Moving Data Across Platforms using External Tables

Tao Zuo
Tao_zuo@npd.com
NPD Inc.
9/2009

Contents

- Methods of moving data across platforms
- External Tables introduction
- How to move data using External Tables
- External Tables workload
- Case Study: Move 1TB data from a database on HP-UX (64-bit) to another database on Linux IA (64-bit).

Methods of moving data across platforms

- Database Link
- “Hand-rolled" data unloader with SQL*Plus spools, utl_file writes, etc. and SQL*Loader
- Convert transportable tablespaces with RMAN
- Export and import
- External tables with ORACLE_DATAPUMP access driver
- 3rd Party ETL tools

External Tables Introduction

- External tables access data in external sources as if it were in a table in the database
  - Introduced in 9*i*. Read-only in text-only datafiles.
  - Enhanced in 10*g* to be read-only or read-write, composed of flat files in a binary format.
  - Further enhanced in 11*g* to take advantage of COMPRESSION and ENCRYPTION options.

External Tables Introduction

- External tables are created using the SQL CREATE TABLE... ORGANIZATION EXTERNAL statement
- Attributes include:
  - TYPE: ORACLE_LOADER, ORACLE_DATAPUMP
  - DEFAULT DIRECTORY
  - ACCESS PARAMETERS
  - LOCATION

External Tables Introduction

- Use ORACLE_DATAPUMP Access Driver to unload and load data
  - The data unloaded into flat files are read only by ORACLE_DATAPUMP driver.
  - Neither DML nor index creation operation are allowed.
  - The data unloaded can be used for another external tables in the same or a different database.
  - The degree of parallelization is tied to the number of files to which you are unloading.
  - Dump files populated by different external tables can be aggregated into one and vice versa

External Tables Introduction

- ORACLE_DATAPUMP access drive automatically resolves:
  - Different character national character sets
  - Endianness
- For the supported datatypes:
  - Character, RAW, Number, Date, ROWID, etc.

External Tables Introduction

- Query External Tables
  - select
  - join
  - sort
  - views and synonyms
  - statistics

External Tables Introduction

- Alter External Tables
  - Default directory
  - Access Parameter
  - Location

External Tables Introduction

- Dropping External Tables
  - Use DROP TABLE statement
  - Only the table metadata is removed
  - No effect on the actual data

External Tables Introduction

- System privileges applied:
  - CREATE ANY TABLE
  - ALTER ANY TABLE
  - DROP ANY TABLE
  - SELECT ANY TABLE

