이기종 환경을 AWS 위해 대용량 Oracle 데이터베이스를 로 마이그레이션

AWS 권장 가이드

AWS 권장 가이드: 이기종 환경을 AWS 위해 대용량 Oracle 데이터베이스를 로 마이그레이션

Table of Contents

플랫폼 간 환경을 AWS 위해 대용량 Oracle 데이터베이스를 로 마이그레이션

Incheol Roh, Amazon Web Services(AWS)

2021년 3월(문서 기록)

온프레미스에서 Amazon Web Services(AWS) 클라우드로 100TB를 초과하는 Oracle 데이터베이스를 마이그레이션할 때 마이그레이션 가동 중지 시간은 심각한 요인입니다. 특히 시스템이 글로벌 엔터프라이즈 리소스 계획(ERP) 또는 제조 실행 시스템(MES)과 같은 미션 크리티컬 시스템인 경우 비즈니스 연속성을 위해 가동 중지 시간을 최대한 줄이고자 합니다.

Strategies for Migrating Oracle Databases to에 언급된 대로 다양한 엔디안 형식의 시스템 간에 Oracle 데이터베이스를 마이그레이션하는 AWS 방법에는 여러 가지가 있습니다. 이러한 접근 방식 중 하나는 Oracle 교차 플랫폼 전송 가능 테이블스페이스(XTTS)이며, 이를 사용하여 마이그레이션 가동 중지 시간을 며칠에서 몇 시간으로 줄일 수 있습니다.

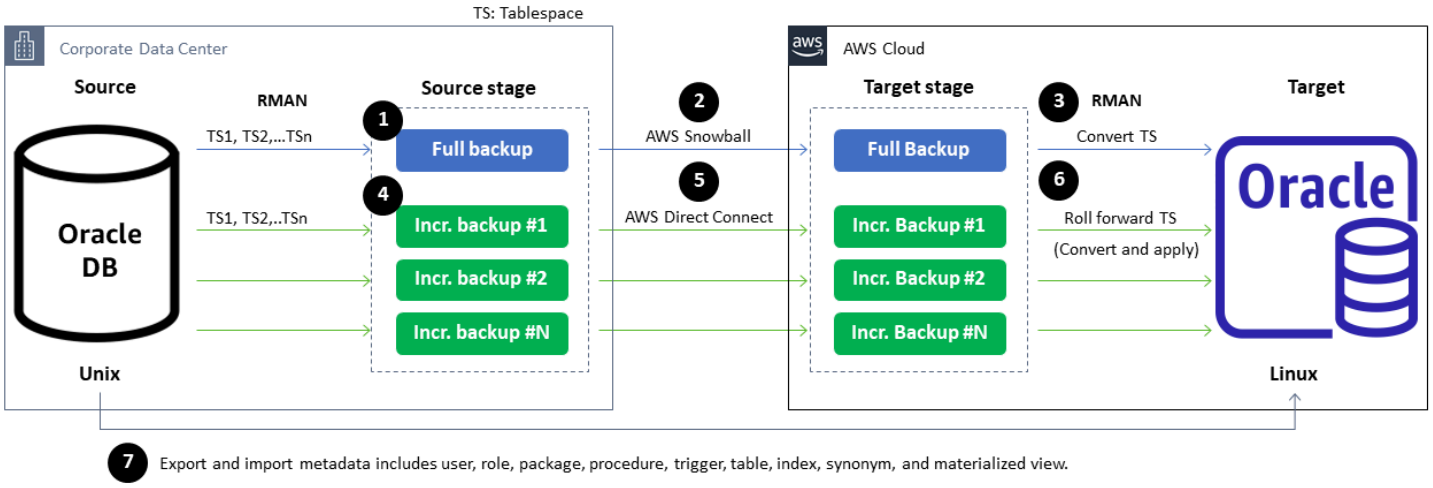
Oracle XTTS를 Oracle Recovery Manager(RMAN) 증분 백업과 결합하면 다양한 엔디안 형식을 실행하는 플랫폼 간에 데이터를 이동하는 데 필요한 가동 중지 시간을 크게 줄일 수 있습니다.

