Even Newer Features of Oracle Data Pump 23ai

- Migrate Like a Pro

DOUG-day 2024, October 2024

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

The materials in this presentation pertain to Oracle Health, Oracle, Oracle Cerner, and Cerner Enviza which are all wholly owned subsidiaries of Oracle Corporation. Nothing in this presentation should be taken as indicating that any decisions regarding the integration of any EMEA Cerner and/or Enviza entities have been made where an integration has not already occurred.

Oracle

DBAs

run the world

Daniel Overby Hansen

Distinguished Product Manager

Daniel.Overby.Hansen@oracle.com

Connect with us

dohdatabase

@dohdatabase

https://dohdatabase.com

Even **faster** exports and imports

Use the Data Pump Bundle Patch

- 196 functional and performance fixes
- Doc ID 2819284.1

Importing a complete application with data drops from almost 2.5 hours to 48 minutes – by just applying the Data Pump bundle patch

A global provider of financial services

Bundle Patch

The patch is non-binary online installable

- Apply while the database instance is running
- Don't use Data Pump or `DBMS_METADATA`

The patch is not RAC rolling installable

Ensure dictionary and
fixed objects statistics are accurate

```
-- Gather stats before export, before import and after import

begin
   dbms_stats.gather_schema_stats('SYS');
   dbms_stats.gather_schema_stats('SYSTEM');
   dbms_stats.gather_fixed_objects_stats;
end;
```

After gathering dictionary stats, our Data Pump export went from 46 to 8 minutes

A global provider of sports betting and gambling

Use parallel and multiple dump files

```
--Apply parallelism by simply specifying a degree
expdp ... parallel=8


--Use different parallel degree on import
impdp ... parallel=32
```

Parallel Degree

Oracle Cloud Infrastructure

Number of OCPUs

Number of ECPUs / 4

On-prem (x86-64)

2 x physical cores

On-prem (other)

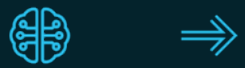
Depends

Parallel Architecture

```
expdp ... parallel=4
```

Parallel Architecture

```
expdp ... parallel=4
```



```
select * from t1
select * from t2
select * from t3
select * from t4
```

Control process

Worker processes

Parallel Architecture

```
expdp ... parallel=4
```

```
select /*+ parallel(2) */ * from t1
select * from t2
select * from t3
select * from t4
```

Worker 4 goes idle

Control process

Worker processes

```
--Use %L to allow multiple dump files

expdp ... parallel=8 dumpfile=exp%L.dmp
```

```
--Use %U in older releases of Oracle Database
--%L is introduced in Oracle Database 12.2
```

~~expdp ... parallel=8 dumpfile=exp%L.dmp~~

```
expdp ... parallel=8 dumpfile=exp%U.dmp
```

Parallel import does not need multiple dump files

```
--Split dump files into minor files for easier transport

expdp ... parallel=8 dumpfile=exp%L.dmp filesize=10G
```

21c

```
-- After export, store a checksum in the dump file.
-- Detects in-flight corruption or alteration.
-- Specify other algorithms using checksum_algorithm parameter.

expdp ... checksum=yes

impdp ... verify_checksum=yes
          verify_only=yes
```

For best protection against dump file tampering, use encrypted dump files

- Checksum is a weaker protection
- Requires Advanced Security Option

```
-- Protect your dump files from alteration by using encryption
-- Creating an encrypted dump file requires Advanced Security Option

expdp ... encryption=all encryption_algorithm=AES256
```

```
-- Protect your dump files from alteration by using encryption
-- Creating an encrypted dump file requires Advanced Security Option

expdp ... encryption=all encryption_algorithm=AES256
```

New default value

New In 23ai

Use parallel on transportable jobs

Benchmark Transportable Jobs

Oracle E-Business Suite database

600.000+ objects

Export parallel 1	2h 2m	Export parallel 16	1h 8m
Import parallel 1	6h 44m	Import parallel 16	1h 23m
Total	**8h 46m**	**Total**	**2h 31m**

Even **faster** constraint imports

Speed up imports by adding constraints in NOVALIDATE mode

A Constraint Can Be

VALIDATED

All data in the table obeys the constraint.
The database guarantees that data is good.

