ORACLE
CloudWorld

Oracle Data Pump Deep Dive with Development

Alex Zaballa
Practice Director, Accenture

Roy Swonger
Vice President, Oracle

Mike Dietrich
Senior Director Product Management, Oracle

Roy Swonger

Vice President

Database Upgrade, Utilities & Patching

royfswonger

@royfswonger

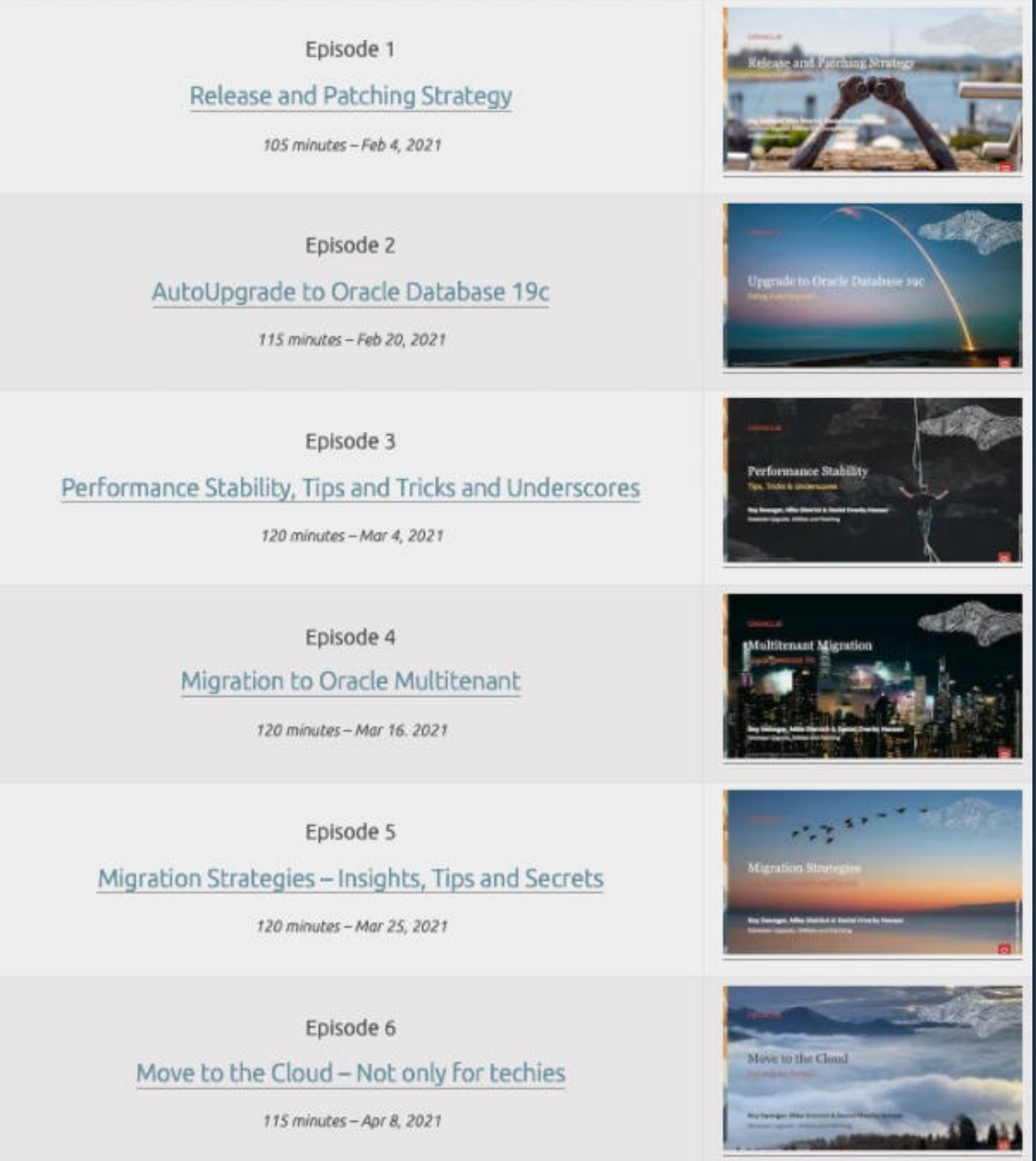
Mike Dietrich

Senior Director Product Management
Database Upgrade

MikeDietrichDE

@MikeDietrichDE

https://mikedietrichde.com



Episode 1

Release and Patching Strategy

105 minutes – Feb 4, 2021

Episode 2

AutoUpgrade to Oracle Database 19c

115 minutes – Feb 20, 2021

Episode 3