이 가이드에서는 RMAN 증분 백업과 함께 Oracle XTTS와 함께 AWS Snowball EdgeAWS Direct Connect, 및 Amazon FSx for Lustre를 사용하는 방법을 소개합니다. 이 접근 방식의 목표는 데이터 세트가 매우 큰 환경에서 마이그레이션 가동 중지 시간을 최소화하는 것입니다.

개요

이는 Snowball Edge 및 FSx for Lustre와 함께 Oracle XTTS Direct Connect및 RMAN 증분 백업을 AWS 사용하여 Oracle 데이터베이스를 로 마이그레이션하는 개념적 프로세스입니다.

다음 다이어그램은 다양한 엔디안 형식의 Oracle 데이터베이스에 대한 상위 수준 마이그레이션 단계를 보여줍니다.



1. 모든 테이블스페이스의 전체 백업을 만듭니다.
2. Snowball Edge를 사용하여 백업을 소스 단계에서 대상 단계로 이동합니다.
3. 테이블스페이스를 대상 데이터베이스로 변환합니다.
4. 증분 백업을 수행합니다.
5. Direct Connect 를 사용하여 소스 단계에서 대상 단계로 증분 백업을 전송합니다.
6. 증분 백업을 롤포워드하여 변환하고 대상 데이터베이스에 적용합니다.
7. 전송 중인 모든 테이블스페이스에 대한 메타데이터를 내보내고 가져옵니다.

전환 전에 다음을 수행하여 가동 중지 시간을 최소화할 수 있습니다.

- , , USER, 및를 포함한 비세그먼트 기반 객체의 메타데이터 내보내기 PACKAGE_SPEC PACKAGE_BODY PROCEDURE및 가져오기 FUNCTION
- 전체 백업 및 증분 백업을 위한 병렬 처리 증가
- 데이터 파일 변환
- 마이그레이션 중 백업 롤포워드

Oracle 문서 Reduce Transportable Tablespace Downtime using Cross Platform Incremental Backup(2471245.1)에서는 RMAN 증분 백업과 함께 Oracle XTTS를 사용하는 방법을 설명합니다. 이 문서에는 요구 사항 및 권장 사항에 대한 세부 정보도 포함되어 있습니다. 이 문서에서는 온프레미스 환경에서의 Oracle로 Oracle 데이터베이스를 마이그레이션하는 방법 AWS 이나 각 마이그레이션 단계를 병렬화하여 가동 중지 시간을 최소화하는 방법을 설명하지 않습니다.

이 가이드에서는 매우 큰 데이터 크기로 미션 크리티컬 시스템 환경에서 마이그레이션 가동 중지 시간을 최소화하여 단계를 병렬화하는 방법을 제공합니다.

초기 설정 단계 이후 RMAN 증분 백업과 함께 Oracle XTTS를 사용하기 위한 상위 단계에는 다음 단계가 포함됩니다.

1단계 - 준비 단계

준비 단계는 다음 단계로 구성됩니다.

1. 테이블스페이스의 초기 전체 백업(레벨=0)은 소스 데이터베이스에서 소스 단계인 NAS 스토리지로 가져옵니다.
2. 백업 사본은 Snowball Edge를 사용하여 Amazon Simple Storage Service(Amazon S3)와 통합된 FSx for Lustre인 대상 단계로 전송됩니다.
3. 백업 테이블스페이스는 복원되고 리틀 엔디안 형식으로 대상 데이터베이스로 변환됩니다.

이 단계의 단계는 마이그레이션 중에 한 번만 실행됩니다. 이 단계에서는 소스 데이터베이스에서 전송 중인 데이터에 완전히 액세스할 수 있습니다.

2단계 - 롤포워드 단계

롤 포워드 단계는 다음 단계로 구성됩니다.

1. 증분 백업은 소스 데이터베이스에서 소스 단계로 가져옵니다.
2. 증분 백업 복사본은를 통해 대상 단계로 전송됩니다 Direct Connect.
3. 증분 백업 복사본은 리틀 엔디안 형식으로 대상 데이터베이스로 변환됩니다. 그런 다음 사본이 롤 포워드 단계라고 하는 초기 대상 데이터베이스에 적용됩니다.

