Using Oracle Data Pump

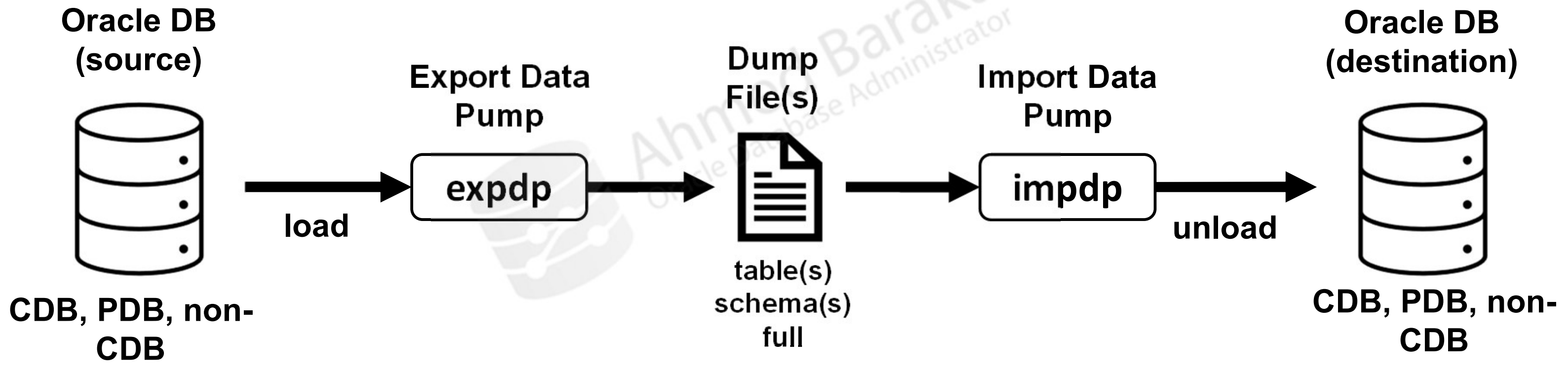
By Ahmed Baraka

Objectives

In this lecture, you will learn how to perform the following:

- Describe Data Pump and its data movement methods
- Use directory objects
- Understand and use Data Pump modes
- Invoke Data Pump utilities in command-line
- Use Export basic parameters
- Use Import basic parameters
- Implement Parallelism in Data Pump

Oracle Data Pump: Command-line Clients, Conventional Method



About Oracle Data Pump

- A technology to move database objects from one Oracle database to another
- When it is used for moving data, it relies on ***logical*** data movement (portable but not efficient)
- Supported interfaces: command-line utilities (**`expdp`**, **`impdp`**) and PL/SQL (**`DBMS_DATAPUMP`** and **`DBMS_METADATA`**)
- Provides four data movement methods:
  - **Conventional path**: most flexible, least efficiency
  - **Direct path**: SQL layer is bypassed, second more efficient, restrictions
  - **External tables**: creates an external table that maps to a dump file
  - **Data file copying**: the most efficient, self-contained data files, restrictions
  - **Network link support**: data transfer via network links

Oracle Data Pump Benefits and Features

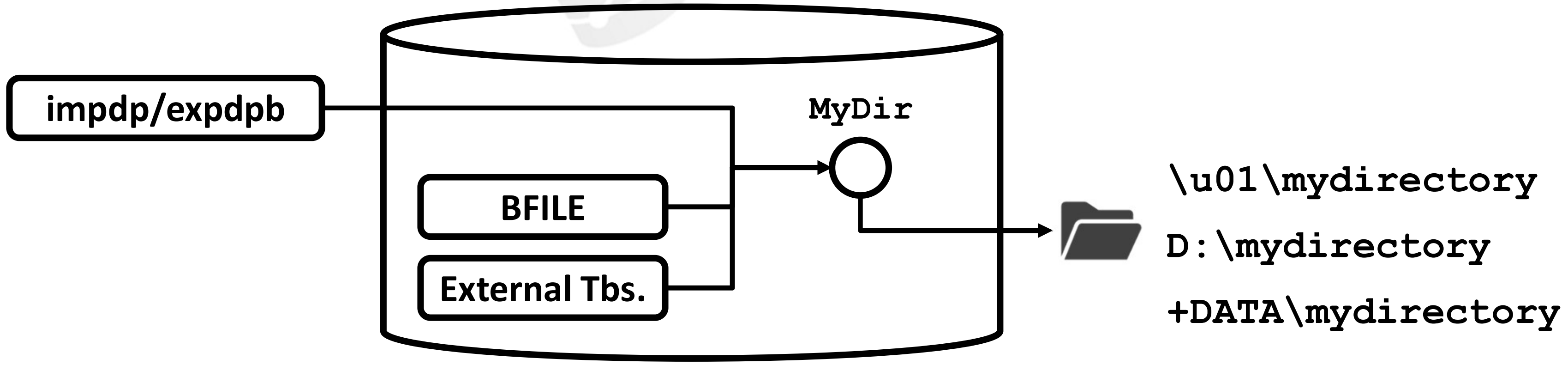
- Fine-grained object and data selection
- Compression of data during a Data Pump export (licensed)
- Security through encryption (licensed)
- Explicit specification of database version
- Parallel execution
- Network mode in a distributed environment
- Estimation of export job space consumption

