_Values表のデータ配分をチェックします。

```
select table_name, column_name, num_distinct, density, num_buckets, histogram from user_tab_col_statistics where
table_name = 'FND_LOOKUP_VALUES';

TABLE_NAME          COLUMN_NAME               NUM_DISTINCT DENSITY    NUM_BUCKETS HISTOGRAM
------------------- ------------------------- ------------ ---------- ----------- ---------
FND_LOOKUP_VALUES   LOOKUP_TYPE               17203        .000058129 1           NONE
FND_LOOKUP_VALUES   LANGUAGE                  3            9.3886E-07 3           FREQUENCY
FND_LOOKUP_VALUES   LOOKUP_CODE               86164        .000011606 1           NONE
FND_LOOKUP_VALUES   MEANING                   330917       3.0219E-06 1           NONE
FND_LOOKUP_VALUES   DESCRIPTION               207029       4.8302E-06 1           NONE
FND_LOOKUP_VALUES   ENABLED_FLAG              4            .25        1           NONE
FND_LOOKUP_VALUES   SECURITY_GROUP_ID         17           .058823529 1           NONE
FND_LOOKUP_VALUES   VIEW_APPLICATION_ID       87           9.3886E-07 87          FREQUENCY
FND_LOOKUP_VALUES   TERRITORY_CODE            10           .1         1           NONE
....
....
```

- view_application_idおよびlookup_typeの様々な値の選択のexplain planをチェックします

```
explain plan for select * from FND_LOOKUP_VALUES where VIEW_APPLICATION_ID=201 and lookup_type=:b2;

Explained.

select plan_table_output from table(dbms_xplan.display('plan_table',null,'serial'));

PLAN_TABLE_OUTPUT
```

```
-------------------------------------------------------------------------------------------------------
| Id | Operation                    | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT             |                      |     1 |   152 |     4   (0)| 00:00:01 |
|  1 |  TABLE ACCESS BY INDEX ROWID |FND_LOOKUP_VALUES     |     1 |   152 |     4   (0)| 00:00:01 |
|* 2 |   INDEX RANGE SCAN           |FND_LOOKUP_VALUES_U1  |     1 |       |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

Predicate Information:   2 - access("LOOKUP_TYPE"=:B2 AND "VIEW_APPLICATION_ID"=201)

explain plan for select * from FND_LOOKUP_VALUES where VIEW_APPLICATION_ID=3 and lookup_type='NO_POSTAL_CODE';
Explained.
select plan_table_output from table(dbms_xplan.display('plan_table',null,'serial'));

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------
| Id | Operation                    | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT             |                      |    12 |  1680 |    13   (0)| 00:00:01 |
|  1 |  TABLE ACCESS BY INDEX ROWID |FND_LOOKUP_VALUES     |    12 |  1680 |    13   (0)| 00:00:01 |
|* 2 |   INDEX RANGE SCAN           |FND_LOOKUP_VALUES_U2  |    12 |       |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

Predicate Information:   2 - access("LOOKUP_TYPE"='NO_POSTAL_CODE' AND "VIEW_APPLICATION_ID"=3)

explain plan for select * from FND_LOOKUP_VALUES where VIEW_APPLICATION_ID=3 and lookup_type='AE_ARIA_CODE';
Explained.

select plan_table_output from table(dbms_xplan.display('plan_table',null,'serial'));

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------------------------------
| Id | Operation                    | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT             |                      |    12 |  1680 |    13   (0)| 00:00:01 |
|  1 |  TABLE ACCESS BY INDEX ROWID |FND_LOOKUP_VALUES     |    12 |  1680 |    13   (0)| 00:00:01 |
|* 2 |   INDEX RANGE SCAN           |FND_LOOKUP_VALUES_U2  |    12 |       |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

Predicate Information:   2 - access("LOOKUP_TYPE"='AE_ARIA_CODE' AND "VIEW_APPLICATION_ID"=3)
```

### 2. 拡張統計のある統計

拡張統計のアプローチでは、データ・ディクショナリ内の列グループの列統計エントリを作成します。この例では、結果がすぐに表示されるように手動で収集されています。

- 複数列の拡張統計を作成する手順

- fnd_extnstats_cols表を問い合せて拡張統計が存在しないことを確認します。fnd_exnstats_colsは、拡張統計が作成される列が保持されるFND表です。

```
select table_name from fnd_extnstats_cols;
no rows selected
```

- FND_LOKUP_VALUES表の現在の統計を削除します

```
analyze table applsys.fnd_lookup_values delete statistics;
Table analyzed.
```

- LOOKUP_TYPEおよびVIEW_APPLICATION_IDの拡張統計を作成します:

```
set serveroutput on
```

- fnd_stats.load_extnstats_colsを実行します。fnd_statsを使用して複数列の拡張統計を作成します

```
exec fnd_stats.load_extnstats_cols(action =>'INSERT',appl_id=>0,tabname => 'FND_LOOKUP_VALUES',colname1
=>'VIEW_APPLICATION_ID',colname2 =>'LOOKUP_TYPE');

PL/SQL procedure successfully completed.
```