Performance Stability, Tips and Tricks and Underscores

120 minutes – Mar 4, 2021

Episode 4

Migration to Oracle Multitenant

120 minutes – Mar 16. 2021

Episode 5

Migration Strategies – Insights, Tips and Secrets

120 minutes – Mar 25, 2021

Episode 6

Move to the Cloud – Not only for techies

115 minutes – Apr 8, 2021

Recorded Web Seminars

https://MikeDietrichDE.com/videos

Data Pump

Architecture

- Overview
- Control table
- DBMS_DATAPUMP
- Parallel

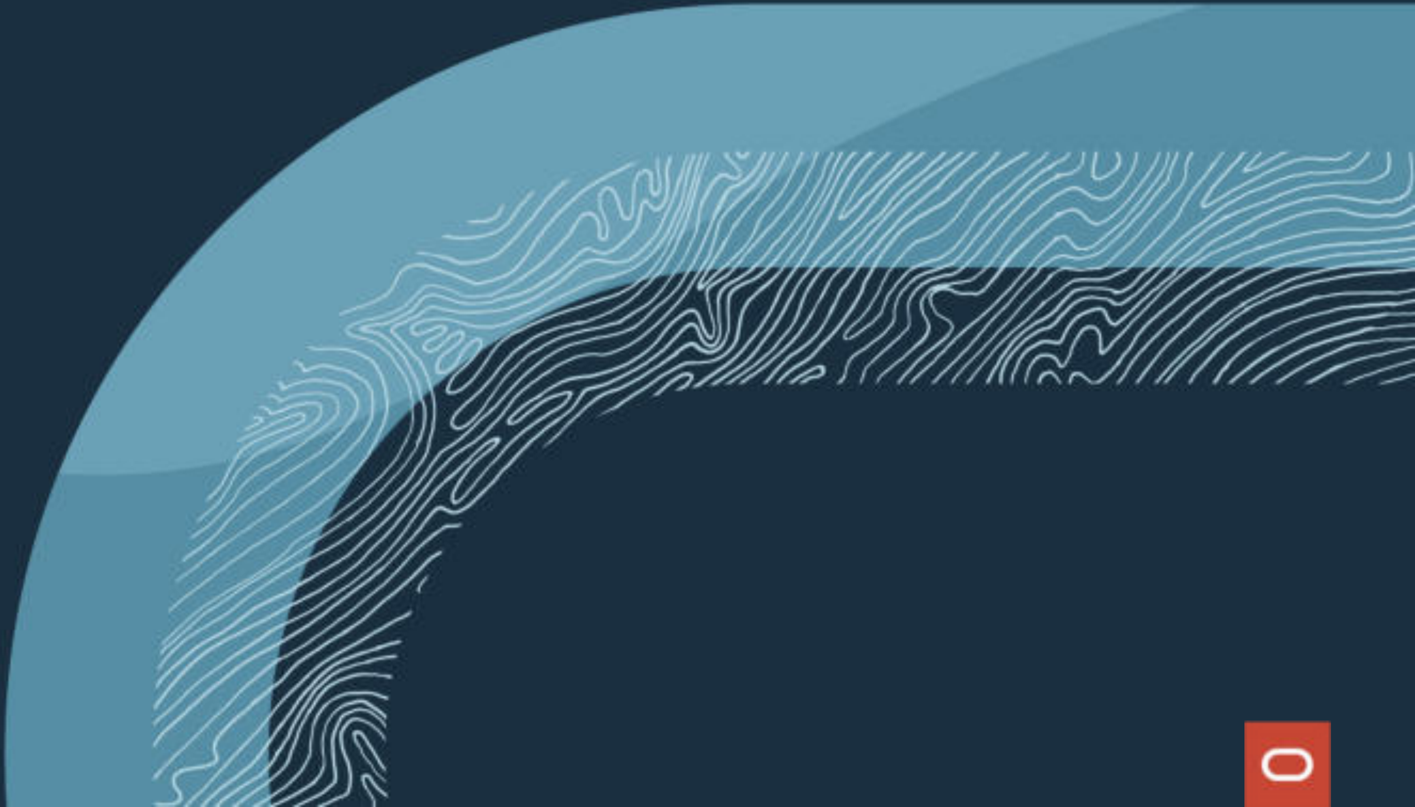
Bundle Patch

Troubleshooting

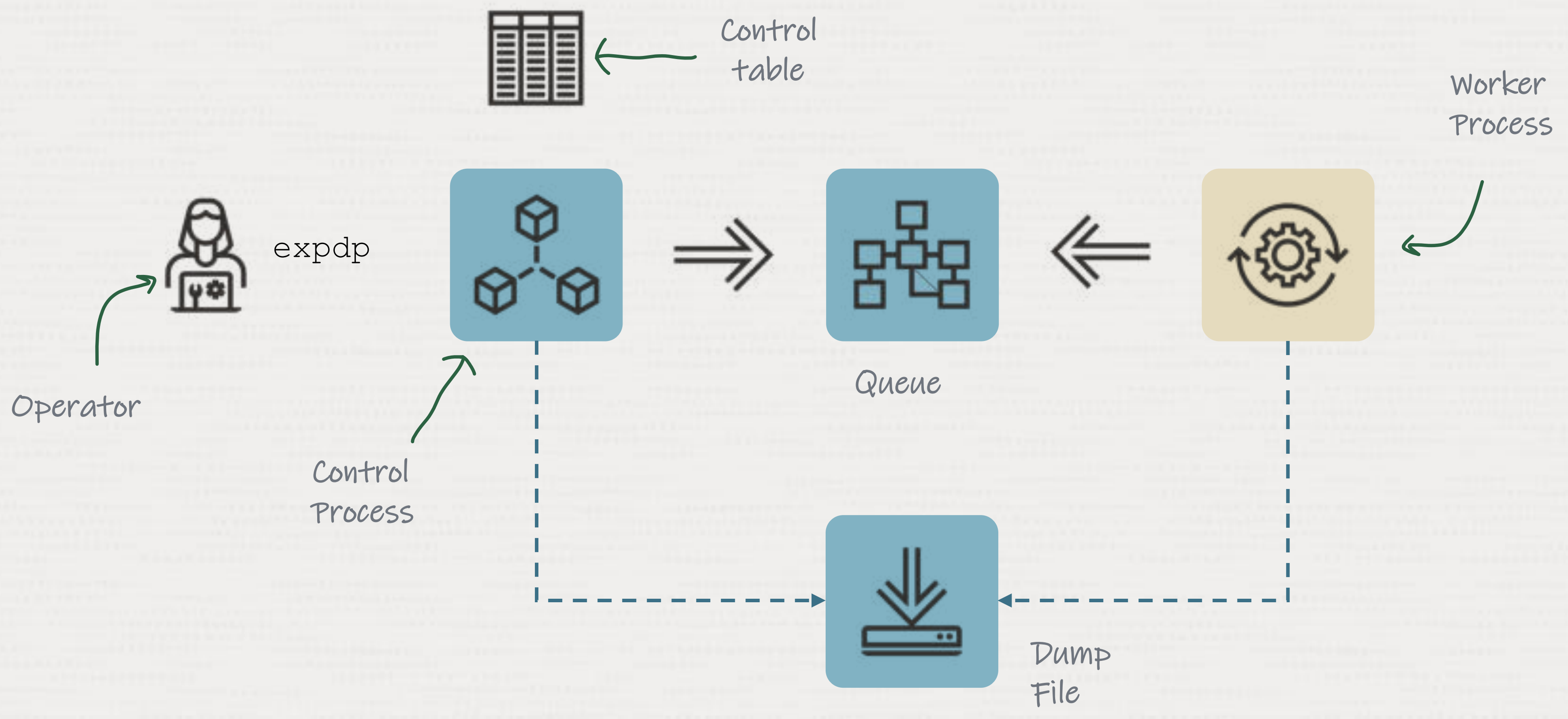
New Features

Architecture

Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another.