이 단계를 여러 번 실행할 수 있습니다. 각 연속 증분 백업은 시간이 덜 걸리며 대상 데이터 파일 복사본이 소스 데이터베이스와 더 최신 상태가 됩니다. 1단계와 마찬가지로이 단계에서는 전송 중인 소스 데이터에 완전히 액세스할 수 있습니다.

3단계 - 전송 단계

세 번째 단계에는 다음 단계가 포함됩니다.

1. 전송 중인 테이블스페이스는 읽기 전용으로 변경됩니다.
2. 최종 증분 백업은 소스 데이터베이스에서 가져옵니다.
3. 메타데이터를 내보냅니다.
4. 백업이 전송되어 대상에 적용됩니다.
5. 객체 메타데이터를 가져옵니다.

이 시점에서 대상 데이터베이스의 시스템 변경 번호(SCN)는 소스 데이터베이스의 시스템 변경 번호와 일치합니다.

전송 가능한 테이블스페이스의 메타데이터는 소스 데이터베이스에서 내보내고 대상 데이터베이스로 가져옵니다. 메타데이터에는 사용자, 역할, 패키지, 프로시저, 함수, 테이블 및 인덱스에 대한 정보가 포함됩니다.

마지막으로 애플리케이션에서 대상 데이터베이스에 대한 전체 액세스를 위해 테이블스페이스를 읽고 쓸 수 있습니다.

이 단계 뒤에는 검증 단계가 이어집니다.

마이그레이션 단계

이 가이드에서는 Oracle 단일 인스턴스와 Oracle RAC를 소스 및 대상 데이터베이스로 마이그레이션하는 방법을 다룹니다. 소스와 대상이 Oracle RAC인 경우 각 마이그레이션 단계의 병렬 처리를 극대화할 수 있습니다.

0단계 - 초기 설정 단계

1. 대상 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 Oracle 데이터베이스 12.1.0.2 이상을 설치합니다.
2. 대상 데이터베이스를 확인합니다.

   대상 데이터베이스 인스턴스의 시간대와 문자 집합이 소스 데이터베이스와 동일한지 확인해야 합니다. 그렇지 않으면이 마이그레이션 접근 방식이 실패합니다.

   소스와 대상 모두의 시간대 버전이 동일한지 확인하려면 다음 명령을 실행합니다.

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

   소스 및 대상에서 DB 문자 집합과 국가 문자 집합이 모두 동일한지 확인하려면 다음 명령을 실행합니다.

   ```
   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 유틸리티를 설정합니다.

   소스 시스템에서 Oracle 문서 2471245.1에서 스크립트 파일 rman_xttconvert_VER4.zip을 다운로드하고 추출합니다. 특정 구성으로 `xtt.properties` 파일의 파라미터를 수정합니다. 그런 다음 `xtt.properties` 및 `xttdriver.pl` 스크립트를 대상 시스템에 복사합니다.

1단계 - 준비 단계(소스 데이터베이스는 온라인 상태로 유지)

이 단계에서는 전송할 테이블스페이스의 데이터 파일이 소스 시스템에 백업됩니다. 백업 복사본은 대상 시스템으로 전송되고 xttdriver.pl 스크립트를 실행하여 복원됩니다.

1단계: 소스 시스템에서 전체(레벨=0) 테이블스페이스 백업 수행

백업 기간을 줄이려면 xtt.properties 및 xttdriver.pl 파일을 여러 디렉터리에 복사합니다. 각 디렉터리 아래의 각 xtt.properties 파일에서 tablespaces 파라미터에 테이블스페이스 이름을 입력합니다. 그런 다음 각 디렉터리에서 --backup 옵션을 xttdriver.pl 사용하여를 실행합니다. 이 명령은 대상 데이터베이스를 설치하는 동안 TEMP생성된 SYSTEM, UNDO, 및 SYSAUX를 제외한 모든 테이블스페이스를 전송합니다.

예를 들어 각 xtt01~xtt04 디렉터리에는 xtt.properties 파일의 tablespaces= 파라미터 값이 서로 다른 모든 xtt 스크립트(xtt.properties 및 xttdriver.pl)가 있습니다. 각 xtt.properties 파일에서 tablespaces 파라미터를 편집할 때는 각 테이블스페이스 그룹의 데이터 파일의 총 크기가 비슷하도록 테이블스페이스 그룹을 나누는 것이 좋습니다.

이 가이드의 예제에서는 네 개의 테이블스페이스 그룹이 있다고 가정합니다. 소스 데이터베이스가 Oracle 단일 인스턴스인 경우 모든 xtt01~04 디렉터리를 생성합니다. 소스 데이터베이스가 Oracle RAC인 경우 각 Oracle RAC 서버에서 각 xtt 디렉터리를 생성할 수 있습니다.

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

다음 표에는 xtt.properties 파일의 tablespaces= 파라미터 예제가 나와 있습니다.

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_MEDIA, APPS_TS_NOLOGGING, APPS_TS_QUEUES, APPS_TS_SEED…

xttdriver.pl가 병렬로 실행되는 동안 기존 데이터 파일 백업 복사본이 삭제되지 않도록 하려면에서 다음 줄을 주석 처리합니다xttdriver.pl.

