異種環境 AWS 向けのかさばる Oracle データベースの への移行

AWS 規範ガイダンス

AWS 規範ガイダンス: 異種環境 AWS 向けのかさばる Oracle データベースの への移行

Table of Contents

かさばる Oracle データベースをクロスプラットフォーム環境 AWS の に移行する

Incheol Roh、Amazon Web Services (AWS)

2021 年 3 月 (ドキュメント履歴)

100 TB を超える Oracle データベースをオンプレミスからAmazon Web Services に移行する場合 (AWS) クラウド、移行のダウンタイムは重大な要因です。特に、システムがグローバルエンタープライズリソースプランニング (ERP) や製造実行システム (MES) などのミッションクリティカルなシステムである場合は、ビジネスの継続性のためにダウンタイムを可能な限り削減する必要があります。

「Oracle データベースの移行戦略」で説明されているように、エンディアン形式が異なるシステム間で Oracle データベースを移行 AWSする方法は多数あります。これらのアプローチの1つは、Oracle クロスプラットフォーム・トランスポータブル・テーブルスペース (XTTS) であり、これを使用して、移行のダウンタイムを数日から数時間に短縮できます。

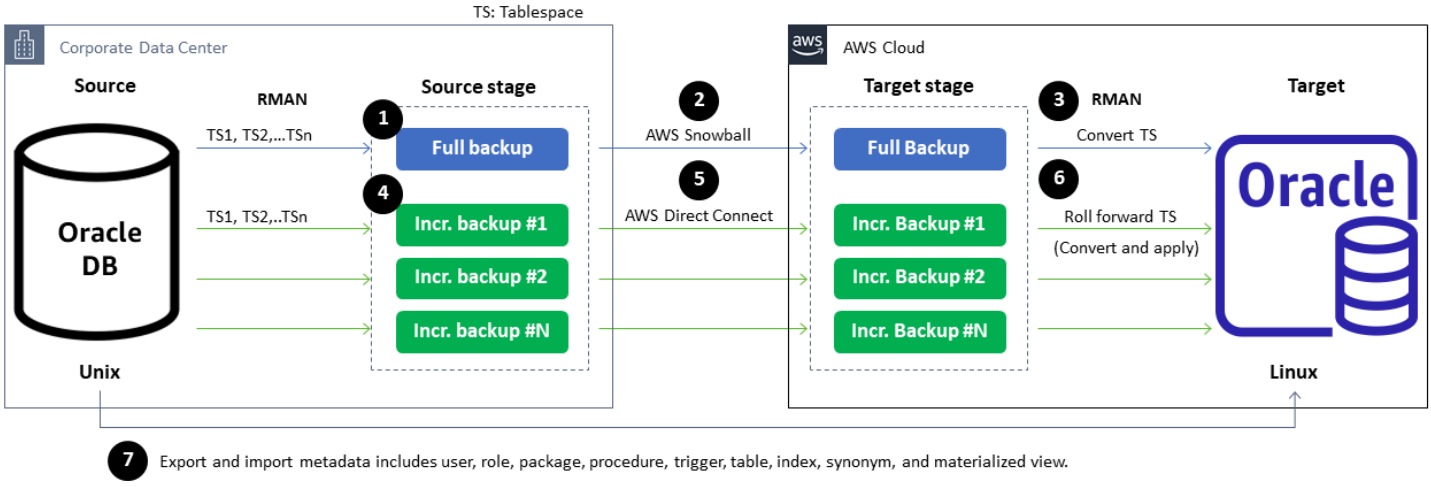
Oracle XTTS と Oracle Recovery Manager (RMAN) 増分バックアップを組み合わせることで、さまざまなエンディアン形式を実行しているプラットフォーム間でデータを移動するために必要なダウンタイムを大幅に削減できます。

このガイドでは、を使用する方法について説明します。AWS Snowball Edge、AWS Direct Connect、および Amazon FSx for Lustre Oracle XTTS と RMAN インクリメンタルバックアップを使用します。このアプローチの目標は、非常に大きなデータセットがある環境での移行のダウンタイムを最小限に抑えることです。

概要

これは、Snowball Edge、 Direct Connect FSx for Lustre で Oracle XTTS および RMAN 増分バックアップ AWS を使用するように Oracle データベースを移行する概念的なプロセスです。