- 仮想列が拡張統計に関係する複数列に対して作成されているかどうかをチェックします。

```
select owner,table_name,column_name,histogram from dba_tab_cols where table_name like 'FND_LOOKUP_VALUES' and owner
='APPLSYS';
```

```
 OWNER                          TABLE_NAME          COLUMN_NAME                    HISTOGRAM
 ------------------------------ ------------------- ------------------------------ --------------------
 APPLSYS                        FND_LOOKUP_VALUES   ZD_EDITION_NAME                NONE
 APPLSYS                        FND_LOOKUP_VALUES   SYS_STU_L5AKW15J1DXUKXDMYW3UJF NONE
 APPLSYS                        FND_LOOKUP_VALUES
```

- 上記の出力は、仮想列"SYS_STU_L5AKW15J1DXUKXDMYW3UJF"が作成されたことを示しています。

- 仮想列にヒストグラムが作成されるかどうかをチェックするために'FND_LOOKUP_VALUES'の統計を収集します

```
exec fnd_stats.gather_table_stats('APPLSYS','FND_LOOKUP_VALUES');

PL/SQL procedure successfully completed.
```

- 仮想列にヒストグラムが作成されているかどうかをチェックします

```
select owner,table_name,column_name,histogram from dba_tab_cols where table_name like 'FND_LOOKUP_VALUES' and owner
='APPLSYS';

 OWNER                          TABLE_NAME COLUMN_NAME                                HISTOGRAM
 ------------------------------ ---------- ------------------------------------------ -------------
 APPLSYS                        FND_LOOKUP_VALUES ZD_EDITION_NAME                     NONE

 APPLSYS                        FND_LOOKUP_VALUES SYS_STU_L5AKW15J1DXUKXDMYW3UJF HEIGHT BALANCED
```

- 上記の出力は、仮想列に高さ調整済ヒストグラムが作成されていることを示しています
- 次に、view_application_idおよびlookup_typeの様々な値の選択のexplain planの変化をチェックします

```
explain plan for select * from FND_LOOKUP_VALUES where VIEW_APPLICATION_ID=201 and lookup_type=:b2;
Explained.
select plan_table_output from table(dbms_xplan.display('plan_table',null,'serial'));

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                      |    30 |  4560 |    27   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| FND_LOOKUP_VALUES    |    30 |  4560 |    27   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | FND_LOOKUP_VALUES_U2 |    30 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("LOOKUP_TYPE"=:B2 AND "VIEW_APPLICATION_ID"=201)

explain plan for select * from FND_LOOKUP_VALUES where VIEW_APPLICATION_ID=3 and lookup_type='NO_POSTAL_CODE'
Explained.
select plan_table_output from table(dbms_xplan.display('plan_table',null,'serial'));

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------
| Id | Operation          | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                   | 12580 |  1867K|  2729   (2)| 00:00:33 |
|*  1 |  TABLE ACCESS FULL| FND_LOOKUP_VALUES | 12580 |  1867K|  2729   (2)| 00:00:33 |
-----------------------------------------------------------------------------------------


Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("LOOKUP_TYPE"='NO_POSTAL_CODE' AND "VIEW_APPLICATION_ID"=3)

explain plan for select * from FND_LOOKUP_VALUES where VIEW_APPLICATION_ID=3 and lookup_type='AE_ARIA_CODE';
Explained.
select plan_table_output from table(dbms_xplan.display('plan_table',null,'serial'));

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                      |    24 |  3648 |    22   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| FND_LOOKUP_VALUES    |    24 |  3648 |    22   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | FND_LOOKUP_VALUES_U2 |    24 |       |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------
```

```
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("LOOKUP_TYPE"='AE_ARIA_CODE' AND "VIEW_APPLICATION_ID"=3)
```

結果: 上記の結果は、複数列の拡張統計の作成後にカーディナリティの見積が変わることを示しています。

## 2. 増分統計収集

増分統計収集は11G Databaseの新機能です。増分統計は、最後に実行されてから変更があったパーティションのみをスキャンすることによってグローバル統計が増分更新されるパーティション表に対してのみ有効です。この簡単な例は、Oracle E-Business SuiteでOracle 11gの増分統計の収集機能を配置する方法をデモします。

- 表xla_ae_lines_bkpをこのテスト・ケースを実行するパーティション表として作成します
- xla_ae_lines表からデータを挿入します

```
insert into xla.xla_Ae_lines_bkp select * from xla.xla_ae_lines;

1909409 rows created.
```

- 増分プリファレンスを設定しないテスト・ケース

```
set time on timing on

 exec fnd_stats.gather_table_stats('XLA','XLA_AE_LINES_BKP');

PL/SQL procedure successfully completed.

Elapsed: 00:00:29.06
```