Oracle Data Pump Usage

- Moving data from one database to another
- Database migration and upgrade
- Data archiving
- Testing and development
- Logical backup and recovery

About Directory Objects

- An alias for a directory on the server file system
- Can be accessed by BFILEs, external tables, PL/SQL code, Data Pump import and export utilities
- Better than hard coding file system path names
- Are **not** owned by individual schema



Creating Directory Objects

- **`CREATE ANY DIRECTORY`** system privilege is needed
- The creator will automatically have the **`READ`**, **`WRITE`**, and **`EXECUTE`** object privileges on the created directory object.
- The common syntax:

```
CREATE [OR REPLACE] DIRECTORY <dir-name> AS '<directory-path>'
```

- Example:

```
CREATE DIRECTORY DP_DIR AS '/u01/dpfiles';
GRANT READ, WRITE ON DIRECTORY DP_DIR TO HR
```

Directory Objects and Data Pump

- **Non-CDB:**
  - **`DATA_PUMP_DIR`** is created automatically and granted to **`SYS`** and **`SYSTEM`**
  - It points to the following directory:

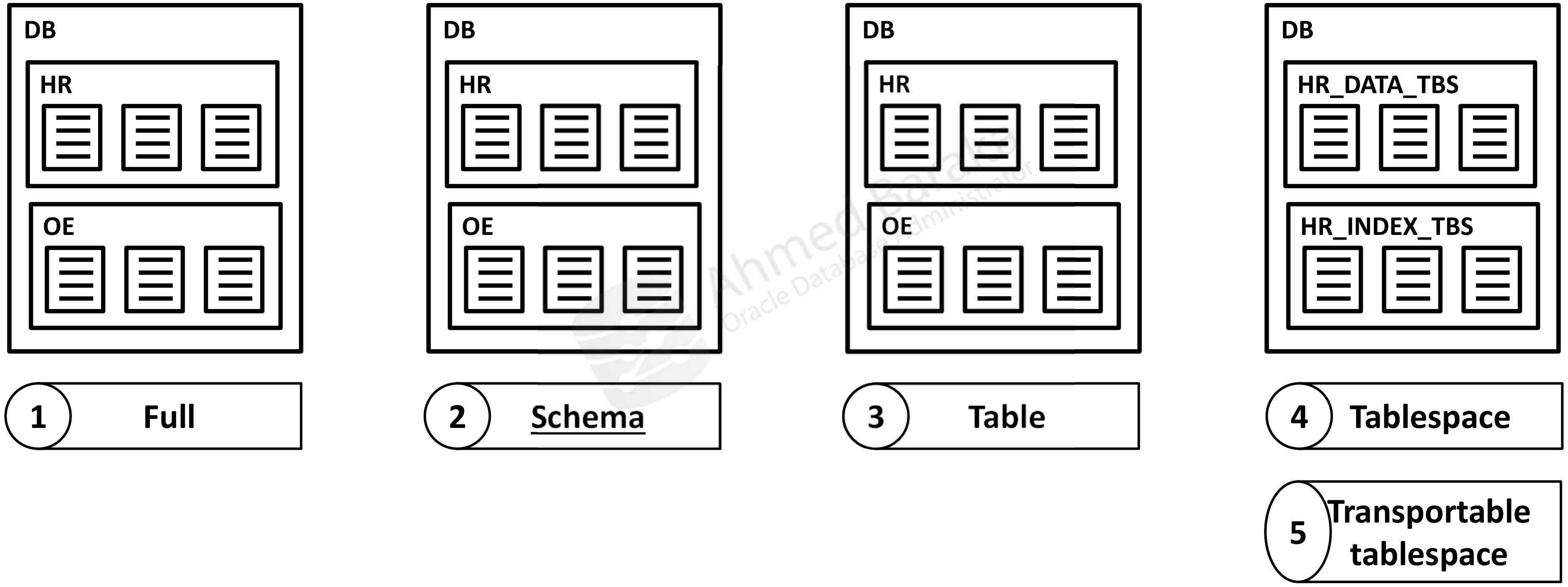
```
$ORACLE_HOME/admin/$ORACLE_SID/dpdump/
```