次の図は、さまざまなエンディアン形式にわたる Oracle データベースの高レベルの移行ステップを示しています。



1. すべての表領域のフルバックアップを作成します。
2. Snowball Edge を使用して、バックアップをソースステージからターゲットステージに移動します。
3. 表領域をターゲット・データベースに変換します。
4. 増分バックアップを作成します。
5. Direct Connect を使用して、ソースステージからターゲットステージに増分バックアップを転送します。
6. 増分バックアップをロールフォワードし、変換してターゲット・データベースに適用します。
7. 転送されるすべての表領域のメタデータをエクスポートおよびインポートします。

カットオーバーの前に、以下を実行することでダウンタイムを最小限に抑えることができます。

- USER、PACKAGE_SPEC、PACKAGE_BODY、PROCEDURE、および FUNCTION を含む非セグメントベースのオブジェクトのメタデータをエクスポート、およびインポートします。
- フル・バックアップと増分バックアップの並列性を高めます。
- データファイルを変換する

- 移行時のロールフォワードバックアップです。

Oracle 社のドキュメント「クロスプラットフォーム増分バックアップを使用したトランスポータブル表領域のダウンタイムの短縮」(2471245.1) では、RMAN 増分バックアップで Oracle XTTS を使用する方法について説明しています。このドキュメントには、要件と推奨事項の詳細も含まれています。このドキュメントでは、オンプレミス環境から Oracle on AWS へ Oracle データベースを移行する方法や、ダウンタイムを最小化するために各移行ステップを並列化する方法については説明されていません。

このガイドでは、各フェーズを並列化し、非常に大きなデータサイズのミッションクリティカルなシステム環境での移行のダウンタイムを最小限に抑える方法について説明します。

初期セットアップ· フェーズに続いて、Oracle XTTS を RMAN 増分バックアップとともに使用するためのハイレベル· ステップには、次のフェーズが含まれます。

フェーズ 1 - 準備段階のフェーズ

準備フェーズは、次のステップで構成されます。

1. 表領域の最初のフル· バックアップ (レベル=0) がソース· データベースでソース· ステージ、NAS ストレージに作成されます。
2. バックアップコピーは、Snowball Edge を使用して、Amazon Simple Storage Service (Amazon S3) と統合された FSx for Lustre であるターゲットステージに転送されます。
3. バックアップされた表領域はリストアされ、リトルエンディアン形式でターゲットデータベースに変換されます。

このフェーズのステップは、移行中に一度だけ実行されます。転送されるデータは、このフェーズ中にソースデータベースで完全にアクセスできます。

フェーズ2 - ロールフォワードフェーズ

ロールフォワードフェーズは、次のステップで構成されます。

1. ソース· データベースからソース· ステージへの増分バックアップを取得します。
2. 増分バックアップのコピーは、ターゲット· ステージに Direct Connectで転送されます。

3. 増分バックアップコピーは、リトルエンディアン形式でターゲットデータベースに変換されます。このコピーは、ロールフォワードステップと呼ばれる初期ターゲットデータベースに適用されます。

このフェーズは複数回実行できます。増分バックアップを繰り返すたびに、バックアップにかかる時間は短くなり、宛先データファイルのコピーはソース· データベースと同じ状態になります。フェーズ 1 と同様に、転送されるソース· データはこのフェーズで完全にアクセス可能です。

フェーズ 3 - 転送フェーズ

3 番目のフェーズには、次のステップが含まれます。

1. 転送される表領域は、読み取り専用に変更されます。
2. 最終的な増分バックアップがソース· データベースから取得されます。
3. メタデータがエクスポートされます。
4. バックアップが転送され、宛先に適用されます。
5. オブジェクト· メタデータがインポートされます。

この時点で、宛先データベースのシステム変更番号 (SCN) は、ソース· データベースのシステム変更番号 (SCN) と一致しています。

転送可能な表領域のメタデータを転送元データベースからエクスポートし、転送先データベースへインポートします。メタデータには、ユーザー、ロール、パッケージ、プロシージャ、関数、テーブル、およびインデックスの情報が含まれます。

最後に、アプリケーションからの宛先データベースへのフル· アクセスのために、表領域は読み取り/書き込み可能になります。

このフェーズの後に、検証フェーズが続きます。

移行フェーズ

このガイドでは、OracleシングルインスタンスとOracle RACの両方をソースデータベースおよびターゲットデータベースとして移行する方法について説明します。ソースとターゲットがOracle RACの場合、各移行ステップの並列度を最大化できます。