- データ挿入後の最初の統計収集には29.06秒かかりました。
- グローバルおよびパーティション・レベルの統計をチェックします

```
select table_name,to_Char(last_analyzed,'DD-MON-YY HH24:MI:SS') "last_analyzed" from dba_Tables where table_name like
'XLA_AE_LINES_BKP';

TABLE_NAME                     last_analyzed
------------------------------ ------------------------
XLA_AE_LINES_BKP               23-SEP-12 07:04:34

select partition_name,to_Char(last_analyzed,'DD-MON-YY HH24:MI:SS') "last_analyzed" from dba_Tab_partitions where
table_name like 'XLA_AE_LINES_BKP';

PARTITION_NAME                 last_analyzed
------------------------------ ------------------------
AP                             23-SEP-12 07:04:15
AR                             23-SEP-12 07:04:16
CE                             23-SEP-12 07:04:16
CST                            23-SEP-12 07:04:23
DPP                            23-SEP-12 07:04:23
FUN                            23-SEP-12 07:04:23
FV                             23-SEP-12 07:04:23
GMF                            23-SEP-12 07:04:23
IGC                            23-SEP-12 07:04:23
IGI                            23-SEP-12 07:04:23
LNS                            23-SEP-12 07:04:23
OFA                            23-SEP-12 07:04:23
OKL                            23-SEP-12 07:04:23
OZF                            23-SEP-12 07:04:23
PA                             23-SEP-12 07:04:24
PAY                            23-SEP-12 07:04:24
PN                             23-SEP-12 07:04:24
PO                             23-SEP-12 07:04:24
PSB                            23-SEP-12 07:04:24
```

- 1つのパーティションからデータを削除して統計がどのように計算されるかをチェックします

```
delete from xla.xla_ae_lines_bkp where application_id=222;
```

- fnd_Statsの使用による統計の収集

```
exec fnd_stats.gather_table_stats('XLA','XLA_AE_LINES_BKP');

PL/SQL procedure successfully completed.

Elapsed: 00:00:14.06
```

- データの削除後にプリファレンスを設定せずに統計収集を実行しました。かかった時間は14.06秒でした

- グローバル統計とパーティション統計をチェックします

```
select table_name,to_Char(last_analyzed,'DD-MON-YY HH24:MI:SS') "last_analyzed" from dba_Tables where table_name like
'XLA_AE_LINES_BKP';

TABLE_NAME                     last_analyzed
------------------------------ ------------------------
XLA_AE_LINES_BKP               23-SEP-12 07:10:26


select partition_name,to_Char(last_analyzed,'DD-MON-YY HH24:MI:SS') "last_analyzed" from dba_Tab_partitions where
table_name like 'XLA_AE_LINES_BKP';

PARTITION_NAME                 last_analyzed
------------------------------ ------------------------
AP                             23-SEP-12 07:10:14
AR                             23-SEP-12 07:10:14
CE                             23-SEP-12 07:10:14
CST                            23-SEP-12 07:10:15
DPP                            23-SEP-12 07:10:15
FUN                            23-SEP-12 07:10:15
FV                             23-SEP-12 07:10:15
GMF                            23-SEP-12 07:10:15
IGC                            23-SEP-12 07:10:15
IGI                            23-SEP-12 07:10:15
LNS                            23-SEP-12 07:10:16
OFA                            23-SEP-12 07:10:16
OKL                            23-SEP-12 07:10:16
OZF                            23-SEP-12 07:10:16
PA                             23-SEP-12 07:10:17
PAY                            23-SEP-12 07:10:17
PN                             23-SEP-12 07:10:17
PO                             23-SEP-12 07:10:17
PSB                            23-SEP-12 07:10:17

19 rows selected.
```

**注意：** ARパーティションのデータのみが削除されますが、すべてのパーティションの統計が収集され、すべてのパーティションのlast_analyzedが更新されます。

- プリファレンスをtrueに設定した後に同じテスト・ケースを実行します。次のコマンドにより表レベルで増分プリファレンスが設定され、次に統計の収集が実行されるときには、この表に対する統計増分が収集されます
- 表xla_ae_lines_bkpを削除し、このテスト・ケースを実行するパーティション表として再作成します
- xla_ae_lines表からデータを挿入します

```
insert into xla.xla_Ae_lines_bkp select * from xla.xla_ae_lines;

1909409 rows created.

exec dbms_stats.set_table_prefs('XLA','XLA_AE_LINES_BKP','INCREMENTAL','TRUE');
```

- プリファレンスが設定されているかどうかをチェックします

```
select dbms_stats.get_prefs('INCREMENTAL','XLA','XLA_AE_LINES_BKP') from dual;

DBMS_STATS.GET_PREFS('INCREMENTAL','XLA','XLA_AE_LINES_BKP')
------------------------------------------------------------
TRUE
```

- プリファレンスの設定後に統計を収集します。

```
exec fnd_stats.gather_table_stats('XLA','XLA_AE_LINES_BKP');

PL/SQL procedure successfully completed.

Elapsed: 00:00:22.91
```