NOT VALIDATED

All data in the table may obey the constraint.
The database does not know if data is good.

Most constraints are VALIDATED

A Constraint Can Be

A constraint becomes validated by:

- Checking all existing data
- Checking all data during insert and update

On import, Data Pump creates constraints in the same state as in the source

```
--Example of which commands Data Pump import might execute as part of an import

create table sales ( .... );

insert into sales as select ... ;

alter table sales add constraint c_sales_1 check (c1 in (0,1)) enable validate;
alter table sales add constraint c_sales_2 check (c2 in ('A','B')) enable validate;
alter table sales add constraint c_sales_3 check (c3 > 0) enable validate;
```

Recursive full table scan

Recursive full table scan

Recursive full table scan

```
-- Add constraints with NOVALIDATE keyword regardless of state in source database
-- Significantly speeds up add constraints for larger tables

impdp ... transform=constraint_novalidate:y
```

```
--Example of which commands Data Pump import might execute as part of an import

create table sales ( .... );

insert into sales as select ... ;

alter table sales add constraint c_sales_1 check (c1 in (0,1)) enable novalidate;
alter table sales add constraint c_sales_2 check (c2 in ('A','B')) enable novalidate;
alter table sales add constraint c_sales_3 check (c3 > 0) enable novalidate;
```

Instant, no full table scan

```
--Example of which commands Data Pump import might execute as part of an import

create table sales ( .... );

insert into sales as select ... ;

alter table sales add constraint c_sales_1 check (c1 in (0,1)) enable novalidate;
alter table sales add constraint c_sales_2 check (c2 in ('A','B')) enable novalidate;
alter table sales add constraint c_sales_3 check (c3 > 0) enable novalidate;
```

Database validates new rows

Benchmark, 1 billion rows

Importing VALIDATE constraints

```
10-AUG-24 00:32:28.716: W-1 Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
10-AUG-24 00:36:42.762: W-1 . . imported "FUSION"."hwr_topic_tl" 151.2 GB 1044625000 rows in 254 seconds using external_table
10-AUG-24 00:45:41.226: W-1 Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
10-AUG-24 00:55:35.787: W-1      Completed 7 CONSTRAINT objects in 594 seconds
```

Importing NOVALIDATE constraints

```
10-AUG-24 00:14:56.050: W-1 Processing object type TABLE_EXPORT/TABLE/TABLE_DATA
10-AUG-24 00:19:10.311: W-1 . . imported "FUSION"."hwr_topic_tl" 151.2 GB 1044625000 rows in 254 seconds using external_table
10-AUG-24 00:29:20.841: W-1 Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
10-AUG-24 00:29:21.101: W-1      Completed 7 CONSTRAINT objects in 1 seconds
```

NOVALIDATE constraints prevent the optimizer from certain query rewrites

- Check `QUERY_REWRITE_INTEGRITY`

Validate constraints after import, or even after go-live

- Still requires a full scan of the table
- But can use parallel query
- And no table lock!

```
alter table sales add constraint c_sales_1 check (c1 in (0,1)) enable novalidate;

-----------------------
------- GO LIVE -------
-----------------------

#Validate constraints
#Optionally, use parallel query
alter session force parallel query;
alter table sales modify constraint c_sales_1 enable validate;
```

Exceptions

Data Pump always validates certain constraints:

1. On `DEFAULT ON NULL` columns
2. Used by a reference partitioned table
3. Used by a reference partitioned child table
4. Table with Primary key OID
5. Used as clustering key on a clustered table

Use with care if
you are transforming data on import

Also available in Oracle Database 19c via 19.23.0 Data Pump Bundle Patch

- Doc ID 2819284.1

Even **faster** LOB operations

You get the fastest LOB operations with SecureFile LOBs

2007

Oracle Database 11*g* Release 1

```
-- Do you still have any old BasicFile LOBs in your database?

select * from dba_lobs where securefile='NO';
```