Architecture

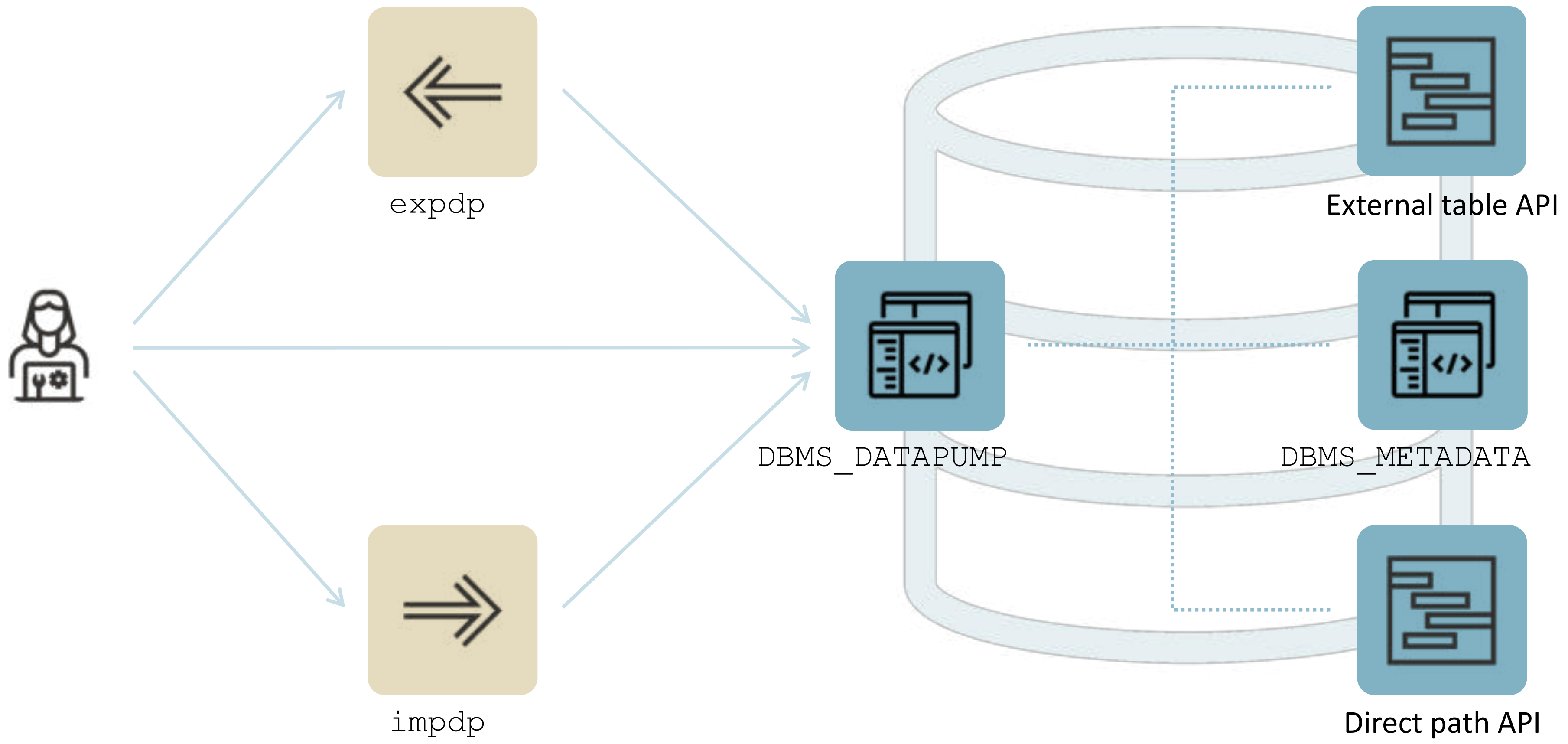


Control Table

```
SQL> select name, value_t from dpuser.sys_export_schema_01;

NAME                   VALUE_T
---------------------- --------------------------------------------
SYS_EXPORT_SCHEMA_01   DB19.LOCALDOMAIN
LOG_FILE_DIRECTORY     DATA_PUMP_DIR
LOG_FILE_NAME          export.log
CLIENT_COMMAND         dpuser/******** schemas=app keep_master=y
SCHEMA_LIST            'APP'
SCHEMA_EXPR            IN ('APP')
COMPRESSION            METADATA_ONLY
COMPRESSION_ALGORITHM  BASIC
DATA_ACCESS_METHOD     AUTOMATIC
.
.
.
```

API


expdp
impdp
DBMS_DATAPUMP
External table API
DBMS_METADATA
Direct path API

`DBMS_DATAPUMP` is
a supported and documented API

- Zero Downtime Migration
- Enterprise Manager
- SQL Developer
- SQLcl

DBMS_DATAPUMP

Client

```
expdp directory=mydir \
   logfile=exp.log \
   dumpfile=exp%u.dmp \
   schemas=app \
   parallel=4 \
   metrics=y \
   logtime=all
```

API

```
h1 := DBMS_DATAPUMP.OPEN(
   operation => 'EXPORT',
   job_mode => 'SCHEMA',
   remote_link => null,
   job_name => 'MY_JOB',
   version => null);

-- Create a Data Pump job to do a schema
-- export. Give it a meaningful name
```

Use 10046 trace to generate `DBMS_DATAPUMP` calls

Generate PL/SQL

1. Enable SQL trace on a test database

```
SQL> alter system
     set event='10046 trace name context forever, level 4';
```

2. Execute your Data Pump command

```
$ impdp system ... parfile=import.par
```

3. Examine the trace file

```
$ vi ORCL_ora_12345.trc
```

Pro tip: Grep for `DBMS_DATAPUMP` to find the right trace file

Use **PARALLEL** to speed up your Data Pump job

Parallel

```
PARALLEL=4
```

```
SELECT * FROM t1
```
1

```
SELECT /*+ parallel(2) */ * FROM t2
```
2,3

```
SELECT * FROM t3
```
4

idle

Why isn’t my job using all the PARALLEL that I gave it?

Why Might Data Pump Workers Be Idle?

Some possibilities…

1. Data Pump might be using Parallel Query
   - PX processes count against the total parallelism
2. BasicFile LOBs do not allow parallel DML
3. Export parallelism requires multiple dumpfiles
4. `NETWORK_LINK` jobs
   - Export and import metadata serially
   - Cannot use Parallel Query (one worker per partition/subpartition, but no PQ within a partition)

Get all the details
in our webinar on YouTube

Data Pump

Apply the Data Pump bundle patch

- Data Pump Recommended Proactive Patches For 19.10 and Above (Doc ID 2819284.1)

Data Pump bundle Patch

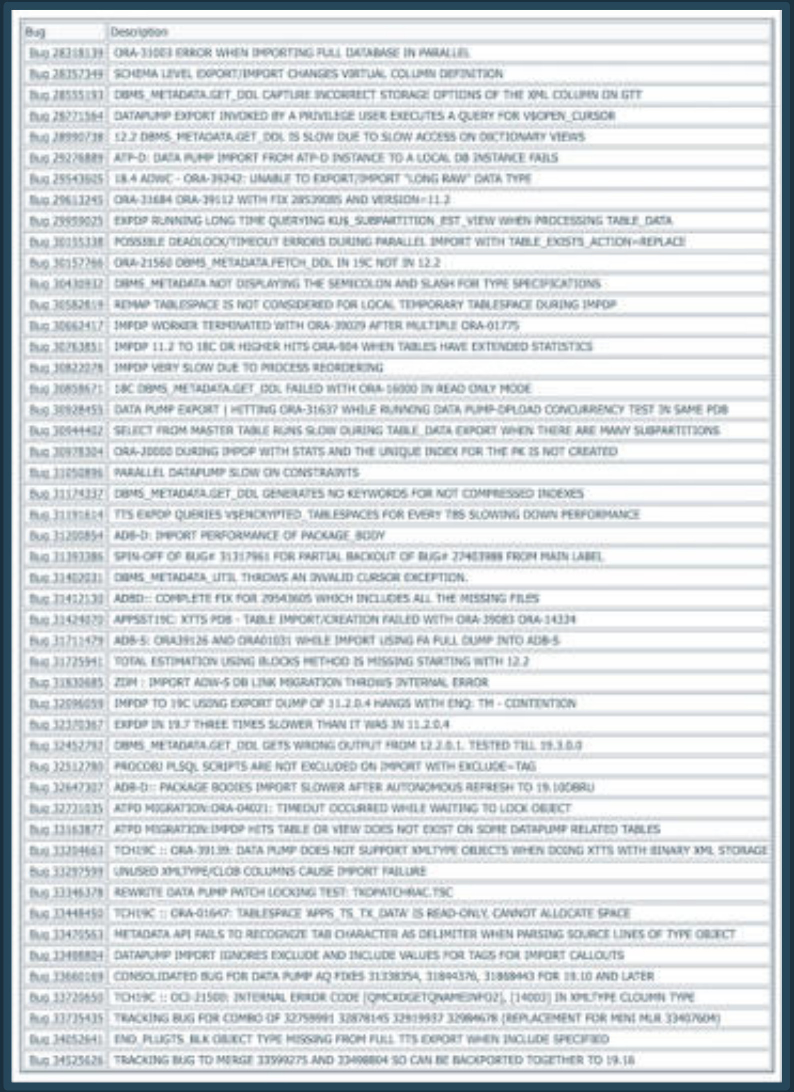
Fewer Bugs

Important patches are included.
Monitor for bugs that affects many customers.

Faster Patching

The bundle patch changes the way Data Pump is patched. Subsequent patches apply faster.