- プリファレンスの設定後の、データ挿入後の最初の統計収集には22.91秒かかりました。
- 統計の所要時間のチェック

```
select table_name,to_Char(last_analyzed,'DD-MON-YY HH24:MI:SS') "last_analyzed" from dba_Tables where table_name like
'XLA_AE_LINES_BKP';

TABLE_NAME                     last_analyzed
------------------------------ ------------------------
XLA_AE_LINES_BKP               23-SEP-12 07:17:32

select partition_name,to_Char(last_analyzed,'DD-MON-YY HH24:MI:SS') "last_analyzed" from dba_Tab_partitions where
table_name like 'XLA_AE_LINES_BKP';
```

```
PARTITION_NAME                 last_analyzed
------------------------------ ------------------------
AP                             23-SEP-12 07:17:30
AR                             23-SEP-12 07:17:12
CE                             23-SEP-12 07:17:10
CST                            23-SEP-12 07:17:21
DPP                            23-SEP-12 07:17:21
FUN                            23-SEP-12 07:17:12
FV                             23-SEP-12 07:17:10
GMF                            23-SEP-12 07:17:10
IGC                            23-SEP-12 07:17:10
IGI                            23-SEP-12 07:17:12
LNS                            23-SEP-12 07:17:10
OFA                            23-SEP-12 07:17:10
OKL                            23-SEP-12 07:17:12
OZF                            23-SEP-12 07:17:30
PA                             23-SEP-12 07:17:12
PAY                            23-SEP-12 07:17:12
PN                             23-SEP-12 07:17:12
PO                             23-SEP-12 07:17:10
PSB                            23-SEP-12 07:17:30

19 rows selected.
```

- 1つのパーティションからデータを削除し、増分統計収集が次の統計の収集時にどのように役立つかを見てみます。

```
delete from xla_ae_lines_bkp where application_id=222;

100233 rows deleted
```

- application_id 222のパーティションARからデータを削除した後に統計を収集します

```
exec fnd_stats.gather_table_stats('XLA','XLA_AE_LINES_BKP');

PL/SQL procedure successfully completed.

Elapsed: 00:00:04.11
```

- 1つのパーティションのデータの削除後、統計の増分収集に4.11秒かかりました
- グローバルおよびパーティション統計をチェックします

```
select table_name,to_Char(last_analyzed,'DD-MON-YY HH24:MI:SS') "last_analyzed" from dba_Tables where table_name like
'XLA_AE_LINES_BKP';

TABLE_NAME                     last_analyzed
------------------------------ ------------------------
XLA_AE_LINES_BKP               23-SEP-12 07:26:20

select partition_name,to_Char(last_analyzed,'DD-MON-YY HH24:MI:SS') "last_analyzed" from dba_Tab_partitions where ta
ble_name like 'XLA_AE_LINES_BKP';

PARTITION_NAME                 last_analyzed
------------------------------ ------------------------
AP                             23-SEP-12 07:17:30
AR                             23-SEP-12 07:26:18
CE                             23-SEP-12 07:17:10
CST                            23-SEP-12 07:17:21
DPP                            23-SEP-12 07:17:21
FUN                            23-SEP-12 07:17:12
FV                             23-SEP-12 07:17:10
GMF                            23-SEP-12 07:17:10
IGC                            23-SEP-12 07:17:10
IGI                            23-SEP-12 07:17:12
LNS                            23-SEP-12 07:17:10
OFA                            23-SEP-12 07:17:10
OKL                            23-SEP-12 07:17:12
OZF                            23-SEP-12 07:17:30
PA                             23-SEP-12 07:17:12
PAY                            23-SEP-12 07:17:12
PN                             23-SEP-12 07:17:12
PO                             23-SEP-12 07:17:10
PSB                            23-SEP-12 07:17:30

19 rows selected.

Elapsed: 00:00:00.01
```

注意： 統計が収集され、ARパーティションについてのみlast_analyzed dateが変更されており、所要時間も短縮されました。

- 上記のテスト・ケースでは、増分プリファレンスが設定されたときに所要時間に改善が見られることがわかります。

- 所要時間情報

```
Preference set    Table_name      Operation                                          Timing
---------------   -------------   -------------------------------------------------- --------------
NO                xla_ae_lines    stats gathered after inserting data for first time    29.06 sec
Yes               xla_ae_lines    stats gathered after inserting data for first time    22.91 sec
NO                xla_ae_lines    stats gathered after modifying one partition          14.06 sec
Yes               xla_ae_lines    stats gathered after modifying one partition          04.11 sec
---------------   -------------   -------------------------------------------------- --------------
```

## 3. コンカレント統計収集

コンカレント統計収集は11gR2で導入された新機能です。コンカレント統計収集は、複数の表およびパーティションに対して同時に統計収集を実行するのに使用されます。この簡単な例は、Oracle E-Business SuiteでOracle 11gのコンカレント統計の収集機能を配置する方法をデモします。