フェーズ 0 — 初期セットアップフェーズ

1. Oracle データベース 12.1.0.2 以降をターゲットである Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにインストールします。
2. ターゲットデータベースを確認する。

   ターゲットのデータベースインスタンスのタイムゾーンと文字セットがソースデータベースと同じかどうかを確認する必要があります。そうしないと、この移行方法は失敗します。

   ソースとターゲットの両方のタイムゾーンのバージョンが同じであることを確認するには、次のコマンドを実行します。

```
SQL> select * from v$timezone_file;
FILENAME            VERSION    CON_ID
----------------------   -------------   ----------
timezlrg_14.dat        14               0
```

   DB 文字セットと各国文字セットの両方がソースとターゲットで同じであることを確認するには、次のコマンドを実行します。

```
SQL> select * from nls_database_parameters
         where parameter in ('NLS_CHARACTERSET','NLS_NCHAR_CHARACTERSET');
PARAMETER                                    VALUE
-------------------------------------------   --------------------
NLS_NCHAR_CHARACTERSET          UTF8
NLS_CHARACTERSET                     AL32UTF8
```

3. Oracle XTTS ユーティリティをセットアップします。

   ソースシステムで、Oracle 文書2471245.1からスクリプトファイルrman_xttconvert_VER4.zipをダウンロードして解凍します。特定の設定を含む `xtt.properties` ファイルのパラメータを

変更します。次に、xtt.properties そして xttdriver.pl スクリプトを宛先システムへコピーします。

フェーズ 1 — 準備フェーズ (ソースデータベースはオンラインのまま)

このフェーズでは、転送される表領域のデータファイルがソースシステムにバックアップされます。バックアップコピーは、宛先システムに転送され、xttdriver.pl スクリプトを実行することで復元されます。

ステップ 1: ソースシステムでフル (レベル=0) の表領域バックアップを行う

バックアップ時間を短くするには、xtt.properties そして xttdriver.pl ファイルを複数のディレクトリにコピーします。各ディレクトリの下の各 xtt.properties ファイルで、tablespaces パラメータの表領域の名前を入力します。次に、各ディレクトリの下で、xttdriver.pl と --backup オプションを実行します。このコマンドは、ターゲットデータベースのインストール時に作成された SYSTEM、SYSAUX、UNDO、および TEMP 以外のすべての表領域を転送します。

例えば、各 xtt01~xtt04 ディレクトリはすべての xtt スクリプト (xtt.properties そして xttdriver.pl) に対して異なる値が tablespaces= パラメータにあり、それは xtt.properties ファイル内にあります。編集する tablespaces パラメーターがある各 xtt.properties ファイルは、各表領域グループ内のデータファイルの合計サイズが同じようになるように、表領域グループを分割することを検討してください。

このガイドでは、4 つの表領域グループがあることを前提としています。ソースデータベースが Oracle シングルインスタンスの場合は、xtt01~04 ディレクトリすべてを作成します。ソースデータベースが Oracle RAC の場合は、各 Oracle RAC サーバ内の各 xtt ディレクトリを作成できます。

```
[oracle@erp expimp]$ ls
out  xtt01  xtt02  xtt03    xtt04
[oracle@erp out]$ ls
out01 out02 out03 out04
```

次の表は、tablespaces= パラメータが、xtt.properties ファイルに表示されている例です。

xtt01	tablespaces=APPS_TS_TX_DATA
xtt02	tablespaces=APPS_TS_TX_IDX
xtt03	tablespaces=APPS_TS_SUMMARY
xtt04	tablespaces=APPS_CALCLIP, APPS_OMO, APPS_TS_ARCHIVE, APPS_TS_DISCO, APPS_TS_DISCO_OLAP , APPS_TS_INTERFACE, APPS_TS_M EDIA, APPS_TS_NOLOGGING, APPS_TS_QUEUES, APPS_TS_SEED…

既存のデータファイルのバックアップコピーが削除されないようにするには、xttdriver.pl が並行して実行されている場合は、xttdriver.pl の次の行をコメントアウトしてください。

```
   if (@present)
   {
      ## Try deleting any existing copied datafiles
##Comment out it for executing rman command in parallel by AWS
##        system("\\rm -f $dfcopydir/*.tf");
      sleep 5;
   }
```

xttdriver.pl を使用して、ソースシステムの RMAN バックアップを作成します。

ソースシステムが Oracle RAC の場合、各 Oracle RAC インスタンスで同時に実行できます。

xttdriver.pl の出力は、TMPDIR 環境変数に格納されます。各 xttdriver.pl コマンドを --backup オプションと実行し、--debug オプションを使用して、デバッグモードをオンにします。