If exporting SecureFile LOBs is slow, apply 19.23.0 Data Pump Bundle Patch

- Alternatively, trick Data Pump with fake stats

By applying the Data Pump Bundle Patch our 4.3 TB export with huge LOBs went from over 21 hours to 3 hours 22 minutes

A European government agency

… Plus, by increasing parallel from 4 to 12 the export dropped to 1 hour 51 minutes

A European government agency

… Finally, we moved to faster ASM based storage bringing it to 1 hour 7 minutes

A European government agency

Do you still have BasicFile LOBs?

- Use DIY parallelism during export
- Be sure to convert to SecureFile LOB on import

```
--Converting a BasicFile LOB to SecureFile during import,
--is faster than not converting it.
--Overview of Oracle LOBs (Doc ID: 1490228.1)

impdp ... transform=lob_storage:securefile
```

Importing as BasicFile LOBs

```
... imported "SCHEMA"."TABLE"   31.83 GB  681025 rows in 804 seconds using direct_path
```

Importing as SecureFile LOBs

```
... imported "SCHEMA"."TABLE"   31.83 GB  681025 rows in 261 seconds using external_table
```

Do you still have `LONG` and `LONG RAW`?

- Deprecated since Oracle8*i*

```
-- Convert LONG to CLOB, and LONG RAW to BLOB on import
-- Be sure to change your application as well,
-- PL/SQL interface for accessing LOBs and LONGs are not the same

impdp ... transform=long_to_lob:y
```

Even **faster** index imports

New In 23ai

Use table size to determine parallel degree on index creation

- Coming in future 23ai Data Pump Bundle Patch

How Data Pump Create Indexes

Really good for big indexes

Before 12.1

Worker 1 `CREATE INDEX ... PARALLEL 16`

Really good for small indexes

From 12.1

Worker 1 `CREATE INDEX ... PARALLEL 1`

Worker 2 `CREATE INDEX ... PARALLEL 1`

... `CREATE INDEX ... PARALLEL 1`

Worker 16 `CREATE INDEX ... PARALLEL 1`

How Data Pump Create Indexes

The best of both worlds

From 23

Worker 1	`CREATE INDEX ... PARALLEL 1`
Worker 2	`CREATE INDEX ... PARALLEL 8`
Worker 3	`CREATE INDEX ... PARALLEL 4`
Worker 4	`CREATE INDEX ... PARALLEL 3`

How Data Pump Create Indexes

1. Calculate the optimal parallel degree
2. Create indexes

How Data Pump Create Indexes

1 Calculate the optimal parallel degree

- Always parallel 1 when a table is less than 150 MB
- Customizable via `INDEX_THRESHOLD`
- Get optimal parallel degree using `EXPLAIN PLAN`

```
SQL> explain plan for create index i1 on t1(c1) parallel;

Explained.
```

```
SQL> explain plan for create index i1 on t1(c1) parallel;

Explained.

SQL> select * from table(dbms_xplan.display(format => 'ALL'));

...

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 4 because of degree limit
   - estimated index size: 655K bytes
```

How Data Pump Create Indexes

2 Create indexes

- One worker creates small indexes (parallel 1) in large batches
- The next worker starts with the biggest index (measured by optimal parallel degree)

How Data Pump Create Indexes

```
impdp ... parallel=16
```

Worker 1	CREATE INDEX ... PARALLEL 1	CREATE INDEX ... PARALLEL 1	CREATE INDEX ... PARALLEL 1	CREATE INDEX ... PARALLEL 1
Worker 2	CREATE INDEX ... PARALLEL 15	CREATE INDEX ... PARALLEL 8	CREATE INDEX ... PARALLEL 5	
Worker 3		CREATE INDEX ... PARALLEL 7		
Worker 4			CREATE INDEX ... PARALLEL 3	
...				
Worker 16				