- コンカレント統計収集は、グローバル・プリファレンスの設定により管理されます

```
exec DBMS_STATS.SET_GLOBAL_PREFS('CONCURRENT','TRUE');
```

- 統計を同時に実行するには、job_queue_processesを4より大きい数字に設定する必要があります
- 統計を同時に収集するテスト・ケースです。
- job_queue_processesを10に設定します。

```
alter system set job_queue_processes=10;
system altered.
```

- コンカレント・プログラム・プリファレンスをFALSEに設定します

```
exec DBMS_STATS.SET_GLOBAL_PREFS('CONCURRENT','FALSE');
```

- 表の統計を削除します

```
exec dbms_stats.delete_table_stats('XLA','XLA_AE_HEADERS');
```

- fnd_statsを使用して統計を収集します

```
exec fnd_stats.gather_table_stats('XLA','XLA_AE_HEADERS');
Elapsed: 00:29:30.97
```

- グローバル・プリファレンスをTRUEに設定します

```
exec DBMS_STATS.SET_GLOBAL_PREFS('CONCURRENT','TRUE');
```

- 統計を削除します

```
exec dbms_stats.delete_table_stats('XLA','XLA_AE_HEADERS');
PL/SQL procedure successfully completed.
```

- 統計の収集

```
exec fnd_stats.gather_table_stats('XLA','XLA_AE_HEADERS');
PL/SQL procedure successfully completed.
Elapsed: 00:26:59.54
Results of the test case:

Method      Level (schema/table)   Concurrent processing option  Timing
----------  ---------------------  ----------------------------  ------------
FND_STATS   table(XLA_AE_HEADERS)  FALSE                         00:29:30.97
FND_STATs   table(XLA_AE_HEADERS)  TRUE                          00:26:59.54
```

上記のテスト・ケースから、グローバル・プリファレンスをFALSEに設定して(fnd_statsはデフォルトではパラレル・オプションありで実行されることから、パラレル・オプションありで) fnd_statsを実行した場合は29分30秒かかったことがわかります。グローバル・プリファレンスをTRUEに設定してfnd_statsを実行した場合は、26分59秒かかりました。統計収集の所要時間が2分5秒短縮されたことがわかります。この小さな短縮ですが、すでにキャッシュにあるブロックにより説明できます。

# 付録A: 関連ドキュメント

この付録では、有益な情報が提供されている重要なOracleドキュメント、My Oracle Supportナレッジ・ドキュメントおよびブログ参照を紹介します。

## ホワイトペーパー

次のペーパーは一般的な内容であり、特にOracle E-Business Suiteにフォーカスしたものではありません。

- *オプティマイザ統計の収集のベスト・プラクティス*
  このペーパーではOracle Databaseの統計収集プロセスの幅広い概要が提供されており、このペーパーの前に見ておくと役立ちます。これは、このドキュメントに出てくる多くの一般的な文のソースという位置づけです。これは、Oracle Optimizerブログ: *https://blogs.oracle.com/optimizer/*およびOracle Technet: *http://www.oracle.com/technetwork/database/bi-datawarehousing/dbbi-tech-info-optmztn-092214.html*で提供されています。
- *オプティマイザ統計の理解*
  この文書には、オプティマイザ統計についてのより基礎的な情報および関連トピックが含まれています。たとえば、各種タイプのヒストグラムおよびそれがどこで使用できるかについてのわかりやすい説明が含まれています。これは、*『オプティマイザ統計の収集のベスト・プラクティス』*ペーパーの前に読んでおくとよいでしょう。これは、Oracle Technet: *http://www.oracle.com/technetwork/database/focus-areas/bi-datawarehousing/twp-optimizer-stats-concepts-110711-1354477.pdf*で提供されています。
- *Oracle Database 10gから11gへのアップグレード: オプティマイザでできること*
  このペーパーでは、新しいOracle 11gオプティマイザ機能について紹介し、計画変更に関連して起こりうるパフォーマンス低下を避けるためにアップグレードの前後に踏む必要があるステップの概要について説明します。これは、Oracle Technet: *http://www.oracle.com/technetwork/database/focus-areas/bi-datawarehousing/twp-upgrading-10g-to-11g-what-to-ex- 133707.pdf*で提供されています。

## My Oracle Supportナレッジ・ドキュメント

- *Fixed Objects Statistics Considerations* (Doc ID 798257.1)
  この一般的なドキュメントには、パフォーマンスの低下やハングアップにつながりかねない、たとえばx$/固定表の統計の欠如や不正といった問題がまとめられています。
- *Managing CBO Stats during an upgrade to Oracle 10g or Oracle 11g* (Doc ID 465787.1)
  記載されている主な内容は、アプリケーション・スキーマについて引き続き通常どおりオプティマイザ統計を収集しつつ、非アプリケーション・オブジェクト(データ・ディクショナリ、固定オブジェクトおよびシステム・パフォーマンス)についてのクリーンなベースラインを作成する方法です。
- *bde_last_analyzed.sql - Verifies CBO Statistics* (Doc ID 163208.1)
  このスクリプトは、すべての表、索引およびパーティションのデータ・ディクショナリのオプティマイザ統計を検証します。また、'SYS'により所有される表と索引の統計も検証します。
- *EBPERF FAQ - Collecting Statistics with Oracle EBS 11i and R12 (Doc ID 368252.1)*