```
   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.pl를 사용하여 소스 시스템의 RMAN 백업을 만듭니다.

소스 시스템이 Oracle RAC인 경우 각 Oracle RAC 인스턴스에서 동시에 실행할 수 있습니다.

xttdriver.pl의 출력은 TMPDIR 환경 변수에 저장됩니다. --backup 옵션을 사용하여 각 xttdriver.pl 명령을 실행하고 --debug 옵션을 사용하여 디버그 모드를 켭니다.

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

이 예제에서 <nn>은 테이블스페이스 그룹을 나타냅니다. 테이블스페이스 그룹이 4개인 경우 <nn>은 01, 02, 03,가 됩니다04.

2단계: 전체 백업 복사본을 대상 시스템으로 전송

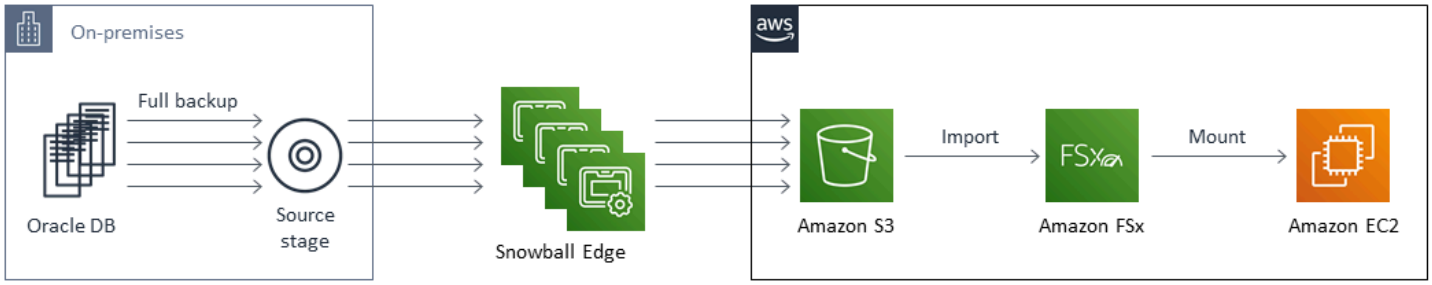
다음 두 옵션 중 하나를 사용하여 백업 복사본을 대상 시스템으로 전송합니다.

옵션 1 - 사용 AWS Snowball Edge

경로: 소스 데이터베이스 -> 소스 단계 AWS Snowball Edge -> Amazon S3 -> FSx for Lustre

> **Note**
>
> AWS Snowball Edge 는 더 이상 신규 고객이 사용할 수 없습니다. 신규 고객은 온라인 전송, 안전한 물리적 전송AWS DataSync을 위한 AWS 데이터 전송 터미널 또는 AWS Partner 솔루션을 탐색해야 합니다. 엣지 컴퓨팅의 경우를 살펴봅니다AWS Outposts.

다음 다이어그램은 Snowball Edge를 사용한 전체 백업 전송을 보여줍니다.



Snowball Edge는 대규모 데이터를 이동하는 데 사용할 수 있는 견고한 물리적 스토리지 및 컴퓨팅 디바이스입니다 AWS. Snowball Edge는 긴 전송 시간, 사용 가능한 대역폭 부족, 보안 문제 등 대규모 데이터 전송 시 발생할 수 있는 문제를 극복하는 데 도움이 됩니다.