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --backup --debug 3
```

この例では、<nn> は表領域グループを表します。表領域グループが 4 つある場合は、<nn> は01、02、03 および 04 になります。

ステップ 2: フルバックアップコピーを宛先システムに転送する

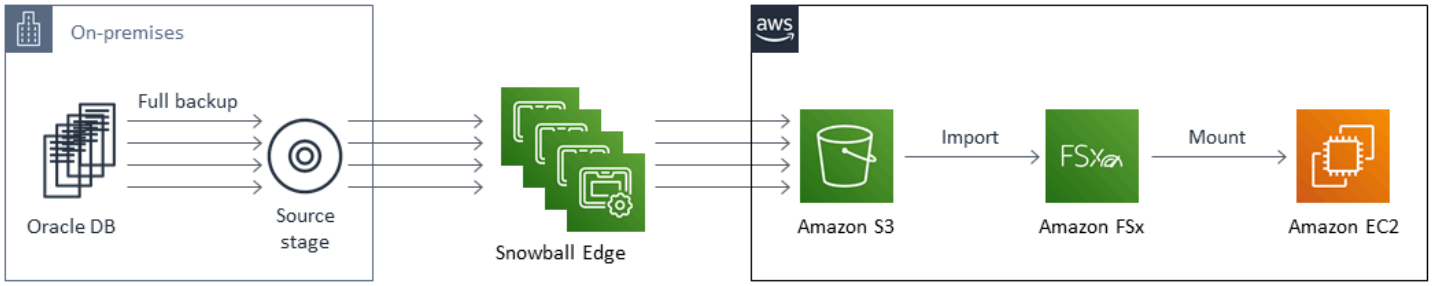
バックアップコピーを宛先システムに転送するには、次の 2 つのオプションのいずれかを使用します。

オプション 1 – の使用 AWS Snowball Edge

パス: ソースデータベース -> ソースステージ -> AWS Snowball Edge -> Amazon S3 -> FSx for Lustre

> **Note**
>
> AWS Snowball Edge は新規顧客には利用できなくなりました。新規のお客様は、オンライン転送、安全な物理的な転送のためのAWS データ転送ターミナル、または AWS Partner ソリューションAWS DataSyncを検討する必要があります。エッジコンピューティングについては、「」を参照してくださいAWS Outposts。

次の図は、Snowball エッジを使用したフルバックアップの転送を示しています。



Snowball エッジは、大規模なデータを AWSに移動するために使用できる、丈夫な物理ストレージおよびコンピューティングデバイスです。Snowball エッジは、転送時間が長い、使用可能な帯域幅の不足、セキュリティに懸念があるなど、大規模データを転送するときに発生する可能性のある課題を解決するのに役立ちます。

データファイルのバックアップコピー全体が Snowball エッジによって Amazon S3 に転送されます。ソースシステムのサイズが 100 TB を超える場合は、複数の Snowball エッジデバイスを使用して転送時間を短縮する必要があります。

Amazon FSx for Lustre は Amazon S3 と深く統合しています。S3 バケットにリンクされた FSx for Lustre ファイルシステムを数分で作成できます。Amazon S3 データリポジトリにリンクすると、FSx for Lustre ファイルシステムによって Amazon S3 オブジェクトがファイルとして透過的

に表示されます。実際の環境では、FSx for Lustreのパフォーマンスは、ストレージ容量、各ファイルのサイズ、およびファイルシステムへのファイルの分散具合に依存します。FSx for Lustre をターゲットステージストレージとして使用する場合、Amazon S3 から Amazon Elastic Block Store（Amazon EBS）または Amazon Elastic File System を表領域のバックアップコピーに復元する必要はありません。