  このFAQノートは、Apps 11iおよびR12の統計の収集に関する一般的な質問に答えています。

次のMy Oracle Supportナレッジ・ドキュメントは、FND_STATSスタンドアロン・パッチを適用できない場合に役立ちます。

- *coe_stats.sql - Automates CBO Stats Gathering using FND_STATS* (Doc ID 156968.1)
  この暫定的なソリューションは、範囲、サンプル・サイズおよび収集頻度に基づいて推奨を提供します。

## ブログ

Oracle Optimizerのブログ( https://blogs.oracle.com/optimizer )では、*『オプティマイザ統計の収集のベスト・プラクティス』*ペーパーの著者による、いくつかのトピックに関する優れた一般情報が提供されています。

含まれるトピックには次のようなものがあります。

- 拡張統計
- コンカレント統計収集
- 大規模なパーティション表の統計の保守
- 固定オブジェクトの統計、およびなぜそれが重要か

加えて、カーディナリティ・フィードバックなど、チューニングに関連した他の問題に関する記事があります。

# 付録B: パッケージ仕様

この付録では、「表統計の収集」および「スキーマ統計の収集」の各プログラムについて説明します。このどちらも、作業のできるだけ多くのパラレル化を試み、オプションで、新しい統計を収集する前にFND_STATTAB表の既存の統計をバックアップでき(バックアップ・フラグ・パラメータの値がBACKUPの場合のみ)、必要に応じて復元できます。

## 表統計の収集

「表統計の収集」プログラムは表、列および索引の統計を収集します。このプログラムはFND_STATS.GATHER_TABLE_STATSパッケージを呼び出します。パラメータは表B1に示されています。これは、時間の制約によりメインの統計収集から除外されている非常に大規模な表の統計を収集するのに役立ちます。また、1つのSQL文のパフォーマンスを調査する際に小さな表セットの統計を更新するのにも役立ちます。