전체 데이터 파일 백업 사본은 Snowball Edge에서 Amazon S3로 전송됩니다. 소스 시스템 크기가 100TB를 초과하는 경우 여러 Snowball Edge 디바이스를 사용하여 전송 기간을 줄여야 합니다.

Amazon FSx for Lustre는 Amazon S3와 긴밀하게 통합되어 있습니다. 몇 분 안에 S3 버킷에 연결된 FSx for Lustre 파일 시스템을 생성할 수 있습니다. Amazon S3 데이터 리포지토리에 연결되면 FSx for Lustre 파일 시스템은 Amazon S3 객체를 투명하게 파일로 표시합니다. 실제 환경에서 FSx for Lustre의 성능은 스토리지 용량, 각 파일의 크기, 파일이 파일 시스템에 얼마나 잘 배포되는지에 따라 달라집니다. FSx for Lustre를 대상 단계 스토리지로 사용하는 경우 Amazon S3에서 Amazon Elastic Block Store(Amazon EBS) 또는 Amazon Elastic File System으로 테이블스페이스 백업 복사본을 복원할 필요가 없습니다.

1. S3 버킷을 FSx for Lustre로 가져옵니다.

   Snowball Edge를 사용하여 소스 스테이지 영역(예: `src_backups`)을 S3 버킷(예:)으로 전송하고 저장합니다`s3-src-backups`.

   FSx for Lustre 파일 시스템(예: `dest_backups`)을 데이터 파일 백업 복사본의 대상 단계 영역으로 생성합니다.

대상 시스템(Amazon EC2)에 오픈 소스 Lustre 클라이언트를 설치합니다. 자세한 내용을 알아보려면 Amazon FSx for Lustre 사용 설명서를 참조하세요.

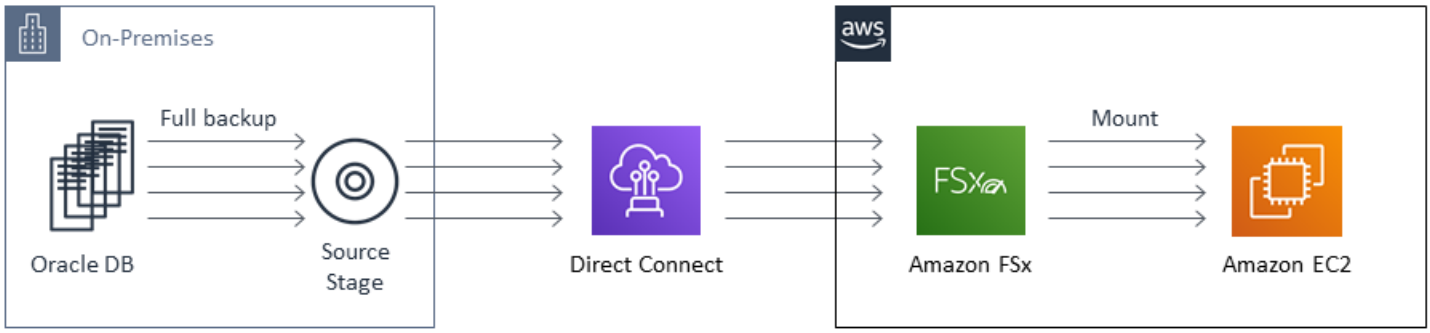
Lustre 클라이언트를 설치한 후 FSx for Lustre 파일 시스템을 대상 시스템(Amazon EC2)에 탑재합니다.

2. 소스의 `$TMPDIR res.txt`에서 대상의 `$TMPDIR` 로 전송합니다.

옵션 2 - 사용 AWS Direct Connect

경로: 소스 데이터베이스 -> 소스 단계 AWS Direct Connect -> FSx for Lustre

다음 다이어그램은를 사용하여 전체 백업을 전송하는 방법을 보여줍니다 AWS Direct Connect.



Direct Connect 를 사용하면 데이터 센터, 사무실 또는 콜로케이션 환경과 간에 프라이빗 네트워크 연결을 생성할 수 있습니다 AWS. 이러한 프라이빗 네트워크 연결은 네트워크 비용을 줄이고 처리량을 늘리며 일관된 경험을 제공합니다.