External Tables Introduction

- Object privileges applied:
  - ALTER
  - SELECT
  - READ
  - WRITE

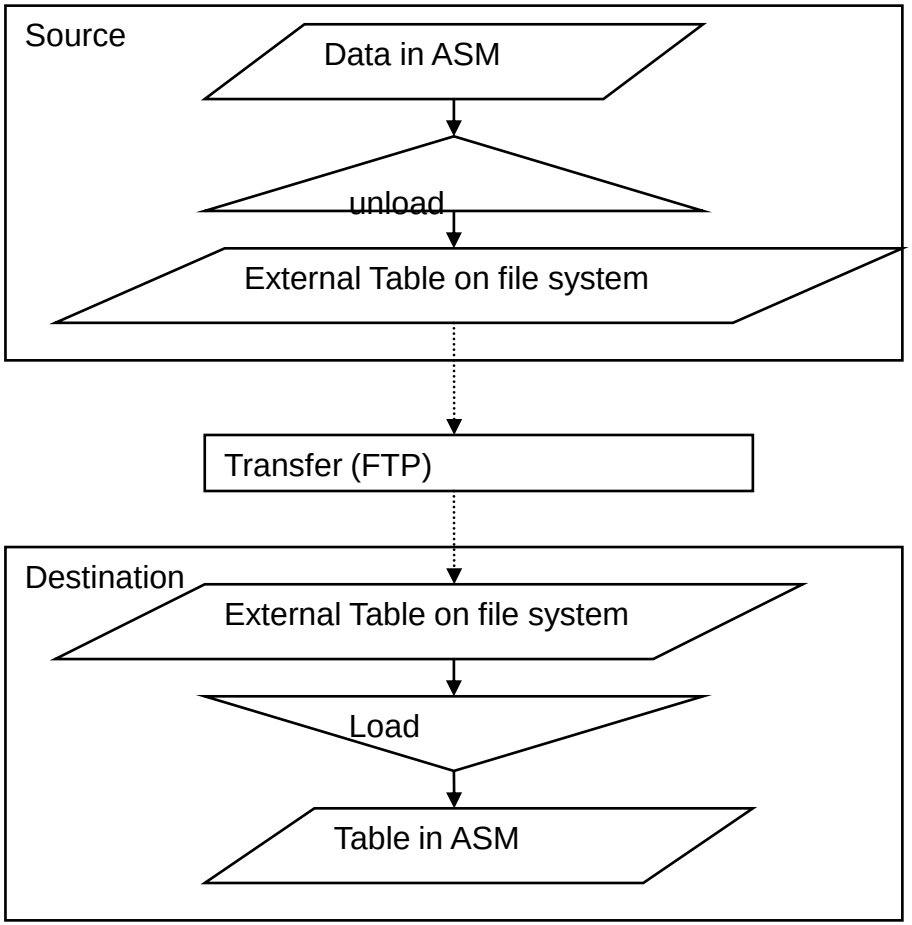
External Tables Introduction

- Views specific to external tables:
  - *_EXTERNAL_TABLES
  - *_EXTERNAL_LOCATIONS

How to move data using external tables

- In source database:
  - Create a DIRECTORY object in source database
  - Create “unloading” external table
- Verify dump files created
- Ship the dump file to target server
- In target database:
  - Create a DIRECTORY object the target database
  - Create “load” external table in the target database
  - CTAS target table with external table

How to move data using external tables



External Tables Workload

- **Top User Events:**
  - Direct path read 51.93%
  - Datapump dump file I/O 36.29%
  - CPU+Wait for CPU 10.86%

Case Study

Move 1TB data across platforms

	Source	**Destination**
Platform	HP-UX(64-bit)	Linux IA (64-bit)
OS Version	HP-UX 11.23	Linux 2.6.18-128.el5
Db Block Size	8K	16K
Data Compression	YES	YES
Partitioning	Range-List (64*746=47,744 segs)	Range-List (64*746=47,744 segs)

- The network bandwidth between the source and target server is limited to average of 40Mbit/sec.

Case Study

- Data Move across platform methods considered:
  - Transportable tablespaces
  - Direct load insert using dblink
  - Export/Import
  - Data Pump
  - UTL_FILE writes
  - External table with ORACLE_PUMP access driver

Case Study

- Benchmark test on “unloading” methods considered

Elasped (h:mi:ss)	Unload	Compress	FTP	Un-Compress	Insert/ copy	Total
TTS Datafile platform convert (timing is prorated from 400G to 45G)	0:55:00				0:09:00	* Failed on 3TB datafile convert
Database link					2:13:34	2:13:34
UTL_FILE	9:02:52					
Export (direct=y)	5:29:20					
External table w. ORALCE_PUMP access driver	0:12:33	0:13:56	0:08:16	0:01:00	1:28:12	2:03:57

Case Study

- Move data using External Tables with ORACLE_PUMP access driver

Case Study

- Break down 1TB source data into manageable chunks:
  - Use the first range partition key: key
  - One external table for one key of data, with parallel degree of 8.
  - Each external table contains 8 dump files.
  - Each dump file is about 20G with an average of 260million records.

Case Study

1. Create unload directory
2. Create load directory
3. Create load db link
4. For each source table, Unload statistics
    5. For each partition key value, Create an unload external table
        6. Compress 8 dump files
        7. ftp 8 dump files
        8. Un-compress 8 dump files
        9. Create 8 external tables for load
        10. Load 8 external tables
    11. Repeat step 5. to 10. until all partition key values are covered
12. Load statistics
13. Repeat step 4. to 12. until all source tables are covered.

Case Study

- Create a unload DIRECTORY object in Source:
  - As user sys:
    - Create directory unload as ‘/unload’;
    - Grant read, write on directory unload to <unload_user>;

Case Study

- Create a load DIRECTORY object in destination:
  - As user sys:
    - Create directory load as ‘/load’;
    - Grant read, write on directory load to <load_user>;

Case Study

- Unload table statistics
  - Create stats table on source
  - Export statistics to stats table on source