1. S3 バケットを FSx for Lustre にインポートします。

   Snowball エッジを使用して、ソースステージエリア（例えば、`src_backups`)を S3 バケット（例えば、`s3-src-backups`) に転送して保存します。

   データファイルバックアップコピーの宛先ステージエリアとして FSx for Lustre ファイルシステム (例えば、`dest_backups`) を作成します。

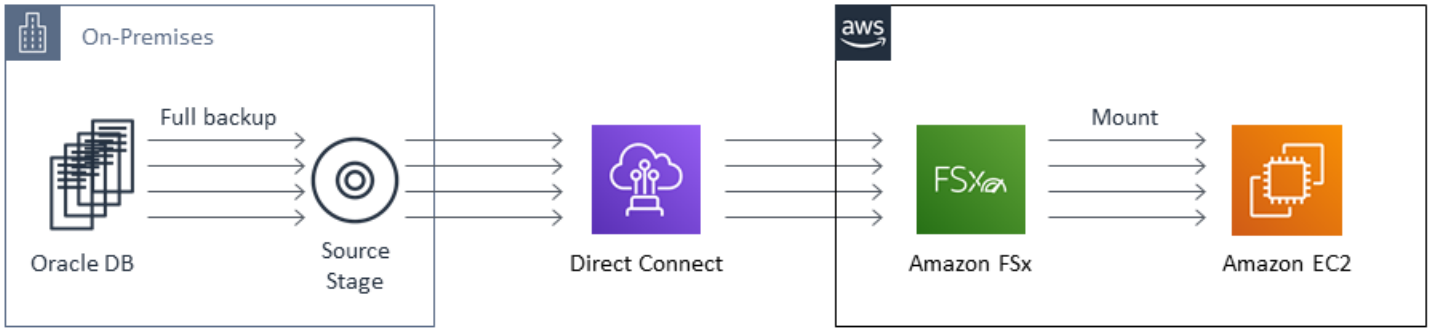
   宛先システム (Amazon EC2) にオープンソース Lustre クライアントをインストールします。詳細については、「Amazon FSx for Lustre ユーザーガイド」を参照してください。

   Lustre クライアントをインストールしたら、FSx for Lustre ファイルシステムを宛先システム（Amazon EC2）にマウントします。
2. `res.txt` は、ソースの `$TMPDIR` から宛先 `$TMPDIR` へ転送される。

オプション 2 – の使用 AWS Direct Connect

パス :ソースデータベース -> ソースステージ -> AWS Direct Connect -> FSx for Lustre

次の図は、 AWS Direct Connectを使用したフルバックアップの転送を示しています。



Direct Connect では、データセンター、オフィス、またはコロケーション環境と の間にプライベートネットワーク接続を作成できます AWS。これらのプライベートネットワーク接続は、ネットワークコストを削減し、スループットを向上させ、一貫したエクスペリエンスを提供します。

データファイルのバックアップコピーは、ソースシステムから Amazon FSx for Lustre ファイルシステムがマウントされている宛先システム (Amazon EC2) に直接転送できます。 Direct Connectの高帯域幅に対応できる場合、このオプションは Snowball エッジオプションよりも高速です。

データファイルのバックアップコピーが Snowball Edge または によって転送されたら Direct Connect、ターゲットステージエリアの FSx for Lustre ファイルシステムでそれらを確認できます。

ステップ 3: 完全バックアップコピーを復元し、データファイルを変換する

フルバックアップコピーを復元し、データファイルを宛先システムエンディアン形式に変換します。復元および変換の時間を短縮するには、`xttdriver.pl` コマンドを表領域グループごとの `--restore` オプションで実行します。

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --restore --debug 3
```