소스 시스템에서 Amazon FSx for Lustre 파일 시스템이 탑재된 대상 시스템(Amazon EC2)으로 데이터 파일 백업 사본을 직접 전송할 수 있습니다. 이 옵션은 높은 대역폭을 수용할 수 있는 경우 Snowball Edge 옵션보다 빠릅니다 Direct Connect.

Snowball Edge 또는에서 데이터 파일 백업 복사본을 전송 Direct Connect한 후 대상 단계 영역의 FSx for Lustre 파일 시스템에서 볼 수 있습니다.

3단계: 전체 백업 복사본 복원 및 데이터 파일 변환

전체 백업 복사본을 복원하고 데이터 파일을 대상 시스템 엔디안 형식으로 변환합니다. 복원 및 변환 기간을 줄이려면 각 테이블스페이스 그룹에 대한 `--restore` 옵션을 사용하여 `xttdriver.pl` 명령을 실행합니다.

```
cd /u01/oracle/expimp/xtt<nn>
```

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

단계가 완료되면 데이터 파일이 대상 시스템의 `xtt.properties` 파일에 정의된 dest_datafile_location에 배치됩니다.

2단계 - 롤포워드 단계(소스 데이터베이스는 온라인 상태로 유지)

롤 포워드 단계를 필요한 만큼 반복하여 대상 데이터 파일을 소스 데이터베이스까지 포착할 수 있습니다.

롤포워드 단계는 다음 단계로 구성됩니다.

1. 소스 데이터베이스에서 증분 백업을 생성합니다.
2. 백업을 대상 시스템으로 전송합니다.
3. 백업을 대상 시스템 엔디안 형식으로 변환합니다.
4. 변환된 대상 데이터 파일 사본에 백업을 적용하여 롤포워드합니다.

각 연속 증분 백업은 시간이 덜 걸리며 대상 데이터 파일 복사본이 소스 데이터베이스와 더 최신 상태가 됩니다.

1단계: 증분 백업을 병렬로 가져오기

소스 시스템에서 병렬로 전송 중인 테이블스페이스 그룹의 증분 백업을 수행합니다.

이 단계에서는 `xtt.properties` 파일에 나열된 모든 테이블스페이스=에 대한 증분 백업을 생성합니다.

소스 시스템에서 블록 변경 추적 기능을 활성화하면 증분 백업 시간을 크게 줄일 수 있습니다.

다음 명령은 전체 백업 후 다음 SCN을 자동으로 인식합니다.

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

2단계: 증분 백업 및 res.txt 파일을 대상 시스템으로 전송

대역폭이 충분한 경우를 사용하여 전송 기간을 줄일 수 있습니다 Direct Connect.

src_scratch_location 및 간에 증분 백업을 전송합니다dest_scratch_location. $TMPDIR 소스 시스템의와 $TMPDIR 대상 시스템의 간에 res.txt 파일을 전송합니다.

여러 증분 백업을 수행하는 경우 대상 시스템에 적용하기 전에 마지막 증분 백업 이후에 res.txt 파일을 복사해야 합니다.

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

3단계: 증분 백업 변환 및 적용

증분 백업을 대상 시스템 엔디안 형식으로 변환하고 대상 시스템의 대상 데이터 파일 사본에 증분 백업을 적용합니다.

이 안내서에서 테이블스페이스 그룹이 4개라고 가정합니다. 각 테이블스페이스 그룹에 대한 --restore 옵션을 사용하여 각 xttdriver.pl 명령을 실행합니다.

이 단계에서 Oracle XTTS 유틸리티는 대상 데이터베이스를 다시 시작합니다. 명령을 병렬로 실행하려면 Perl 스크립트에서 다음 사용자 지정을 사용해야 합니다xttdriver.pl.

1. 다음 줄에 주석을 추가합니다.
   - 4867행: my $outputstart = `sqlplus -L -s \"/ as sysdba\" \@xttstartupnomount.sql`;
   - 4868행: checkError ("Error in executing xttstartupnomount.sql", $outputstart);
   - 4992행: my $outputstart = `sqlplus -L -s \"/ as sysdba\" \@xttdbopen.sql`;\
2. 대상 DB를 NOMOUNT 상태로 만듭니다.