  - To allow a user access it via the Data Pump utilities:

```
GRANT READ, WRITE ON DATA_PUMP_DIR TO HR;
```

- **CDB:**
  - The default Data Pump directory object is automatically created in the root but not in the PDBs.
  - Explicitly create an object directory in the PDB and grant access on it to the user

Data Pump Export and Import Modes



Data Pump Export and Import Modes and Their Corresponding Parameters

Mode	Parameter	Example
Full	`FULL`	`FULL=Y`
Schema	`SCHEMAS`	`SCHEMAS=HR,SOE`
Table	`TABLES`	`TABLES=HE.EMPLOYEES,HR.JOB`
Tablespace	`TABLESPACES`	`TABLESPACES=HR_DATA_TBS`
Transportable Tablespace	`TRANSPORT_TABLESPACES`	`TRANSPORT_TABLESPACES=HR_DATA_TBS`

Invoking Data Pump Export and Import Utilities in Command-line Mode

- The roles **`DATAPUMP_EXP_FULL_DATABASE`** and **`DATAPUMP_IMP_FULL_DATABASE`** are required
- Invoking Data Pump Export in command line:

```
expdp <parameter>=<value>
```

- Invoking Data Pump Import in command line:

```
impdp <parameter>=<value>
```

Data Pump Export: Basic Examples

- Export **HR** schema into **expdat.dmp** :

```
expdp HR/ABcd##1234@oradb directory=DATA_PUMP_DIR
```

- Export **HR** schema into **hr.dmp** :

```
expdp HR/ABcd##1234@oradb directory=DATA_PUMP_DIR dumpfile=hr.dmp
```

- Export **HR** schema into **hr.dmp** and log output into **hr.log**:

```
expdp HR/ABcd##1234@oradb directory=DATA_PUMP_DIR dumpfile=hr.dmp
logfile=hr.log
```

```
expdp HR/ABcd##1234@oradb dumpfile=DATA_PUMP_DIR:hr.dmp
logfile=DP_LOG_DIR:hr.log
```

- **SYSTEM** exports **HR** schema into **hr.dmp**

```
expdp system/ABcd##1234@oradb schemas=HR directory=MY_DIR
dumpfile=hr.dmp logfile=hr.log
```

Data Pump Export: Basic Examples

- Export multiple schemas:

```
expdp system/ABcd##1234@oradb schemas=HR,SOE directory=DATA_PUMP_DIR
dumpfile=users.dmp logfile=users.log
```

- Export specific tables:

```
expdp HR/ABcd##1234@oradb directory=DATA_PUMP_DIR dumpfile=hr.dmp
logfile=hr.log tables="HR"."EMPLOYEES","HR"."JOBS"
```

- Export all non-system schemas with no log file:

```
expdp system/ABcd##1234@oradb full=yes directory=DATA_PUMP_DIR
dumpfile=oradb.dmp nologfile=yes
```

Using Parameter Files

- Applies to Export and Import utilities
- Pass the parameter file name to **`parfile`**:

```
expdp parfile=myfile.txt
```

```
cat myfile.txt

userid=hr/ABcd##1234@oradb
directory=mydirectory
dumpfile=hr%U.dmp
logfile=hr.log
filesize=50M
```