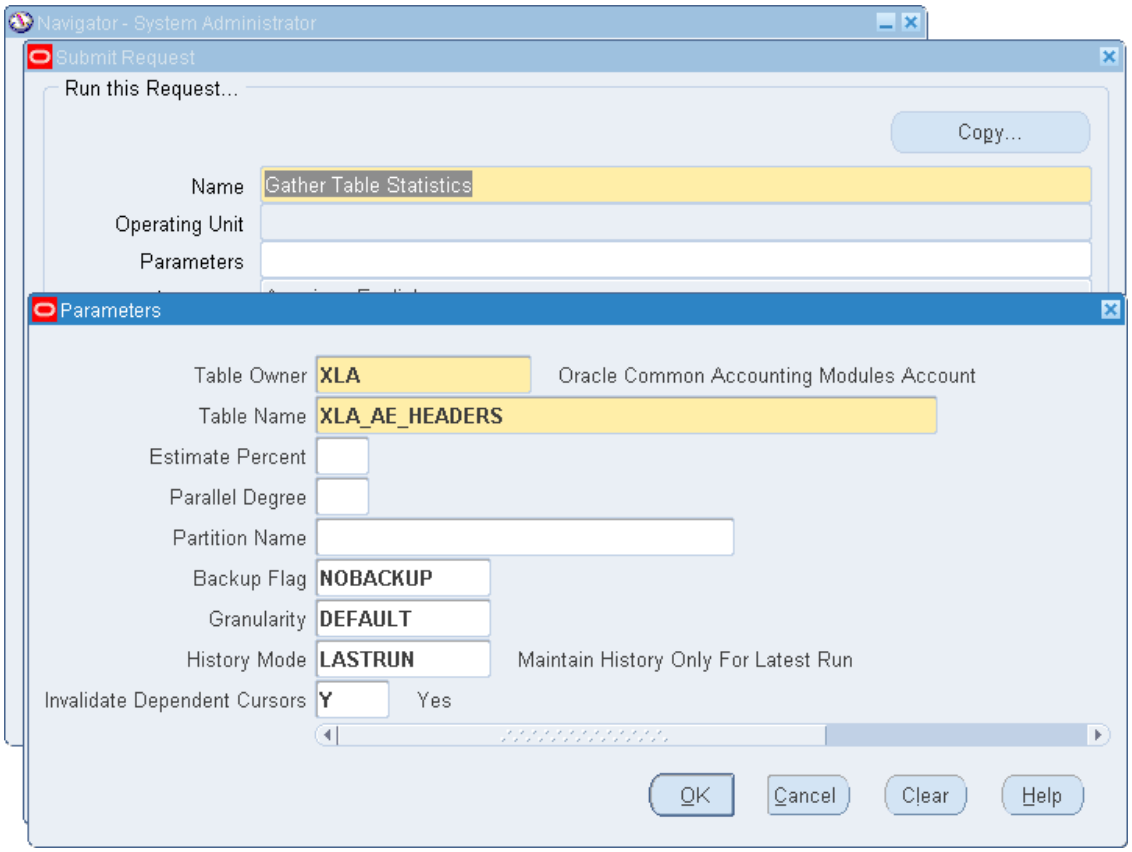


**図1:**
**Oracle E-Business Suite -** ***「表統計の収集」*** **コンカレント要求発行ウィンドウ**

図1は、表統計の収集の発行の例を示しています。「評価率」が空白になっています。パラメータについて次に説明します。

**表B1. FND_STATS.GATHER_TABLE_STATSのパラメータ**

| パラメータ | 説明 |
|---|---|
| 所有者名 | 表の所有者です。 |
| 表名 | 表名です。 |
| 評価率 | サンプリング率です。空白のままにした場合、11gではデフォルトでdbms_stats.auto_sample_sizeとなり、10g以前ではデフォルト値10が使用されます。有効な範囲は0から100までです。 |

| | |
|---|---|
| 並列度 | 統計の収集時に使用される並列度です。並列度が指定されていない場合、デフォルトでは parallel_max_serversおよびcpu_countのいずれか小さい方になります。 |
| パーティション名 | パーティション名です。 |
| バックアップ・フラグ | 統計をバックアップするにはBACKUPを指定します。(BACKUP/NOBACKUP) |
| 粒度 | 収集する統計の粒度です(パーティション表の場合のみ関係します)。有効な値は次のとおりです。<br>• DEFAULT - グローバル統計およびパーティションレベルの統計を収集します。<br>• SUBPARTITION - サブパーティションレベルの統計を収集します。<br>• PARTITION - パーティションレベルの統計を収集します。<br>• GLOBAL - グローバル統計を収集します。<br>• ALL - すべての(サブパーティション、パーティションおよびグローバル)統計を収集します。 |
| 履歴モード | このパラメータでは履歴レコードの数が制御されます。「履歴モード」セクションでも説明されていますが、このオプションにはLASTRUN、FULLおよびNONEがあります。デフォルトはLASTRUNです。 |
| 依存カーソルの無効化 | これは、依存カーソルを無視するかどうかを示します。このパラメータはOracle 9iR2より前の場合は無視されます |

## スキーマ統計の収集

「スキーマ統計の収集」プログラムでは、スキーマのすべてのオブジェクトについて、表、列および索引の統計を収集します。このプログラムはFND_STATS.GATHER_SCHEMA_STATSパッケージを呼び出します。パラメータについては表B2に示されています。これは、Oracle E-Business Suiteの統計を更新するために実行する推奨プロセスです。このペーパーのメイン部分で、様々な考慮事項が説明されています。使用可能な時間帯を収集時間が超過する場合、非常に大規模な表を除外し、それについては別の時にスケジュールするという方法が場合によっては役立ちます。

操作中にこのプロシージャが失敗した場合、失敗した要求の要求IDを指定して再起動できます。FND_STATS_HIST表から問合せ可能な要求IDは、プログラムがコンカレント・マネージャから開始されたときに取得されるか、FND_STATS.GATHER_SCHEMA_STATSが直接またはカスタム・スクリプト内で使用されている場合に自動的に作成されます。