ステップが完了すると、データファイルは、宛先システム上の `xtt.properties` ファイルで、定義された dest_datafile_location に配置されます。

フェーズ 2 — ロールフォワードフェーズ (ソースデータベースはオンラインのまま)

ロールフォワードフェーズを必要な回数繰り返して、宛先データファイルをソースデータベースに追いつかせることができます。

ロールフォワードフェーズは、次のステップで構成されます。

1. ソースデータベースから増分バックアップを作成します。
2. バックアップを宛先システムに転送します。
3. バックアップを宛先システムのエンディアン形式に変換します。
4. 変換された宛先データファイルのコピーにバックアップを適用して、ロールフォワードします。

増分バックアップを繰り返すたびに、バックアップにかかる時間は短くなり、宛先データファイルのコピーはソースデータベースと同じ状態になります。

ステップ 1: 増分バックアップを並列に実行する

ソースシステムで転送される表領域グループの増分バックアップを並列に実行します。

この手順では、xtt.properties ファイルにリストされているすべての表領域に対して増分バックアップを作成します。

ソースシステムでBLOCK CHANGE TRACKING機能を有効化できれば、増分バックアップの時間を大幅に短縮できます。

次のコマンドは、フルバックアップ後の次の SCN を自動的に認識します。

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --backup --debug 3
```

ステップ 2: 増分バックアップと res.txt ファイルを宛先システムに転送する

十分な帯域幅がある場合は、 Direct Connectを使用して転送時間を短縮できます。

src_scratch_location および dest_scratch_location 間で増分バックアップを転送する res.txt ファイルを、$TMPDIR のソースシステムと $TMPDIR の宛先システム間で転送する。

複数の増分バックアップを行う場合は、最後の増分バックアップの後に res.txt ファイルをコピーしなければ、宛先システムで適用することはできません。

```
[source]$ scp $TMPDIR/res.txt oracle@[dest]:/u01/oracle/expimp/out/out<nn>
[source]$ scp <src_scratch_location>/* oracle@[dest]:<dest_scratch_location>
```

ステップ 3: 増分バックアップを変換して適用する

増分バックアップを宛先システムのエンディアン形式に変換し、宛先システムの宛先データファイルのコピーに適用します。

このガイドでは、4 つの表領域グループがあることを前提としています。各 xttdriver.pl コマンドを、表領域グループごとに --restore オプションを指定して実行します。

このステップでは、Oracle XTTS ユーティリティがターゲットデータベースを再起動します。コマンドを並行して実行するには、Perl スクリプト xttdriver.pl で次のカスタマイズを使用する必要があります。

1. 以下の行をコメントアウトします。

   - 4867 行目： `my $outputstart = `sqlplus -L -s \"/ as sysdba\" \@xttstartupnomount.sql`;`
   - 4868 行目： `checkError ("Error in executing xttstartupnomount.sql", $outputstart);`
   - 4992 行目： `my $outputstart = `sqlplus -L -s \"/ as sysdba\" \@xttdbopen.sql`;\`

2. NOMOUNT ステータスのターゲット DB を作成します。

```
sqlplus / as sysdba @xttstartupnomount.sql
```

3. 増分バックアップのロールフォワードを並行して実行します。

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --restore --debug 3
```

4. すべての増分バックアップをロールフォワードしたら、OPEN ステータスのターゲットDBを設定します。

```
sqlplus / as sysdba @xttdbopen.sql
```

この時点で、ソースデータベースと宛先データベースの SCN が十分に近くなるまで、フェーズ 2 を繰り返すことができます。特定のターゲットのダウンタイムに応じて、フェーズ 3 に進むか、フェーズ 2 を繰り返すかを決定する必要があります。

フェーズ 3 (転送フェーズ) のダウンタイムを削減するために、USER、PACKAGE、PROCEDURE、および FUNCTIONといった非セグメントベースのオブジェクトのメタデータを エクスポートおよびインポートすることができます。ソースデータベースを READ ONLY に設定する前に行ってください。

ソースデータベース内のデータベースオブジェクトの数が非常に多い場合 (数十万)、メタデータのエクスポートとインポートには数時間かかります。この場合は、メタデータを エクスポートすることを検討します。

非セグメントベースのオブジェクトのエクスポートは、ソースシステム上で読み取り/書き込みで実行されます。次に、ソースシステムが READ ONLY に設定される前に、オブジェクトを宛先システム

にインポートする必要があります。このとき、PACKAGE、PROCEDURE、および FUNCTION といったデータベースのソースオブジェクトを変更しないようにします。

これは、USER、PACKAGE_SPEC、PACKAGE_BODY、PROCEDURE、および FUNCTION など、非セグメントベースのオブジェクトのメタデータを エクスポートするために使用されるダンプパラメータファイルの例です。

```
directory=dmpdir
dumpfile=xttsmsc%U.dmp
full=y
filesize=1048576000
logfile=expmsc.log
metrics=y
exclude=TABLE,INDEX,CONSTRAINT,COMMENT,
MATERIALIZED_VIEW,MATERIALIZED_VIEW_LOG,SCHEMA_CALLOUT
```

メタデータを エクスポートするには、以下のコマンドを使用します。

```
SQL> create directory dmpdir as <location>;
expdp system/<system password> parfile=<parameter file>
```