   ```
   sqlplus / as sysdba @xttstartupnomount.sql
   ```
3. 증분 백업의 롤포워드를 병렬로 수행합니다.

   ```
   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. 모든 증분 백업을 롤포워드한 후 대상 DB를 OPEN 상태로 설정합니다.

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

이 시점에서 소스 및 대상 데이터베이스의 SCNs 충분히 근접할 때까지 2단계를 반복할 수 있습니다. 주어진 목표 가동 중지 시간에 따라 3단계로 진행할지 아니면 2단계를 반복할지 결정해야 합니다.

3단계(전송 단계) 중 가동 중지 시간을 줄이기 위해 소스 데이터베이스를 로 설정하기 전에 , FUNCTION, 및 USERPACKAGEPROCEDURE와 같은 비세그먼트 기반 객체의 메타데이터를 내보내고 가져올 수 있습니다READ ONLY.

소스 데이터베이스의 데이터베이스 객체 수가 매우 많으면(수만 개) 메타데이터를 내보내고 가져오는 데 몇 시간이 걸립니다. 이 경우 메타데이터를 내보내는 것이 좋습니다.

세그먼트 기반이 아닌 객체 내보내기는 소스 시스템에서 읽기/쓰기로 실행됩니다. 그런 다음 소스 시스템을 로 설정하기 전에 대상 시스템으로 객체를 가져와야 합니다READ ONLY. 이때 , PACKAGE PROCEDURE및와 같은 데이터베이스 소스 객체를 FUNCTION 변경하지 않고 유지해야 합니다.

다음은 , , USER, PACKAGE_SPEC,를 포함하여 세그먼트 기반이 아닌 객체의 메타데이터를 내보내는 데 사용되는 PACKAGE_BODY덤프 파라미터 파일의 예입니다PROCEDUREFUNCTION.

```
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
```

다음 명령을 사용하여 메타데이터를 내보냅니다.

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

다음은 비세그먼트 객체의 메타데이터를 가져오는 덤프 파라미터 파일의 예입니다.

```
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,
…..
```

현재 대상 시스템에는 , SYSTEMUNDO, 및 SYSAUXTEMP를 제외한 테이블스페이스가 없습니다. 따라서 SYSTEM 테이블스페이스로 가져올 모든 객체를 다시 매핑해야 합니다.

다음 명령을 사용하여 메타데이터를 가져옵니다.

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

이제 대상 데이터베이스에서 생성된 객체를 볼 수 있습니다.

```
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, UNDO및를 제외한 테이블스페이스는 없습니다TEMP. USER 객체는를 기본 테이블스페이스USERS로 사용하여 생성됩니다. 3단계에서는 전송 가능한 테이블스페이스가 생성된 후 USER 객체가 원래 기본 테이블스페이스로 변경됩니다.

3단계 - 전송 단계(소스 데이터베이스는 읽기 전용)

이 단계에서는 소스 시스템이 읽기 전용이 됩니다. 대상 시스템의 데이터 파일은 최종 증분 백업을 적용하여 소스 시스템과 일치시킵니다. 그런 다음 소스 시스템에서 객체 메타데이터를 내보내고 대상 시스템으로 가져옵니다.

1단계: 소스 데이터베이스의 테이블스페이스를 읽기 전용으로 설정

SYSDBA로 소스 시스템에서 전송 중인 모든 테이블스페이스READ ONLY를 만듭니다.

가동 중지 시간을 줄이기 위해 다음 두 단계를 동시에 실행할 수 있습니다.

2단계: 최종 증분 백업 생성

소스 시스템에서 전송 중인 테이블스페이스의 최종 증분 백업을 생성합니다.

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

이 단계에서는 "ORA-20001: TABLESPACE(S) IS READONLY"와 같은 오류를 반환합니다. 오류가 예상되므로 무시해도 됩니다.

3단계: 메타데이터 내보내기

소스 데이터베이스에서 전송 가능한 테이블스페이스의 메타데이터를 내보냅니다.