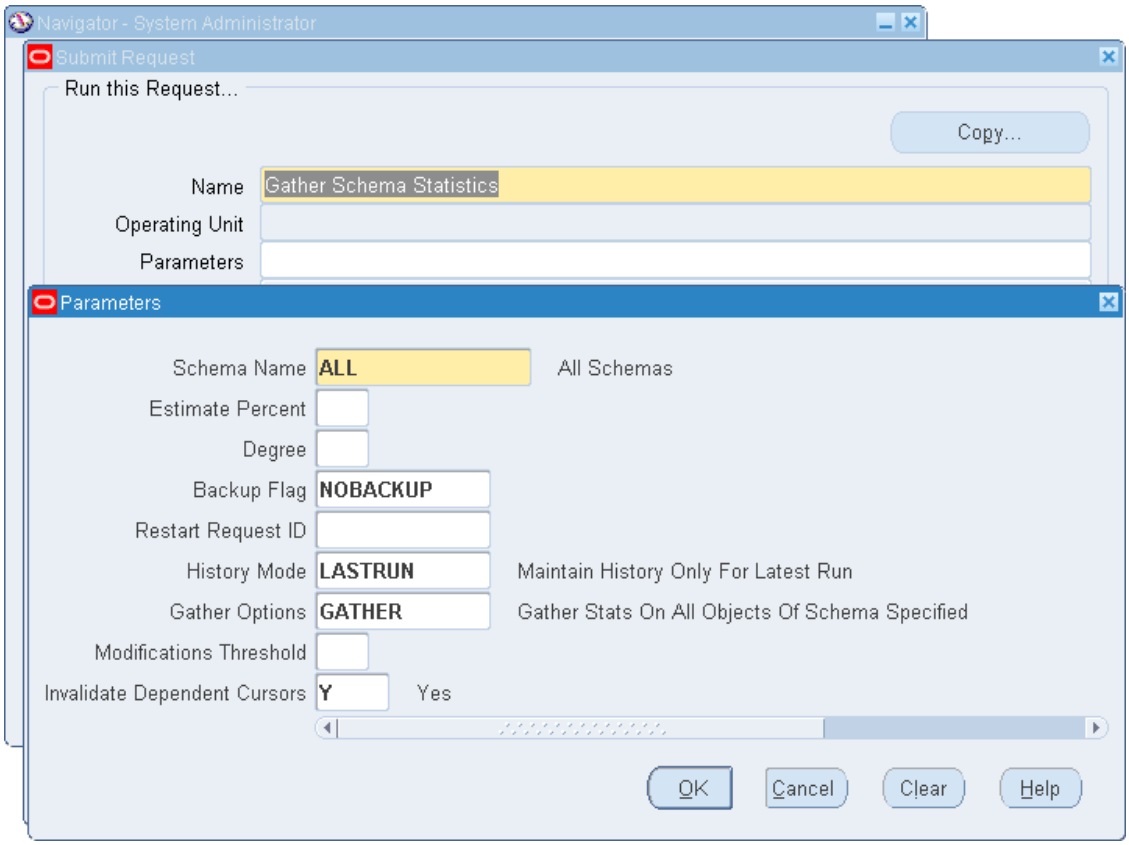


**図2:**
**Oracle E-Business Suite - *「スキーマ統計の収集」*コンカレント要求発行ウィンドウ**

図2は、スキーマ統計の収集の発行の例を示しています。「評価率」が空白になっています。パラメータについて次に説明します。

**表B2: FND_STATS.GATHER_SCHEMA_STATSのパラメータ**

| パラメータ | 説明 |
|---|---|
| スキーマ名 | ALLを指定すると、FND_PRODUCT_INSTALLATIONS表にリストされているすべての登録済Oracle E-Business Suiteスキーマについて統計が収集されます。 |
| 評価率 | サンプリング率です。空白のままにした場合、11gではデフォルトでdbms_stats.auto_sample_sizeとなり、10g以前ではデフォルト値10が使用されます。有効な範囲は0から100までです。 |

| | |
|---|---|
| 並列度 | 統計の収集時に使用される並列度です。並列度が指定されていない場合、デフォルトでは parallel_max_serversおよびcpu_countのいずれか小さい方になります。 |
| バックアップ・フラグ | 統計をバックアップするにはBACKUPを指定します。(BACKUP/NOBACKUP) |
| 要求IDの再起動 | スキーマ統計の収集の実行に失敗したかまたは停止した場合、このパラメータを使用して再送信でき、失敗した実行がどこで停止したかが判別されます。 |
| 履歴モード | このパラメータでは履歴レコードの数が制御されます。「履歴モード」セクションでも説明されていますが、このオプションにはLASTRUN、FULLおよびNONEがあります。デフォルトはLASTRUNです。 |
| 収集オプション | このパラメータでは、統計収集のためにオブジェクトがどのように選択されるかを指定します。<br>• GATHER (デフォルト) - すべてのスキーマ表および索引が統計収集の対象として選択されます。<br>• GATHER_AUTO - 変更率がmodpercentを超えたスキーマschemanameの表が統計収集の対象として選択されます。このオプションを使用するには、表モニターが有効化されている必要があります。<br>• GATHER_EMPTY - 統計のない表および索引についてのみ統計が収集されます。<br>• LIST_AUTO - GATHER_AUTOが使用されている場合の統計収集のためにオブジェクトのリストのみを提供します。<br>• LIST_EMPTY - GATHER_EMPTYが使用されている場合の統計収集のためにオブジェクトのリストのみを提供します。 |
| 変更しきい | このオプションはGATHER AUTOおよびLIST AUTOオプションにのみ適用可能です。このパラメータでは、AUTO統計収集で抽出されるために、表で行われている必要がある変更の率を指定します。 |
| 依存カーソルの無効化 | これは、依存カーソルを無視するかどうかを示します。デフォルトは'Y'です。 |

Oracle E-Business Suiteにおける統計収集のベスト・プラクティス
2013年8月

著者: Deepak Bhatnagar、Mohammed Saleem、Andy Tremayne

編集者: Robert Farrington

Oracle E-Business Suiteパフォーマンス・グループ

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問い合わせ窓口:
電話: +1.650.506.7000
FAX: +1.650.506.7200

oracle.com