Case Study

- Create “unloading” external table

```
create table s_999xt
    ORGANIZATION EXTERNAL
    (
      TYPE ORACLE_DATAPUMP
      DEFAULT DIRECTORY unload
      ACCESS PARAMETERS (LOGFILE s_999xt)
      LOCATION
     ('s_999xt1.dmp','s_999xt2.dmp','s_999xt3.dmp','s_999xt4.dmp','s_99
     9xt5.dmp','s_999xt6.dmp','s_999xt7.dmp','s_999xt8.dmp')
    )
parallel 8
 as
select * from s
where key=999;
```

Case Study

- Verify extenal table created:

```
SQL> desc s_999xt
```

Case Study

- Verify the dump files created:

```
$ls -l s_999xt?.dmp
-rw-rw----  1 oracle    dba      28434382848 Mar 19 18:32 s_999xt1.dmp
-rw-rw----  1 oracle    dba      29189681152 Mar 19 18:32 s_999xt2.dmp
-rw-rw----  1 oracle    dba      28618231808 Mar 19 18:32 s_999xt3.dmp
-rw-rw----  1 oracle    dba      29596987392 Mar 19 18:32 s_999xt4.dmp
-rw-rw----  1 oracle    dba      28748021760 Mar 19 18:32 s_999xt5.dmp
-rw-rw----  1 oracle    dba      29367820288 Mar 19 18:32 s_999xt6.dmp
-rw-rw----  1 oracle    dba      28694409216 Mar 19 18:32 s_999xt7.dmp
-rw-rw----  1 oracle    dba      29159788544 Mar 19 18:32 s_999xt8.dmp
```

Case Study

- Transfer dump files with limited network bandwidth:
  - Compress dump files, one external table at a time, with parallel degree of 8
  - gzip is used as compression utility, compress ratio is close to 10:1
  - ftp the compressed dump files, one external table at a time.
  - Uncompress the dump files on destination server

Case Study

- Compress the dump files generated:

```
let i=1
while [ ${i} –le 8 ]; do
gzip s_999xt${i}.dmp &
let i=i+1
done
wait
```

Case Study

- Compressed dump files on Unix server:

```
-rw-rw----  1 oracle  dba  2812041660 Mar 19 18:32 s_999xt1.dmp.gz
-rw-rw----  1 oracle  dba  2942319139 Mar 19 18:32 s_999xt2.dmp.gz
-rw-rw----  1 oracle  dba  2731853408 Mar 19 18:32 s_999xt3.dmp.gz
-rw-rw----  1 oracle  dba  2873751337 Mar 19 18:32 s_999xt4.dmp.gz
-rw-rw----  1 oracle  dba  2813938907 Mar 19 18:32 s_999xt5.dmp.gz
-rw-rw----  1 oracle  dba  2878901853 Mar 19 18:32 s_999xt6.dmp.gz
-rw-rw----  1 oracle  dba  2844076017 Mar 19 18:32 s_999xt7.dmp.gz
-rw-rw----  1 oracle  dba  2835788086 Mar 19 18:32 s_999xt8.dmp.gz
```

Case Study

- ftp the compressed dump files from HP-UX server to Linux Server:

```
let i=1
While [ $i –le 8 ]; do
ftp -n -i ${d_svr} << EOF
        user oracle ${d_pwd}
        bin
        cd ${load_dir}
        put s_999xt${i}.dmp.gz
        bye
EOF
…
let i=i+1
done
```

Case Study

- Un-compress the dump files on Linux server:

```
let i=1
while [ ${i} –le 8 ]; do
gunzip s_999xt${i}.dmp.gz &
p[${i}]=$!
let i=i+1
done
wait ${[p$*]}
```

Case Study

- Un-compressed dump files on Linux server:

```
-rw-r--r-- 1 oracle oinstall 28434382848 Mar 20 19:32 s_999xt1.dmp
-rw-r--r-- 1 oracle oinstall 29189681152 Mar 20 19:37 s_999xt2.dmp
-rw-r--r-- 1 oracle oinstall 28618231808 Mar 20 19:43 s_999xt3.dmp
-rw-r--r-- 1 oracle oinstall 29596987392 Mar 20 19:48 s_999xt4.dmp
-rw-r--r-- 1 oracle oinstall 28748021760 Mar 20 19:54 s_999xt5.dmp
-rw-r--r-- 1 oracle oinstall 29367820288 Mar 20 20:00 s_999xt6.dmp
-rw-r--r-- 1 oracle oinstall 28694409216 Mar 20 20:05 s_999xt7.dmp
-rw-r--r-- 1 oracle oinstall 29159788544 Mar 20 20:11 s_999xt8.dmp
```

Case Study

- Populate data into the destination:
  - Create external tables in a stage schema in destination database with ORACLE_PUMP access driver reading the dump files transferred from HP-UX to Linux platform.
    - One external table per dump file
  - Create target table in target schema in destination database with data pump.
  - Insert data into target table with direct load from the external tables one external table at a time, to maintain a manageable transaction size.
- Import CBO stats from source to destination for the partitions with data populated.