New way of creating indexes on by default

- Controlled by parameter `ONESTEP_INDEX`

Benchmark, 1 billion rows

Importing with 19c settings constraints

```
10-AUG-24 00:55:35.830: Job "SYSTEM"."SYS_IMPORT_TABLE_01" successfully completed at Sat Aug 10 00:55:35 2024 elapsed 0 00:23:09
```

Importing NOVALIDATE constraints + new index method

```
10-AUG-24 01:48:38.844: Job "SYSTEM"."SYS_IMPORT_TABLE_01" successfully completed at Sat Aug 10 01:48:38 2024 elapsed 0 00:10:40
```

We expect much better result with more complex schemas

We'd love to see this feature in Oracle Database 19c

- Planned for future Data Pump Bundle Patch

Bits and pieces

```
-- Transforms all tablespace storage clauses to the user's default tablespace

impdp ... transform=tablespace:y
```

Time Zone File Version Check

Source
Version 43

Target
Version 42

```
create table t1 (
   ...
   c1 timestamp with timezone
   ...
)
```

```
Import: Release 19.0.0.0.0 - Production on Sun Sep 1 06:17:06 2024
Version 19.21.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 -
Production
ORA-39002: invalid operation
ORA-39405: Oracle Data Pump does not support importing from a source database with
TSTZ version 43 into a target database with TSTZ version 42.
```

Time zone conversion is a one-way street

... But my dump file doesn't contain any tables with TSTZ columns

```
-- Disable Data Pump time zone file version check during import.
-- Use with care and manually ensure no TSTZ is included in the dump file.
-- Currently available in Oracle Database 21c and 23ai, coming in 19c.

alter system set "_datapump_bypass_tstz_check"=true;
```

Improper use may logically corrupt your TSTZ data

- Use with caution

```
-- Always add diagnostic information to the Data Pump log files

expdp ... logtime=all metrics=yes

impdp ... logtime=all metrics=yes
```

No diagnostics

```
Processing object type DATABASE_EXPORT/FINAL_POST_INSTANCE_IMPCALLOUT/MARKER
Processing object type DATABASE_EXPORT/AUDIT_UNIFIED/AUDIT_POLICY_ENABLE
Processing object type DATABASE_EXPORT/POST_SYSTEM_IMPCALLOUT/MARKER
. . exported "SYS"."KU$_USER_MAPPING_VIEW"                5.890 KB      25 rows
. . exported "SYSTEM"."REDO_DB"                           25.59 KB       1 rows
```

Full diagnostics

```
02-NOV-21 19:43:56.061: W-1 Processing object type DATABASE_EXPORT/FINAL_POST_INSTANCE_IMPCALLOUT/MARKER
02-NOV-21 19:43:56.064: W-1      Completed 1 MARKER objects in 0 seconds
02-NOV-21 19:43:59.171: W-1 Processing object type DATABASE_EXPORT/AUDIT_UNIFIED/AUDIT_POLICY_ENABLE
02-NOV-21 19:43:59.195: W-1      Completed 2 AUDIT_POLICY_ENABLE objects in 0 seconds
02-NOV-21 19:43:59.380: W-1 Processing object type DATABASE_EXPORT/POST_SYSTEM_IMPCALLOUT/MARKER
02-NOV-21 19:43:59.387: W-1      Completed 1 MARKER objects in 0 seconds
02-NOV-21 19:43:59.830: W-1 . . exported "SYS"."KU$_USER_MAPPING_VIEW"    5.890 KB  25 rows in 0 seconds using external_table
02-NOV-21 19:43:59.923: W-1 . . exported "SYSTEM"."REDO_DB"               25.59 KB   1 rows in 0 seconds using direct_path
```

```
-- How do you deal with large Data Pump import log files?
-- In this example, the Data Pump import log file has almost 200.000 lines

$ du -h import.log
 29M   import.log

$ wc -l import.log
  189931 import.log
```

```
$ python3 dpla.py import.log

========================
 Data Pump Log Analyzer
========================

...

Operation Details
~~~~~~~~~~~~~~~~~
 Operation:                  Import
 Data Pump Version:          19.22.0.0.0
 DB Info:                    Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0
 Job Name:                   DPJOB1
 Status:                     COMPLETED
   Processing:               -
 Errors:                     1267
   ORA- Messages:            1267
 Start Time:                 2024-04-11 09:30:55
 End Time:                   2024-04-12 10:33:01
 Runtime:                    25:03:06