Data Pump Bundle Patch - MOS Note: 2819284.1



Bug	Description
Bug 28318139	ORA-31003 ERROR WHEN IMPORTING FULL DATABASE IN PARALLEL
Bug 28357349	SCHEMA LEVEL EXPORT/IMPORT CHANGES VIRTUAL COLUMN DEFINITION
Bug 28555193	DBMS_METADATA.GET_DDL CAPTURE INCORRECT STORAGE OPTIONS OF THE XML COLUMN ON GTT
Bug 28771564	DATAPUMP EXPORT INVOKED BY A PRIVILEGE USER EXECUTES A QUERY FOR V$OPEN_CURSOR
Bug 28990738	12.2 DBMS_METADATA.GET_DDL IS SLOW DUE TO SLOW ACCESS ON DICTIONARY VIEWS
Bug 29276889	ATP-D: DATA PUMP IMPORT FROM ATP-D INSTANCE TO A LOCAL DB INSTANCE FAILS
Bug 29543605	18.4 ADWC - ORA-39242: UNABLE TO EXPORT/IMPORT "LONG RAW" DATA TYPE
Bug 29613245	ORA-31684 ORA-39112 WITH FIX 28539085 AND VERSION=11.2
Bug 29959025	EXPDP RUNNING LONG TIME QUERYING KU$_SUBPARTITION_EST_VIEW WHEN PROCESSING TABLE_DATA
Bug 30135338	POSSIBLE DEADLOCK/TIMEOUT ERRORS DURING PARALLEL IMPORT WITH TABLE_EXISTS_ACTION=REPLACE
Bug 30157766	ORA-21560 DBMS_METADATA.FETCH_DDL IN 19C NOT IN 12.2
Bug 30430932	DBMS_METADATA NOT DISPLAYING THE SEMICOLON AND SLASH FOR TYPE SPECIFICATIONS
Bug 30582819	REMAP TABLESPACE IS NOT CONSIDERED FOR LOCAL TEMPORARY TABLESPACE DURING IMPDP
Bug 30662417	IMPDP WORKER TERMINATED WITH ORA-39029 AFTER MULTIPLE ORA-01775
Bug 30763851	IMPDP 11.2 TO 18C OR HIGHER HITS ORA-904 WHEN TABLES HAVE EXTENDED STATISTICS
Bug 30822078	IMPDP VERY SLOW DUE TO PROCESS REORDERING
Bug 30858671	18C DBMS_METADATA.GET_DDL FAILED WITH ORA-16000 IN READ ONLY MODE
Bug 30928455	DATA PUMP EXPORT | HITTING ORA-31637 WHILE RUNNING DATA PUMP-DPLOAD CONCURRENCY TEST IN SAME PDB
Bug 30944402	SELECT FROM MASTER TABLE RUNS SLOW DURING TABLE_DATA EXPORT WHEN THERE ARE MANY SUBPARTITIONS
Bug 30978304	ORA-20000 DURING IMPDP WITH STATS AND THE UNIQUE INDEX FOR THE PK IS NOT CREATED
Bug 31050896	PARALLEL DATAPUMP SLOW ON CONSTRAINTS
Bug 31174337	DBMS_METADATA.GET_DDL GENERATES NO KEYWORDS FOR NOT COMPRESSED INDEXES
Bug 31191614	TTS EXPDP QUERIES V$ENCRYPTED_TABLESPACES FOR EVERY TBS SLOWING DOWN PERFORMANCE
Bug 31200854	ADB-D: IMPORT PERFORMANCE OF PACKAGE_BODY
Bug 31393386	SPIN-OFF OF BUG# 31317961 FOR PARTIAL BACKOUT OF BUG# 27403988 FROM MAIN LABEL
Bug 31402031	DBMS_METADATA_UTIL THROWS AN INVALID CURSOR EXCEPTION.
Bug 31412130	ADBD:: COMPLETE FIX FOR 29543605 WHICH INCLUDES ALL THE MISSING FILES
Bug 31424070	APPSST19C: XTTS PDB - TABLE IMPORT/CREATION FAILED WITH ORA-39083 ORA-14334
Bug 31711479	ADB-S: ORA39126 AND ORA01031 WHILE IMPORT USING FA FULL DUMP INTO ADB-S
Bug 31725941	TOTAL ESTIMATION USING BLOCKS METHOD IS MISSING STARTING WITH 12.2
Bug 31830685	ZDM : IMPORT ADW-S DB LINK MIGRATION THROWS INTERNAL ERROR
Bug 32096059	IMPDP TO 19C USING EXPORT DUMP OF 11.2.0.4 HANGS WITH ENQ: TM - CONTENTION
Bug 32370367	EXPDP IN 19.7 THREE TIMES SLOWER THAN IT WAS IN 11.2.0.4
Bug 32452792	DBMS_METADATA.GET_DDL GETS WRONG OUTPUT FROM 12.2.0.1. TESTED TILL 19.3.0.0
Bug 32512780	PROCOBJ PLSQL SCRIPTS ARE NOT EXCLUDED ON IMPORT WITH EXCLUDE=TAG
Bug 32647307	ADB-D:: PACKAGE BODIES IMPORT SLOWER AFTER AUTONOMOUS REFRESH TO 19.10DBRU
Bug 32731035	ATPD MIGRATION:ORA-04021: TIMEOUT OCCURRED WHILE WAITING TO LOCK OBJECT
Bug 33163877	ATPD MIGRATION:IMPDP HITS TABLE OR VIEW DOES NOT EXIST ON SOME DATAPUMP RELATED TABLES
Bug 33204663	TCH19C :: ORA-39139: DATA PUMP DOES NOT SUPPORT XMLTYPE OBJECTS WHEN DOING XTTS WITH BINARY XML STORAGE
Bug 33297599	UNUSED XMLTYPE/CLOB COLUMNS CAUSE IMPORT FAILURE
Bug 33346378	REWRITE DATA PUMP PATCH LOCKING TEST: TKDPATCHRAC.TSC
Bug 33448450	TCH19C :: ORA-01647: TABLESPACE 'APPS_TS_TX_DATA' IS READ-ONLY, CANNOT ALLOCATE SPACE
Bug 33470563	METADATA API FAILS TO RECOGNIZE TAB CHARACTER AS DELIMITER WHEN PARSING SOURCE LINES OF TYPE OBJECT
Bug 33498804	DATAPUMP IMPORT IGNORES EXCLUDE AND INCLUDE VALUES FOR TAGS FOR IMPORT CALLOUTS
Bug 33660169	CONSOLIDATED BUG FOR DATA PUMP AQ FIXES 31338354, 31844376, 31868443 FOR 19.10 AND LATER
Bug 33720650	TCH19C :: OCI-21500: INTERNAL ERROR CODE [QMCXDGETQNAMEINFO2], [14003] IN XMLTYPE CLOUMN TYPE
Bug 33735435	TRACKING BUG FOR COMBO OF 32759991 32878145 32919937 32984678 (REPLACEMENT FOR MINI MLR 33407604)
Bug 34052641	END_PLUGTS_BLK OBJECT TYPE MISSING FROM FULL TTS EXPORT WHEN INCLUDE SPECIFIED
Bug 34525626	TRACKING BUG TO MERGE 33599275 AND 33498804 SO CAN BE BACKPORTED TOGETHER TO 19.16