これは、非セグメントのオブジェクトのメタデータをインポートするための ダンプパラメータファイルの例です。

```
directory=dmpdir
dumpfile=xttsmsc%U.dmp
full=y
logfile=impmsc1.log
EXCLUDE=TABLESPACE, PROCOBJ, RLS_CONTEXT, RLS_GROUP, RLS_POLICY, TABLESPACE_QUOTA
metrics=y
remap_tablespace=
APPS_TS_ARCHIVE:SYSTEM,
APPS_TS_INTERFACE:SYSTEM,
APPS_TS_SEED:SYSTEM,
APPS_TS_SUMMARY:SYSTEM,
…..
```

このとき、宛先システムには、SYSTEM、SYSAUX、UNDO および TEMP 以外の表領域は存在しません。したがって、インポートするすべてのオブジェクトを SYSTEM 表領域に再マップする必要があります。

メタデータをインポートするには、以下のコマンドを使用します。

```
SQL> create directory dmpdir as <location>;
impdp system/<system password> parfile=<parameter file>
```

これで、作成されたオブジェクトが宛先データベースに表示されます。

```
SQL> select object_type, count(*) from dba_objects group by object_type order by
 count(*) desc;
OBJECT_TYPE                    COUNT(*)
------------------------------ ----------
SYNONYM                           89327
PACKAGE                            55670
PACKAGE BODY                     54447
VIEW                                  41378
JAVA CLASS                         31978
SEQUENCE                          12766
…..
```

SYSTEM、SYSAUX、UNDO、および TEMP 以外の表領域はありません。USER オブジェクトは、USERSをデフォルトの表領域として作成されます。フェーズ 3 では、転送可能な表領域が作成され、USER オブジェクトが元のデフォルト表領域で変更されます。

フェーズ 3 — 転送フェーズ (ソースデータベースは読み取り専用)

このフェーズの間、ソースシステムは読み取り専用になります。宛先システム上のデータファイルは、最終的な増分バックアップを適用することにより、ソースシステムと整合性が保たれます。次に、ソースシステムからオブジェクトメタデータを エクスポートし、宛先システムにインポートします。

ステップ 1: ソースデータベースの表領域を読み取り専用にする

SYSDBA として、転送されるすべての表領域を READ ONLY ソースシステム上で作成します。

ダウンタイムを削減するために、次の 2 つの手順を同時に実行できます。

ステップ 2: 最終的な増分バックアップを作成する

ソースシステムで、転送する表領域の最終増分バックアップを作成します。

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --backup --debug 3
```

この手順は、「ORA-20001: TABLESPACE (S) は読み取り専用」というエラーを返します。エラーは想定されており、無視してかまいません。

ステップ 3: メタデータをエクスポートする

転送可能なテーブルスペースのメタデータをソースデータベースからエクスポートします。

これは、転送可能な表領域のメタデータをエクスポートするためのパラメータファイルの例です。

```
directory=dmpdir
metrics=y
dumpfile=xttsmeta%U.dmp
filesize=1048576000
logfile=expxtts.log
transport_tablespaces=
APPS_TS_ARCHIVE,
APPS_TS_INTERFACE,
APPS_TS_MEDIA,
APPS_TS_NOLOGGING,
….
exclude=table_statistics,index_statistics
```

さらに、ソースシステムに多数のテーブルとインデックスがある場合、統計を除外することで、エクスポート中の時間を節約できます。転送可能な表領域をインポートした後、宛先システムに統計をインポートします。

expdp を実行する前に、ソースシステムに格納されているダンプファイルにデータベースディレクトリを作成します。

```
SQL> create directory dmpdir as <location>;
expdp system/<system password> parfile=<parameter file>
```

次の2つのステップは、RMAN増分バックアップを使用したクロスプラットフォームの転送可能な表領域の最後のステップです。これらのステップは、順番に実行する必要があります。

ステップ 4: ファイルを転送し、最終的な増分バックアップを適用する

最終増分バックアップを転送し、ダンプファイルを宛先システムにエクスポートし、変換して最終増分バックアップを適用します。

Direct Connect を使用して、最終的な増分バックアップコピーと res.txt ファイルを宛先に転送します。VPN 接続を使用できますが、 を使用すると、十分な帯域幅があればダウンタイムが大幅に短縮 Direct Connect されます。

最終的な増分バックアップを復元するには、宛先システムで、次のコマンドを `--restore` オプションを使用して、表領域グループごとに実行します。

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl --restore --debug 3
```