다음은 전송 가능한 테이블스페이스의 메타데이터를 내보내기 위한 파라미터 파일의 예입니다.

```
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
```

또한 소스 시스템에 테이블과 인덱스가 많은 경우 통계를 제외하여 내보내는 동안 시간을 절약할 수 있습니다. 전송 가능한 테이블스페이스를 가져온 후 대상 시스템으로 통계를 가져옵니다.

를 실행하기 전에 소스 시스템에 덤프 파일이 저장되는 데이터베이스 디렉터리를 expdp생성합니다.

```
SQL> create directory dmpdir as <location>;
```

```
expdp system/<system password> parfile=<parameter file>
```

다음 두 단계는 RMAN 증분 백업을 사용하는 플랫폼 간 전송 가능한 테이블스페이스의 마지막 단계입니다. 이러한 단계는 순차적으로 수행해야 합니다.

4단계: 파일 전송 및 최종 증분 백업 적용

최종 증분 백업 및 내보내기 덤프 파일을 대상 시스템으로 전송하고 변환한 다음 최종 증분 백업을 적용합니다.

Direct Connect 를 사용하여 최종 증분 백업 복사본과 res.txt 파일을 대상으로 전송합니다. VPN 연결을 사용할 수 있지만 대역폭이 충분하면를 사용하면 가동 중지 시간이 크게 Direct Connect 줄어듭니다.

최종 증분 백업을 복원하려면 대상 시스템에서 각 테이블스페이스 그룹에 대해 `--restore` 옵션을 사용하여 다음 명령을 실행합니다.

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

5단계: 객체 메타데이터 가져오기

Oracle Data Pump를 사용하여 객체 메타데이터를 대상 시스템으로 가져옵니다. 다음 명령을 실행하여 대상 시스템의 `transport_datafiles=` 파라미터에 대한 데이터 파일 목록을 가져옵니다.

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

이전 명령을 실행할 때마다 `transport_datafiles=` 파라미터가 있는 `xttplugin.txt` 파일을 가져옵니다. 모든 `xttplugin.txt` 파일에서 한 줄`transport_datafiles=`로 병합하고 데이터 파일 목록을 가져오기 메타데이터에 대한 파라미터 파일의 `transport_datafiles` 인수에 넣습니다.

다음 코드 조각은 대상 시스템에서 전송 가능한 테이블스페이스를 가져오기 위한 파라미터 파일을 보여줍니다.

```
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'…..
```

를 실행하기 전에 내보내기 덤프 파일의 위치를 가리키는 데이터베이스 디렉터리를 impdp생성합니다.

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

4단계 - 전송된 데이터 검증

1단계: 테이블스페이스의 손상 여부 확인

이 단계에서 전송된 테이블스페이스는 대상 데이터베이스에서만 읽습니다. 다음과 같이 RMAN 명령을 실행하여 테이블스페이스가 유효한지 여부를 확인할 수 있습니다.

```
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
```

또한 테이블, 인덱스, 동의어, 보기 및 패키지 객체와 같은 객체 수를 확인할 수 있습니다.

2단계. 대상 데이터베이스에서 전송된 모든 테이블스페이스를 읽기/쓰기로 설정

```
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
```

결론

이 가이드에서는 AWS Snowball Edge AWS Direct Connect및 Amazon FSx for Lustre와 함께 Oracle 플랫폼 간 전송 가능한 테이블스페이스 및 RMAN 증분 백업을 사용하여 가동 중지 시간을 최소화하는 방법을 배웠습니다.

Oracle 데이터베이스를 온프레미스에서 로 마이그레이션할 때는 다음 변수를 AWS고려하세요.

- 에 필요한 네트워크 대역폭 Direct Connect
- FSx for Lustre의 용량
- Oracle 데이터베이스의 메타데이터 크기
- 증분 백업 크기

이 가이드에서 권장하는 방법을 사용하여 Unix 시스템에서 실행되는 100TB Oracle 데이터베이스에서 로 마이그레이션 AWS할 때 가동 중지 시간을 약 10시간으로 줄일 수 있습니다.

문서 이력

다음 표에서는이 가이드의 중요한 변경 사항을 설명합니다. 향후 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
–	소개 및 개요에서 이기종에 대한 언급을 변경했습니다.	2021년 7월 14일
–	최초 게시	2021년 3월 2일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.