49

fixes

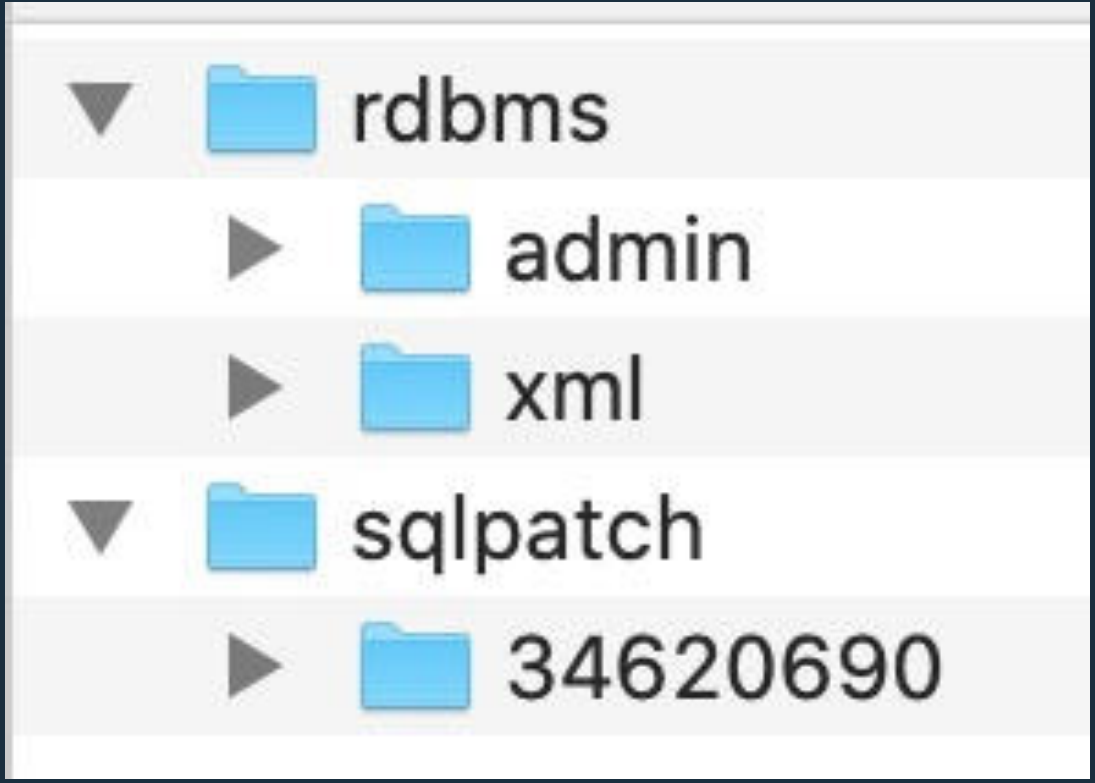
Data Pump Bundle Patch for 19.16.0

Why aren't those fixes included in an RU?

The Data Pump bundle patch is **not** RAC Rolling and Standby-first Installable

But … it's much easier than it looks like

Data Pump Bundle Patch Contents



Bundle Patch contains only:

- sql
- plsql
- xml

But it does not contain any files which require a compilation/make of `rdbms`

➔ It can be applied ONLINE

```
OPatch continues with these patches:   34620690

Do you want to proceed? [y|n]
y
User Responded with: Y
All checks passed.
Backing up files...
Applying interim patch '34620690' to OH
'/u01/app/oracle/product/19'

Patching component oracle.rdbms, 19.0.0.0.0...

Patching component oracle.rdbms.dbscripts, 19.0.0.0.0...
Patch 34620690 successfully applied.
```

When you run `datapatch`, ensure that there are **no active Data Pump jobs**

Non-Binary Online Patching Safeguards

Installing the Data Pump Bundle Patch when Data Pump is in use:

- Built-in 3-minute timeout before signaling an error

```
BEGIN ku$_dpload.initial_phase; END;
*
ERROR at line 1:
ORA-20000: Retry dpload.sql script later when
Data Pump and Metadata API are not in use; current users are:
pid:11720, user:SYS, machine:<Machine>, sid:263,
module:sqlplus@<ConnectString> (TNS V1-
ORA-06512: at "SYS.KU$_DPLOAD", line 1042
ORA-06512: at line 1
```

Non-Binary Online Patching Safeguards

- Attempting to run Data Pump while patching is in progress:

```
Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
ORA-31626: job does not exist
ORA-31637: cannot create job SYS_EXPORT_FULL_01 for user SYSTEM
ORA-06512: at "SYS.KUPV$FT", line 1142
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 95
ORA-06512: at "SYS.KUPV$FT", line 1751
ORA-39062: error creating master process DM00
ORA-39107: Master process DM00 violated startup protocol. Master error:
…
```

- **Note:** With the 19.14 (or later) Data Pump Bundle Patch installed you will see a much better error message:

```
ORA-39442: Data Pump software update in progress
```

Once applied, Data Pump Bundle Patch speeds up future patching significantly

Importing a complete application with data goes from almost 2,5 hours to 48 minutes – by just applying the Data Pump bundle patch

Global provider of financial services

Data Pump

Architecture

Troubleshooting

- Tracing
- Restartability
- Index creation

Bundle Patch

New Features

Troubleshooting

1. LOGS

Find and get the most out of the log files

2. VIEWS

Using views inside the database to monitor

3. TRACE

Enabling trace to debug a specific issue

Always use `METRICS=YES` and `LOGTIME=ALL`

Log Files

- 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
```

Log Files

- Check `alert.log` and upload it with an SR

```
2022-02-21T11:31:23.315021+01:00
db_recovery_file_dest_size of 18432 MB is 1.23% used. This is a
user-specified limit on the amount of space that will be used by this
database for recovery-related files, and does not reflect the amount of
space available in the underlying filesystem or ASM diskgroup.
2022-02-21T11:31:25.810983+01:00
DM00 started with pid=80, OS id=17226, job DPUSER.SYS_EXPORT_SCHEMA_01
2022-02-21T11:31:56.980017+01:00
Thread 1 advanced to log sequence 20 (LGWR switch),  current SCN: 6660216
  Current log# 2 seq# 20 mem# 0: /u02/oradata/DB19/redo02.log
2022-02-21T11:31:57.197532+01:00
ARC1 (PID:16810): Archived Log entry 3 added for T-1.S-19 ID 0x31223092 LAD:1
2022-02-21T11:32:01.650969+01:00
TABLE SYS.WRP$_REPORTS: ADDED INTERVAL PARTITION SYS_P865 (4435) VALUES LESS THAN (TO_DATE(' 2022-02-22 01:00:00',
'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
TABLE SYS.WRP$_REPORTS_DETAILS: ADDED INTERVAL PARTITION SYS_P866 (4435) VALUES LESS THAN (TO_DATE(' 2022-02-22
01:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
TABLE SYS.WRP$_REPORTS_TIME_BANDS: ADDED INTERVAL PARTITION SYS_P869 (4434) VALUES LESS THAN (TO_DATE(' 2022-02-21
01:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN'))
2022-02-21T11:32:12.822559+01:00
ALTER SYSTEM SET streams_pool_size=256M SCOPE=BOTH;
```

Log Files

- Check for Data Pump trace files in `$ORACLE_BASE/diag/rdbms/../../trace`

```
Trace file /u01/app/oracle/diag/rdbms/db19/DB19/trace/DB19_dm00_17468.trc
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.14.0.0.0
Build label:    RDBMS_19.14.0.0.0DBRU_LINUX.X64_211224.3
ORACLE_HOME:    /u01/app/oracle/product/19
System name:    Linux
Node name:      hol.localdomain
Release:        5.4.17-2136.302.7.2.1.el7u
Version:        #2 SMP Tue Jan 18 13:44:44
Machine:        x86_64
Instance name: DB19
Redo thread mounted by this instance: 1
Oracle process number: 58
Unix process pid: 17468, image: oracle@hol


*** 2022-02-21T11:33:25.374300+01:00
*** SESSION ID:(253.19643) 2022-02-21T11:3
*** CLIENT ID:() 2022-02-21T11:33:25.37431
*** SERVICE NAME:(SYS$USERS) 2022-02-21T11
*** MODULE NAME:(Data Pump Master) 2022-02
*** ACTION NAME:(SYS_EXPORT_SCHEMA_01) 202
*** CLIENT DRIVER:() 2022-02-21T11:33:25.374327+01:00
```