Specifying Dump Files and Their Sizes

- Produce a single dump file:

```
DUMPFILE=hr.dmp
```

- Produce multiple files:

```
DUMPFILE=hr1.dmp,hr2.dmp
```

- Using the substitution variable `%U` to produce multiple files:

```
DUMPFILE=hr%U.dmp
```

- You can control the maximum dump file size:

```
FILESIZE=1G DUMPFILE=hr%U.dmp
```

Export Filtering Parameters

- Use **`EXCLUDE`** and **`INCLUDE`** parameters (mutual exclusive) to exclude or include specific object or object types.
- The specified objects and all their dependent objects are exported.
- Examples:

```
EXCLUDE=INDEX
EXCLUDE=TABLE:"LIKE 'EMP%'"
EXCLUDE=SCHEMA:"='HR'"
INCLUDE=TABLE:"IN ('EMP', 'DEPT')"
INCLUDE=PARTITION:"IN ('EMPLOYEES:P1','DEPARTMENTS:P2')"
INCLUDE=TABLE,INDEX,VIEW,SEQUENCE
```

Estimating Export File Size

- Set the parameter **`ESTIMATE_ONLY`** to **`Yes`** (default is **`No`**) to make the Export utility estimates the dump file size without actually performing the export.
- Set the estimation method by setting the parameter **`ESTIMATE`**:

```
ESTIMATE=[BLOCKS | STATISTICS]
```

- Example:

```
expdp hr TABLES=employees ESTIMATE=STATISTICS DIRECTORY=mydir
ESTIMATE_ONLY=yes
```

Export via the Network

- Set the parameter **`NETWORK_LINK`** to the name of the database link that points to the source database.
- **Export**
  - Target: create a dump file locally by transferring the data from the source database
  - Dump files are saved in the **local** database server
  - Database link is needed to point to the **source database**
  - **`EXP_FULL_DATABASE`** privilege is required in each database

```
expdp hr/ABcd##1234 DIRECTORY=dpump_dir1 NETWORK_LINK=SOURCE_HR
DUMPFILE=hr.dmp
```

Import via the Network

- **Import**:
  - No dump file is generated; the objects are created directly from the source into the local database.
  - Directory object is still needed for the log file
  - **`EXP_FULL_DATABASE`** privilege is required in each database

```
impdp hr/ABcd##1234 tables=EMPLOYEES NETWORK_LINK=REMOTE_HR
DIRECTORY=dpump_dir1 LOGFILE=hr_emp.log
```

Preserving Data Consistency

- By default, data consistency is maintained at the table level
- To maintain data consistency at the export job level, set **`CONSISTENT=Y`**
  - All export job is executed in a single transaction (internally it issues **`SET TRANSACTION READ ONLY`**)
  - If high update transactions are being applied during export, the export requires large undo size and it takes longer time to finish
  - Avoid setting **`CONSISTENT=Y`** when the database is under high transactions volume on the exported data
- Can be used with **`FLASHBACK_TIME`** parameter

```
expdp SCHEMAS=hr DIRECTORY=mydir DUMPFILE=hr_dump.dmp CONSISTENT=Y
FLASHBACK_TIME="TO_TIMESTAMP('2023-02-18 14:00:00', 'YYYY-MM-DD
HH24:MI:SS')"
```

Encrypting Dump Files

- To encrypt the dump file contents, set an encryption password using the parameter **`ENCRYPTION_PASSWORD`**

```
expdp hr DIRECTORY=dpump_dir1 ... ENCRYPTION_PASSWORD=myComplexPass
```

- To enter the password after invoking the Export utility, set **`ENCRYPTION_PWD_PROMPT=YES`**
- Encryption algorithm can be set:

```
ENCRYPTION_ALGORITHM = [AES128 | AES192 | AES256]
```

- Oracle Data Pump encryption features require that the **Oracle Advanced Security** option be enabled in the Enterprise Edition license.
- Database wallet (if being used) is still integrated with Data Pump

Moving Data Across Different Oracle Database Versions

- Using **`VERSION`** parameter generates an Oracle Data Pump dump file set that is compatible with a specific version.

```
VERSION=[ COMPATIBLE | LATEST | <version_string> ]
```