ステップ 5: オブジェクトメタデータをインポートする

Oracle Data Pump を使用して、オブジェクトのメタデータを宛先システムにインポートします。次のコマンドを実行して、宛先システムの `transport_datafiles=` パラメータのデータファイルのリストを取得します。

```
cd /u01/oracle/expimp/xtt<nn>
export TMPDIR=/u01/oracle/expimp/out/out<nn>
$ORACLE_HOME/perl/bin/perl xttdriver.pl -e
```

前のコマンドを実行するたびに、`xttplugin.txt` ファイルを取得し、ファイルには `transport_datafiles=` パラメータがあります。`transport_datafiles=` をすべての `xttplugin.txt` ファイルから一行でマージし、データファイルリストをインポートメタデータのパラメータファイルの `transport_datafiles` 引数に入れます。

次のコードスニペットは、宛先システムに転送可能な表領域をインポートするためのパラメータファイルを示しています。

```
directory=dmpdir
metrics=y
dumpfile=xttsmeta%U.dmp
logfile=impxtts.log
exclude=TYPE
```

```
transport_datafiles= '+EBSDATA/APPS_TS_TX_DATA_2.dbf','+EBSDATA/
APPS_TS_TX_DATA_11.dbf','+EBSDATA/APPS_TS_TX_DATA_22.dbf','+EBSDATA/
APPS_TS_TX_DATA_183.dbf','+EBSDATA/APPS_TS_TX_DATA_204.dbf','+EBSDATA/
APPS_TS_TX_DATA_219.dbf','+EBSDATA/APPS_TS_TX_DATA_227.dbf'…..
```

impdp を実行する前に、エクスポートダンプファイルの場所を指すデータベースディレクトリを作成します。

```
SQL> create directory dmpdir as <location>;
impdp system/<system password> parfile=<parameter file>
```

フェーズ 4 — 転送されたデータを検証する

ステップ 1: 表領域に破損がないかどうかをチェックする

このステップでは、転送された表領域が宛先データベースでのみ読み取られます。表領域が有効かどうかを検証するには、次のようにRMANコマンドを実行します。

```
sqlplus / as sysdba;
set feedback off
set pagesize 0
set heading off
spool tbs_val.sql;
select 'validate tablespace '||tablespace_name||' check
 logical;' from dba_tablespaces where tablespace_name not in
 ('SYSTEM','SYSAUX','TEMP','USERS','UNDOTBS1','UNDOTBS2','UNDOTBS3','UNDOTBS3','UNDOTBS4');
spool off;
exit;

--Remove the first and last line in the spool file, tbs_val.sql
rman target /
RMAN> @tbs_val.sql
```

さらに、テーブル、インデックス、シノニム、ビュー、パッケージオブジェクトなどのオブジェクトの数を確認することもできます。

ステップ 2. 宛先データベースに転送されたすべての表領域を読み取り/書き込みにする

```
sqlplus / as sysdba;
```

```
set feedback off
set pagesize 0
set heading off
spool tbs_rw.sql
select 'alter tablespace '||tablespace_name||' read write;' from dba_tablespaces where
 status='READ ONLY';
spool off;
exit;
--Remove the first and last line in the spool file, tbs_rw.sql
sqlplus / as sysdba;
SQL> @tbs_rw.sql
```

結論

このガイドでは、、、および Amazon FSx for Lustre で Oracle クロスプラットフォームのトランスポータブルテーブルスペースと RMAN 増分バックアップを使用して AWS Snowball Edge AWS Direct Connectダウンタイムを最小限に抑える方法を学習しました。

Oracle データベースをオンプレミスから に移行するときは AWS、次の変数を考慮してください。

- に必要なネットワーク帯域幅 Direct Connect
- FSx for Lustreの容量
- Oracle データベースのメタデータのサイズ
- 増分バックアップサイズ

このガイドで推奨される方法を使用して、Unix システムで実行されている 100 TB の Oracle データベースから に移行する場合 AWS、ダウンタイムを約 10 時間に短縮できます。

ドキュメント履歴

このガイドは、このドキュメントの大きな変更点をまとめたものです。今後の更新に関する通知を受け取る場合は、 RSS フィードをサブスクライブできます。

変更	説明	日付
–	概要と概要の「異種」の言及を変更しました。	2021 年 7 月 14 日
–	初版発行	2021 年 3 月 2 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。