```
=================== skgfqio Request Dump =======================
OSD Context: aiopend=0, aiodone=0, limfsiz=4294967295l, sigwinchslot=0
Request flags: READ
- - - - skgfrrq request element 1 - - - -
BLOCKNO = 1
IOV: addr=0x0x6ef687d8, fib=0x0x6d0d2478, maxaio=0, seal=0x45726963,
fd=260
     fsync required?=TRUE, offset=18446744073709551615, aiopend=0
FIB: addr=0x0x6d0d2478, lblksiz=0, ora ftype=18, pblksiz=512, filsiz=1
     maxvec=16, fname=/home/oracle/dp/export.log, serr=0, seal=0x45726963
     fstype=0x58465342, unix ftype=0x81a4, last
block=18446744073709551615
IOSB: addr=0x0x7f0da829dc38, status=3, time=0, qstatus=8,AIO start
time=139696632618072
err=27072 errno=25 ose[0]=4 ose[1]=1 ose[2]=333
BUFFER: addr=0x0x7f0da76b2000, len=4096
```

Background Process

CONTROL PROCESS

Typically one: **`dm00`**

```
DB19_dm00_17468.trc
```

WORKERS

Typically many: **`dwnn`**

```
DB19_dw00_17469.trc
DB19_dw01_17470.trc
DB19_dw02_17471.trc
DB19_dw03_17472.trc
```

Troubleshooting

1. LOGS

Find and get the most out of the log files

2. VIEWS

Using views inside the database to monitor

3. TRACE

Enabling trace to debug a specific issue

Views

Monitor a Data Pump process in **DBA_DATAPUMP_JOBS**

```
$ expdp ... job_name=MYEXPDP1


SQL> select * from DBA_DATAPUMP_JOBS;

OWNER_NAME  JOB_NAME  OPERATION  JOB_MODE  STATE  DEGREE  ATTACHED  DATAPUMP_SESSIONS
----------- --------- ---------- --------- ------ ------- --------- ------------------
SYS         MYEXPDP1  EXPORT     FULL      EXECUT       1         1                  3
```

Use **DBA_DATAPUMP_SESSIONS** as well

Views

Monitor a Data Pump process in `V$SESSION_LONGOPS`

```
$ expdp ... job_name=MYEXPDP1

SQL> select sid, serial#, sofar, totalwork
     from   v$session_longops
     where  opname = 'MYEXPDP1' and sofar != totalwork;
```

`sofar`	Shows how much work in MB has been done so far in relation to `totalwork`
`totalwork`	Shows the total amount of work in MB

Monitoring

Important MOS notes:

- How To Monitor The Progress Of Datapump Jobs (Doc ID 1471766.1)
- Finding Out The Current SQL Statement A Data Pump Process Is Executing (Doc ID 1528301.1)
- How can we monitor a DataPump Job's Progress? (Doc ID 455720.1)

Troubleshooting

1. LOGS

Find and get the most out of the log files

2. VIEWS

Using views inside the database to monitor

3. TRACE

Enabling trace to debug a specific issue

TRACING

Command Line

Using parameter **TRACE**

Interactive Console

Using command **TRACE**

Trace Event

Adding specific event to SPFile

Tracing | Best Practice

- Requires privileged user or role
  - `DBA`
  - `EXP_FULL_DATABASE`
  - `IMP_FULL_DATABASE`

- Ensure **`MAX_DUMP_FILE_SIZE`** is large enough to capture the trace (default=unlimited)

- Most important **`TRACE`** bitmaps:
  - `1FF0300` Recommended Tracing
  - `1FFF0300` Full Tracing
  - For a comprehensive list and further explanation, see MOS Note: 286496.1

Data Pump trace is written to **`dm`*`nn`*** and **`dw`*`nn`*** trace files

- Located in trace directory in `diagnostic_dest`

Tracing

```
SQL> # Data Pump specific trace
SQL> alter system set events = '39089 trace name context forever, level 0x300' ;

SQL> # Multipurpose SQL trace
SQL> alter session set events '10046 trace name context forever,level 8';
```

Tracing

Important MOS notes:

- Export/Import DataPump Parameter TRACE - How to Diagnose Oracle Data Pump (Doc ID 286496.1)

- How To Collect 10046 Trace (SQL_TRACE) Diagnostics for Performance Issues (Doc ID 376442.1)

Extracting metadata

Creating big indexes manually

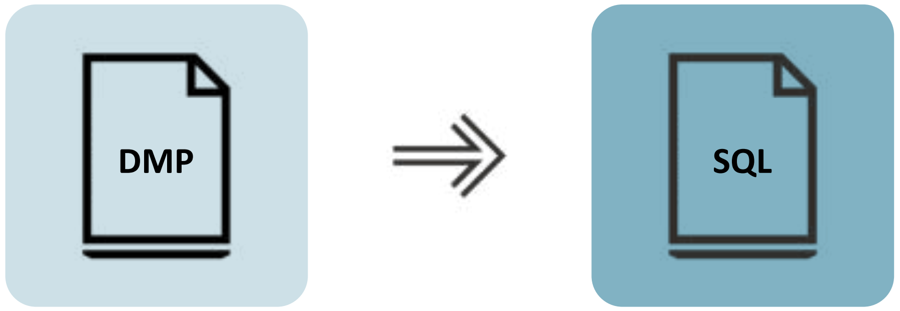
You can extract metadata from a dump file using parameter `SQLFILE`

Generate SQL Statements

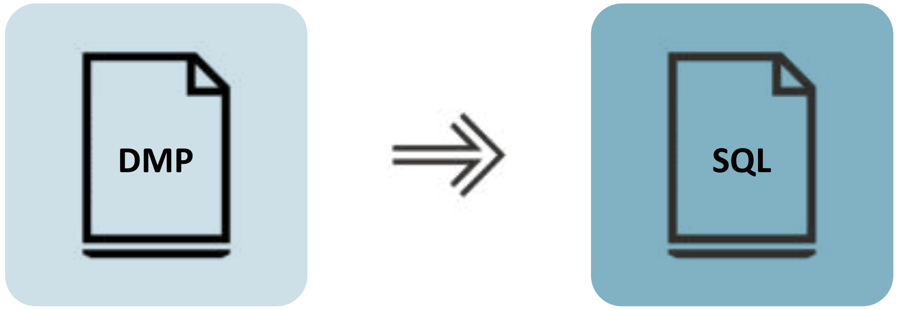
Generate DDLs that **`impdp`** will execute

```
$ more import.par
...
sqlfile=all_statements.sql
...

$ impdp system parfile=import.par
```



Generate SQL Statements



```
CREATE USER "TPCC" IDENTIFIED BY VALUES '...'
      DEFAULT TABLESPACE "TPCCTAB"
      TEMPORARY TABLESPACE "TEMP";
GRANT UNLIMITED TABLESPACE TO "TPCC";
GRANT "CONNECT" TO "TPCC";
GRANT "RESOURCE" TO "TPCC";
DECLARE
  TEMP_COUNT NUMBER;
  SQLSTR VARCHAR2(200);
BEGIN
  SQLSTR := 'ALTER USER "TPCC" QUOTA UNLIMITED ON "TPCCTAB"';
  EXECUTE IMMEDIATE SQLSTR;
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE = -30041 THEN
      SQLSTR := 'SELECT COUNT(*) FROM USER_TABLESPACES
              WHERE TABLESPACE_NAME = ''TPCCTAB'' AND CONTENTS =
''TEMPORARY''';
      EXECUTE IMMEDIATE SQLSTR INTO TEMP_COUNT;
      IF TEMP_COUNT = 1 THEN RETURN;
      ELSE RAISE;
      END IF;
    ELSE
      RAISE;
    END IF;
END;
/
```

Example

Creating big indexes

Creating Indexes

Imagine a schema

Tables	
Small tables	10.000
Big tables	1
Indexes	
Small indexes	10.000
Big indexes	1

Creating Indexes

Data Pump creates indexes with parallel degree 1

Many indexes are created simultaneously

Very efficient for many small indexes

Very <u>in</u>efficient for large indexes

Creating Indexes

Data Pump creates small indexes

You create big indexes with desired parallel degree

Creating Indexes

- Find indexes of interest