- Set the **`VERSION`** parameter value to be the same version as the target.
- Incompatible objects or types are not exported.
- Example:

```
expdp HR/ABcd##1234@oradb directory=MY_DIR dumpfile=hr.dmp
logfile=hr.log version=12
```

Import Utility Fundamental Parameters

- **Import Mode-Related Parameters:**
  - You can perform a Data Pump import in all the modes: **`FULL`**, **`SCHEMAS`**, **`TABLES`**, and **`TABLESPACES`** parameters
- **File- and Directory-Related Parameters:**
  - The Data Pump import utility uses the **`PARFILE`**, **`DIRECTORY`**, **`DUMPFILE`**, **`LOGFILE`**, and **`NOLOGFILE`** parameters in the same way as the Data Pump export utility.
- **Filtering Parameters:**
  - The Data Pump import utility uses the **`EXCLUDE`** and **`INCLUDE`** parameters in the same way as the Data Pump export utility.

Handling Existing Tables

- To define what the import should do when a table already exists, set the parameter **`TABLE_EXISTS_ACTION`**:

Parameter Value	**Description**
`SKIP`	Skip the current table and move to the next table
`APPEND`	Insert import data and leave the existing data untouched. Existing space is not used.
`TRUNCATE`	Truncate the existing table and then load the import data
`REPLACE`	Drop the existing table, create new table, and load it with the import data

Remapping Parameters

- Remapping tables: change table names
  - Tables will not be remapped if they already exist

```
REMAP_TABLE=hr.docs:docs2
```

- Remapping schemas: change schema name

```
REMAP_SCHEMA=hr:soe
```

- Remapping tablespaces: change table tablespaces

```
REMAP_TABLESPACE=hr_tbs:new_tbs
```

- Remapping data: modify the data in a column of a table (next slide)

Remapping Data

- **`REMAP_DATA`** applies a function on a column value in the dump file:

```
REMAP_DATA=[schema.]tablename.column_name:[schema.]pkg.function
```

- Function parameter and return datatype must match column data type
- Function example:

```
CREATE OR REPLACE PACKAGE BODY DATAPUMP_REMAP AS
 FUNCTION MASK_CREDIT_CARD (P_VALUE NUMBER) RETURN NUMBER IS
  BEGIN
 ...
```

- Using the parameter example:

```
.. REMAP_DATA=CUSTOMERS.CREDIT_CARD:DATAPUMP_REMAP.MASK_CREDIT_CARD
```

Note: This mapping parameter can be used in export utility as well.

Remapping Parameters

- Remapping datafiles:

```
REMAP_DATAFILE="'C:\DB1\HR\tbs.dbf':'/db1/hr/tbs.dbf'"
```

- Remapping directories:

```
REMAP_DIRECTORY="'D:\mydir':'/mydir'"
```

- Used with SQL statements that include datafiles or directory path, like **`CREATE TABLESPACE`** , **`CREATE LIBRARY`**, and **`CREATE DIRECTORY`**.

Parallelism in Data Pump

- Allows the Data Pump job to run more than one active execution process (worker processes + I/O server processes) to execute the job
- Aims at utilizing the resources for best elapsed time
- Is set by the following parameter:

```
PARALLEL=<integer>
```

- Best used with the `%U` substitution variable:

```
... DUMPFILE=exp%u.dmp PARALLEL=4
```

- You do not need to set the same parallelism level at the export and import side.
- Requires EE license.

More Useful Parameters

- To display timestamps for each export/import job message, set the parameter **`LOGTIME`** to **`ALL`**

```
LOGTIME=[ NONE | STATUS | LOGFILE | ALL ]
```

- To include export/import metrics (number of objects and elapsed time) in the log file, enable **`METRICS`** parameter

```
METRICS=[ YES | NO ]
```

Monitoring Data Pump Jobs

View	Description
`DBA_DATAPUMP_JOBS`	Shows summary information of all currently running Data Pump jobs.
`V$SESSION_LONGOPS`	Use it to monitor the progress of an export/import jobs

Summary

In this lecture, you should have learnt how to perform the following:

- Describe Data Pump and its data movement methods
- Use directory objects
- Understand and use Data Pump modes
- Invoke Data Pump utilities in command-line
- Use Export basic parameters
- Use Import basic parameters
- Implement Parallelism in Data Pump