Case Study

```
CREATE TABLE D_999XT${n}
(
  …
)
ORGANIZATION EXTERNAL
  (  TYPE ORACLE_DATAPUMP
     DEFAULT DIRECTORY LOAD_DIR
     ACCESS PARAMETER (LOGFILE D_999XT${n})
     LOCATION ('s_999xt${n}.dmp')
  )
```

Case Study

- Create database link LOAD_DBLINK
- Create destination table:
  - impdp <owner>/<pwd> parfile=impdp_network.par
  - impdp_network.par
    content=METADATA_ONLY
    directory=LOAD_DIR
    tables=<owner>.<table>
    job_name=imp_network_<table>
    parallel=4
    network_link=LOAD_DBLINK
    exclude=STATISTICS
    table_exists_action=SKIP
    logfile=LOAD_DIR:impdp_network_<table>.log

Case Study

- **Load Table:**

```
declare
v_code NUMBER;
v_errm VARCHAR2(64);
v_subpart varchar2(30);
v_sql varchar2(2000);
v_inscnt number := 0;

begin
v_sql := 'insert /*+ append */ into <table>';
v_sql := v_sql || ' nologging ';
v_sql := v_sql || ' select * from d_999xt${n}';
--dbms_output.put_line(v_sql);
execute immediate v_sql;
v_inscnt := v_inscnt + sql%rowcount;
commit;
dbms_output.put_line('Total: Inserted: '||v_inscnt||' rows');
EXCEPTION
when others then
v_code := SQLCODE;
v_errm := SUBSTR(SQLERRM, 1 , 64);
DBMS_OUTPUT.PUT_LINE('Error code ' || v_code || ': ' || v_errm);
end;
/
```

Case Study

1. Create unload directory
2. Create load directory
3. Create load db link
4. For each source table, Unload statistics
    5. **For each partition key value, Create an unload external table**
        6. **Compress 8 dump files**
        7. **ftp 8 dump files**
        8. **Un-compress 8 dump files**
        9. **Create 8 external tables for load**
        10. **Load 8 external tables**
    11. **Repeat step 5. to 10. until all partition key values are covered**
12. Load statistics
13. Repeat step 4. to 12. until all source tables are covered.

Case Study

- Load statistics;

impdp <owner>/<pwd> parfile=impdp_network.par

impdp_network.par

```
content=ALL
directory=LOAD_DIR
tables=s${table}_stats
job_name=s${table}_stats _imp_network
parallel=4
network_link=load_dblink
exclude=STATISTICS
table_exists_action=REPLACE
logfile=load_dir:impdp_network_s${table}_stats.log
```

exec dbms_stats.import_schema_stats(‘<owner>',‘s${table}_stats');

Case Study

1. Create unload directory
2. Create load directory
3. Create load db link
4. For each source table, Unload statistics
    5. **For each partition key value, Create an unload external table**
        6. **Compress 8 dump files**
        7. **ftp 8 dump files**
        8. **Un-compress 8 dump files**
        9. **Create 8 external tables for load**
        10. **Load 8 external tables**
    11. **Repeat step 5. to 10. until all partition key values are covered**
12. Load statistics
13. Repeat step 4. to 12. until all source tables are covered.

Case Study

Timing:

Process	Elapsed (D-hh:mm:ss)
Create “unloading” external tables: 1TB	3:05:26
Compress dump files: 100GB	6:00:00
ftp: 51Mbit/sec	4:23:00
Un-compress dump files: 1TB	0:42:00
Create “loading” external tables and insert into target tables: 300G data, 400G index	37:53:00
Total	2D-04:03:27

Conclusion

Using external tables with ORACLE_DATAPUMP access driver to move data across platforms:

- Is efficient on unload
- Is flexible at the granularity level
- Saves stage storage space
- Reduces network costs

Thank You!

Tao Zuo

tao_zuo@npd.com

NPD Inc.

9/2009