```
SQL> select   segment_name, round(bytes/1024/1024/1024) as GB
     from     user_segments
     where    segment_type='INDEX'
     order by GB desc;
```

- Exclude indexes from import

```
$ cat import.par
...
exclude=INDEX:"='BIG1','BIG2','BIG3'"
...

impdp ... parfile=import.par
```

Creating Indexes

- Generate metadata for big indexes

```
$ cat import-sqlfile.par
...
include=INDEX:"='BIG1','BIG2','BIG3'"
sqlfile=index.sql
...

impdp ... parfile=import-sqlfile.par
```

- Change parallel degree and create indexes

```
SQL> CREATE INDEX BIG1 .... PARALLEL n;
SQL> ALTER INDEX INDEX BIG1 .... NOPARALLEL;
...
```

DEMO

Creating big indexes

Watch on YouTube

You can also get index definition from **`DBMS_METADATA.GET_DLL`**

Starting, stopping and restarting Data Pump jobs

Restartability

You can restart an export job after the *estimate* phase has been completed

- Transportable Tablespace jobs as of Oracle Database 21c

Restart | Export

- Tracked in the Control Table
- Workers create/update records with COMPLETION_TIME
- Restart: Workers check for records with missing COMPLETION_TIME

OBJECT_TYPE	START_TIME	COMPLETION_TIME
TABLESPACE	12-SEP-2021:9:04.01	12-SEP-2021:9:05.23
USER	12-SEP-2021:9:05.27	

- Example
  - USER object is incomplete
  - Will be removed and restarted

You can restart an import job using information from control table

- Transportable Tablespace jobs as of Oracle Database 21c

Restart | Import

- Workers track import status via **STATE** and **STATUS**

OBJECT	OBJECT_SCHEMA	OBJECT_NAME	PROCESSING_STATE	PROCESSING_STATUS
TABLE	SCOTT	EMP	W	C
TABLE	SCOTT	DEPT	U	C
INDEX	SCOTT	IDX1_EMP	R	C
INDEX	SCOTT	IDX1_DEPT	R	C

- R = objects were Retrieved (exported)
- C = objects are Current (successfully imported)
- W = objects are Written (imported)
- U = objects are Unknown (import started but did not finish)

Data Pump

Architecture

Bundle Patch

Troubleshooting

New Features

- Exclude
- Include
- Checksum

Combine the use of `INCLUDE` and `EXCLUDE` in the same job

Include and Exclude

```
$ expdp ... schemas=hr,oe include=table exclude=statistics
```

Avoid corruption and ensure dump file integrity using checksum

Checksum

```
$ # Calculate a checksum using the designated algorithm
$ # Stored encrypted in dump file header
$ expdp ... checksum_algorithm=sha384

$ # Verify the sum, no import
$ impdp ... verify_only=yes

$ # Verify the checksum and import
$ # Default, if dump file has a checksum
$ impdp ... verify_checksum=yes
```

Universal Data Pump Client

Before: client and server version had to match

- 12.1.0.2 client to expdp from 12.1.0.2 server, 19c client to impdp to 19c server

Now (since 21c): Data Pump client is backward compatible

- Always use the latest client, attach to any supported database version
- Data Pump and SQL*Loader clients are in the "Tools" package of the Instant Client

Alex Zaballa

CloudWorld

CHRISTIAN, HUSBAND, FATHER, DBA, ACE DIRECTOR, WRITER AT OTN, SPEAKER AND BLOGGER.

Born

Double Citizenship

Living

22 years – Oracle Technologies
Oracle Developer: 2000 – 2007
Oracle DBA: 2007 – 2022
Oracle Cloud Architect: 2016 – 2022
400 Oracle Certifications

ACCENTURE ENKITEC GROUP

ORACLE Diamond Partner

ORACLE Diamond Partner Global Cloud Elite

ORACLE ACE Director

ORACLE Certified Master

ORACLE Certified Professional

ORACLE Certified Expert

ORACLE Certified Specialist

ORACLE Certified Associate

aws CERTIFIED

Microsoft CERTIFIED AZURE FUNDAMENTALS

ORACLE PartnerNetwork Solution Engineering Specialist

ORACLE PartnerNetwork Sales Specialist

Google Cloud Certified Data Engineer

Google Cloud Certified Cloud Architect

http://alexzaballa.blogspot.com/

@alexzaballa

https://www.linkedin.com/in/alexzaballa

GUOB

TOP REASONS TO LOVE DATA PUMP

1. Similar look and feel to the old exp/imp (old school DBAs)
2. The most flexible tool to deal with data migration: Cross-Platform, Multi-Versions, Reorg, etc.
3. Granularity
4. Remap: datafiles, tablespaces, schemas
5. Network_Link: No need to generate files

TOP REASONS TO LOVE DATA PUMP

6. High Speed
7. PL/SQL Interface
8. Easy to track and troubleshooting
9. Resumable
10. Interactive command line

TOP REASONS TO LOVE DATA PUMP

11. Parallelism

12. Compression

13. Consistency

14. Patches availability and quick fixes

15. Transportable tablespaces

TOP REASONS TO LOVE DATA PUMP 21C

1. INCLUDE and EXCLUDE in the Same Operation: INCLUDE and EXCLUDE parameters can be part of the same command. In previous releases INCLUDE and EXCLUDE parameters were mutually exclusive.

2. Parallelizable even for transportable tablespace metadata operations

3. JSON Data Type Support

4. CHECKSUM

TOP REASONS TO LOVE DATA PUMP 21C

5. Index Compression

6. Export to and Import From Cloud Object Stores: It’s no longer restricted to the Autonomous Database.

   - Export to an object store (Oracle Cloud only)
   - Import from an object store (Oracle Cloud, S3, Azure Blob Storage)

CONSIDERATIONS

Data Pump is not a backup, but it’s a great addition to the backup policy

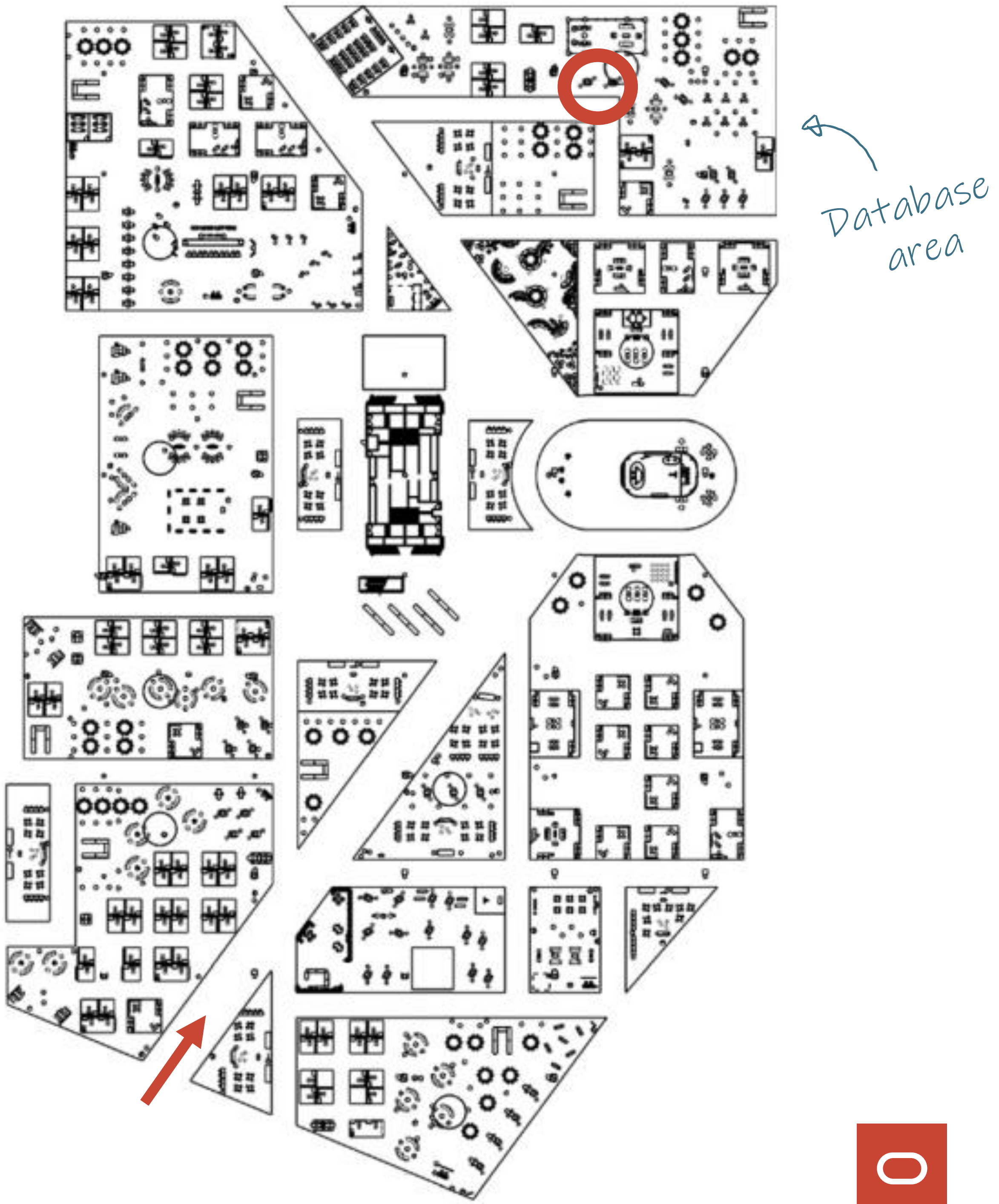
Visit our

DEMO BOOTH

DB-18

Oracle CloudWorld Hub

Tuesday	13-19
Wednesday	08-18
Thursday	08-14



AutoUpgrade 2.0: Internals and New Features

LRN3500	Thursday 13:15	Murano 3202, The Venetian, Level 3

Cloud Premigration Advisor Tool - Your Cloud Premigration Advisor

LIT4104	Thursday 13:40	Ascend Lounge, CloudWorld Hub, The Venetian

ORACLE
CloudWorld

Thank you

Alex Zaballa

Practice Director, Accenture

Roy Swonger

Vice President, Oracle

Mike Dietrich

Senior Director Product Management, Oracle

Data Pump | Documentation

- Oracle Database 19c – Utilities Guide
- Oracle Database 21c – Utilities Guide
- PL/SQL Packages and Types Reference - DBMS_DATAPUMP

- ADB Data Pump Export to Object Store
- ADB Import Data Using Oracle Data Pump

DBMS_DATAPUMP

Client