Data Processing
~~~~~~~~~~~~~~~
 Parallel Workers:           128
 Schemas:                    27
 Objects:                    224755
 Data Objects:               188084
 Overall Size:               13.16 TB
```

```
$ python3 dpla.py import.log -e

========================
 Data Pump Log Analyzer
========================

...

ORA- MESSAGES DETAILS
~~~~~~~~~~~~~~~~~~~~~
(sorted by count):

 Message                                                                                                                  Count
 ------------------------------------------------------------------------------------------------------------------------ -----
 ORA-39346: data loss in character set conversion for object COMMENT                                                       919
 ORA-39082: Object type PACKAGE BODY created with compilation warnings                                                      136
 ORA-39346: data loss in character set conversion for object PACKAGE_BODY                                                   54
 ORA-39082: Object type TRIGGER created with compilation warnings                                                            36
 ORA-39082: Object type PROCEDURE created with compilation warnings                                                          29
 ORA-31684: Object type USER already exists                                                                                  27
 ORA-39111: Dependent object type PASSWORD_HISTORY skipped, base object type USER already exists                             27
 ORA-39346: data loss in character set conversion for object PACKAGE                                                        18
 ORA-39082: Object type PACKAGE created with compilation warnings                                                            10
 ORA-39082: Object type VIEW created with compilation warnings                                                                7
 ORA-39346: data loss in character set conversion for object PROCEDURE                                                        2
 ORA-39082: Object type FUNCTION created with compilation warnings                                                             2
 ------------------------------------------------------------------------------------------------------------------------ -----
 Total                                                                                                                      1267
 ------------------------------------------------------------------------------------------------------------------------ -----
```

```
$ python3 dpla.py import.log -o

========================
 Data Pump Log Analyzer
========================

...

 Object                                  Count      Seconds      Workers     Duration
 ------------------------------- ---------- ----------- ----------- ------------
 SCHEMA_EXPORT/TABLE/TABLE_DATA     188296      6759219          128      6759219
 CONSTRAINT                            767        37253            1        37253
 TABLE                                2112         3225           51          156
 COMMENT                             26442          639          128           18
 PACKAGE_BODY                          197          125          128            5
 OBJECT_GRANT                         5279           25            1           25
 TYPE                                  270            6            1            6
 ALTER_PROCEDURE                       149            5            2            3
 ALTER_PACKAGE_SPEC                    208            4            3            2
 PACKAGE                               208            3            3            1
 PROCEDURE                             149            2            2            1

 ...

 --------------------------------- ---------- ----------- ----------- ------------
 Total                                224755      6800515          128      6796697
 --------------------------------- ---------- ----------- ----------- ------------
```

How about NOVALIDATE constraints?

Data Pump Log Analyzer

Table Details

Search for Table...

Table	Rows	Size	Seconds	Part	Subpart
SALES.ORDERS	118914251151	1.73 TB	878854	278	4448
SALES.INVOICES	115668171592	4.33 TB	805901	588	9408
SALES.TRANSACTIONS	115720037994	3.61 TB	611891	451	7216
FINANCE.EXPENSES	35091517646	258.14 GB	112962	367	0
MARKETING.CAMPAIGNS	11621627768	458.93 GB	82801	16	0
HR.EMPLOYEES	19433932893	296.19 GB	66156	2254	0
SALES.DOCUMENTS	4743542596	345.97 GB	48117	589	9424
SALES.REPORTS	4744610748	263.63 GB	42904	440	7040
INVENTORY.EQUIPMENT	9824954344	51.01 GB	33290	130	0
HR.PARTNERS	3983265247	83.62 GB	16388	3046	0

Data Pump Log Analyzer

- Free to use
- Download from GitHub
- Not an official Oracle tool
- Created by Marcus Doeringer
  Our migration superstar

Troubleshooting and Data Pump

In root and PDB