```
expdp directory=mydir \
   logfile=exp.log \
   dumpfile=exp%u.dmp \
   schemas=app \
   parallel=4 \
   metrics=y \
   logtime=all
```

API

```
h1 := DBMS_DATAPUMP.OPEN(
   operation => 'EXPORT',
   job_mode => 'SCHEMA',
   remote_link => null,
   job_name => 'MY_JOB',
   version => null);

-- Create a Data Pump job to do a schema
-- export. Give it a meaningful name
```

DBMS_DATAPUMP | Comparison

Client

```
expdp directory=mydir \
   logfile=exp.log \
   dumpfile=exp%u.dmp \
   schemas=app \
   parallel=4 \
   metrics=y \
   logtime=all
```

API

```
DBMS_DATAPUMP.METADATA_FILTER(
   handle => h1,
   name => 'SCHEMA_EXPR',
   value => 'IN (''APP'')');

-- Specify the schema to be exported. We let
-- the object_path parameter default in this
-- call, so this applies to all objects in
-- the job
```

DBMS_DATAPUMP | Comparison

Client	API

Client

```
expdp directory=mydir \
   logfile=exp.log \
   dumpfile=exp%u.dmp \
   schemas=app \
   parallel=4 \
   metrics=y \
   logtime=all
```

API

```
DBMS_DATAPUMP.METADATA_FILTER(
   handle => h1,
   name => 'SCHEMA_EXPR',
   value => 'IN (''APP'')');

-- Specify the schema to be exported. We let
-- the object_path parameter default in this
-- call, so this applies to all objects in
-- the job
```

DBMS_DATAPUMP | Comparison

Client

```
expdp directory=mydir \
   logfile=exp.log \
   dumpfile=exp%u.dmp \
   schemas=app \
   parallel=4 \
   metrics=y \
   logtime=all
```

API

```
DBMS_DATAPUMP.ADD_FILE(
   handle => h1,
   filename => 'exp%u.dmp',
   directory => 'MYDIR',
   filetype=>DBMS_DATAPUMP.KU$_FILE_TYPE_DUMP_FILE);

-- Specify the dumpfile for the job using a
-- wildcard. The directory object must be
-- supplied for each file added to the job
-- FILETYPE defaults to dumpfile but we
-- specify it anyway to be clear
```

DBMS_DATAPUMP | Comparison

Client	API

Client

```
expdp directory=mydir \
   logfile=exp.log \
   dumpfile=exp%u.dmp \
   schemas=app \
   parallel=4 \
   metrics=y \
   logtime=all
```

API

```
DBMS_DATAPUMP.ADD_FILE(
   handle => h1,
   filename => 'exp.log',
   directory => 'MYDIR',
   filetype=>DBMS_DATAPUMP.KU$_FILE_TYPE_LOG_FILE);

-- Specify the log file for the job. The directory
-- object must be supplied for each file added to
-- the job.
```

DBMS_DATAPUMP | Comparison

Client	API

Client

```
expdp directory=mydir \
   logfile=exp.log \
   dumpfile=exp%u.dmp \
   schemas=app \
   parallel=4 \
   metrics=y \
   logtime=all
```

API

```
DBMS_DATAPUMP.SET_PARALLEL(
   handle => h1,
   degree => 4 );

-- Set the parallelism for the job
-- Or get a little creative

select value into parallel_degree
from   v$parameter
where  name='cpu_count';
DBMS_DATAPUMP.SET_PARALLEL(
   handle => h1,
   degree => parallel_degree);
```

DBMS_DATAPUMP | Comparison

Client

```
expdp directory=mydir \
   logfile=exp.log \
   dumpfile=exp%u.dmp \
   schemas=app \
   parallel=4 \
   metrics=y \
   logtime=all
```

API

```
DBMS_DATAPUMP.SET_PARAMETER(
   handle => h1,
   name => 'METRICS',
   value => 1);

DBMS_DATAPUMP.SET_PARAMETER(
   handle => h1,
   name => 'LOGTIME',
   value => 'ALL');

-- set other job parameters
```

DBMS_DATAPUMP | Comparison

Client	API
	`DBMS_DATAPUMP.START_JOB ( handle => h1);` `-- now start the job` `-- wait for it to complete` `DBMS_DATAPUMP.WAIT_FOR_JOB ( handle => h1, job_state);`

```
DBMS_DATAPUMP.START_JOB (
   handle => h1);

-- now start the job
-- wait for it to complete

DBMS_DATAPUMP.WAIT_FOR_JOB (
   handle => h1,
   job_state);
```