```
-- Change AWR snap interval to 15 minutes and create snapshot
exec dbms_workload_repository.modify_snapshot_settings(null, 15);
exec dbms_workload_repository.create_snapshot;

-- Optionally, enable SQL trace for Data Pump processes or specific SQL ID
alter system set events 'sql_trace {process: pname = dw | process: pname = dm} level=8';
alter system set events 'sql_trace[SQL: 03g1bnw08m4ds ]';

-- Run Data Pump job with trace (Doc ID 286496.1)
expdp ... metrics=yes logtime=all trace=1FF0300
impdp ... metrics=yes logtime =all trace=1FF0300

-- Create AWR snapshot and produce AWR report
exec dbms_workload_repository.modify_snapshot_settings(null, <original-value>);
exec dbms_workload_repository.create_snapshot;
@?/rdbms/admin/awrrpt
```

```
-- Change AWR snap interval to 15 minutes and create snapshot
exec dbms_workload_repository.modify_snapshot_settings(null, 15);
exec dbms_workload_repository.create_snapshot;

-- Optionally, enable SQL trace for Data Pump processes or specific SQL ID
alter system set events 'sql_trace {process: pname = dw | process: pname = dm} level=8';
alter system set events 'sql_trace[SQL: 03g1bnw08m4ds]';

-- Run Data Pump job with trace (Doc ID 286496.1)
expdp ... metrics=yes logtime=all trace=1FF0300
impdp ... metrics=yes logtime =all trace=1FF0300

-- Create AWR snapshot and produce AWR report
exec dbms_workload_repository.modify_snapshot_settings(null, <original-value>);
exec dbms_workload_repository.create_snapshot;
@?/rdbms/admin/awrrpt
```

```
-- Change AWR snap interval to 15 minutes and create snapshot
exec dbms_workload_repository.modify_snapshot_settings(null, 15);
exec dbms_workload_repository.create_snapshot;

-- Optionally, enable SQL trace for Data Pump processes or specific SQL ID
alter system set events 'sql_trace {process: pname = dw | process: pname = dm} level=8';
alter system set events 'sql_trace[SQL: 03g1bnw08m4ds ]';

-- Run Data Pump job with trace (Doc ID 286496.1)
expdp ... metrics=yes logtime=all trace=1FF0300
impdp ... metrics=yes logtime=all trace=1FF0300

-- Create AWR snapshot and produce AWR report
exec dbms_workload_repository.modify_snapshot_settings(null, <original-value>);
exec dbms_workload_repository.create_snapshot;
@?/rdbms/admin/awrrpt
```

```
-- Change AWR snap interval to 15 minutes and create snapshot
exec dbms_workload_repository.modify_snapshot_settings(null, 15);
exec dbms_workload_repository.create_snapshot;

-- Optionally, enable SQL trace for Data Pump processes or specific SQL ID
alter system set events 'sql_trace {process: pname = dw | process: pname = dm} level=8';
alter system set events 'sql_trace[SQL: 03g1bnw08m4ds ]';

-- Run Data Pump job with trace (Doc ID 286496.1)
expdp ... metrics=yes logtime=all trace=1FF0300
impdp ... metrics=yes logtime=all trace=1FF0300

-- Create AWR snapshot and produce AWR report
exec dbms_workload_repository.modify_snapshot_settings(null, <original-value>);
exec dbms_workload_repository.create_snapshot;
@?/rdbms/admin/awrrpt
```

In root and PDB

Troubleshooting

Collect:

- Data Pump log file
- AWR report - CDB and PDB level
- Data Pump trace files
  - Stored in the database trace directory
  - Control process file name: `*dm*`
  - Worker process file names: `*dw*`

Thank You