17

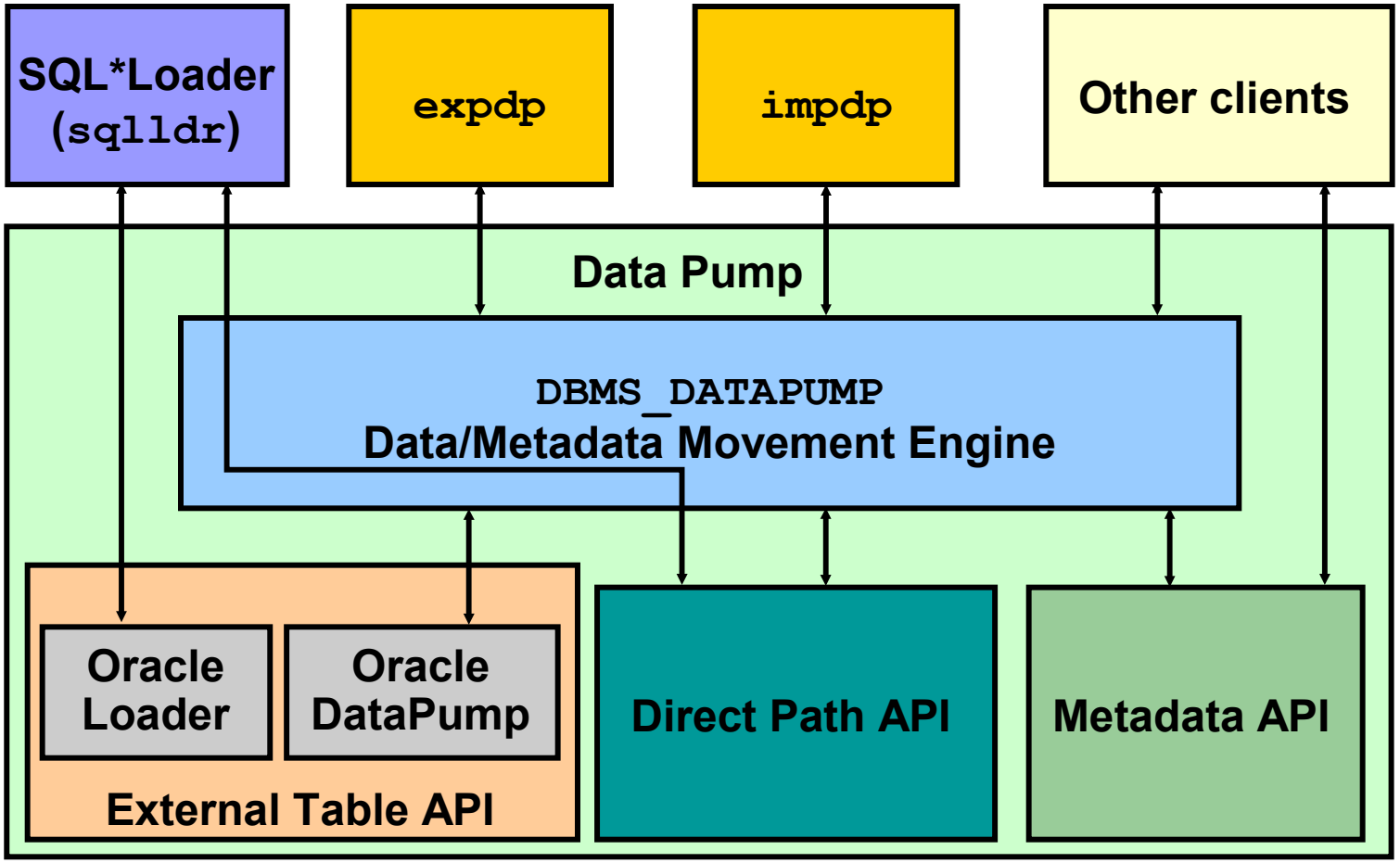
Moving Data

Objectives

After completing this lesson, you should be able to:

- Describe ways to move data
- Create and use directory objects
- Use SQL*Loader to load data from a non-Oracle database (or user files)
- Use external tables to move data via platform-independent files
- Explain the general architecture of Oracle Data Pump
- Use Data Pump Export and Import to move data between Oracle databases

Moving Data: General Architecture



Oracle Data Pump: Overview

As a server-based facility for high-speed data and metadata movement, Oracle Data Pump:

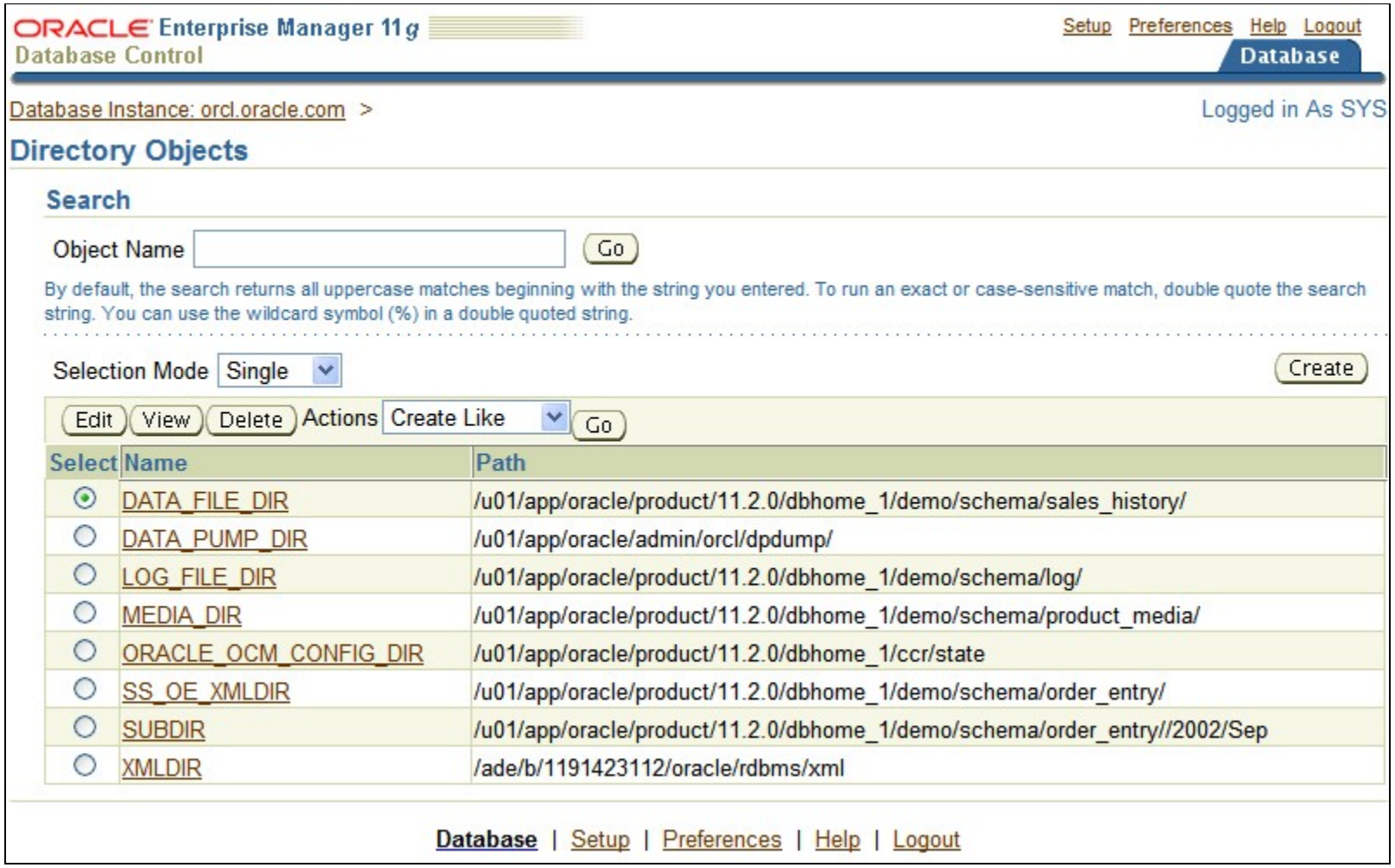
- Is callable via `DBMS_DATAPUMP`
- Provides the following tools:
  - `expdp`
  - `impdp`
  - Web-based interface
- Provides four data movement methods:
  - Data file copying
  - Direct path
  - External tables
  - Network link support
- Detaches from and reattaches to long-running jobs
- Restarts Data Pump jobs

Oracle Data Pump: Benefits

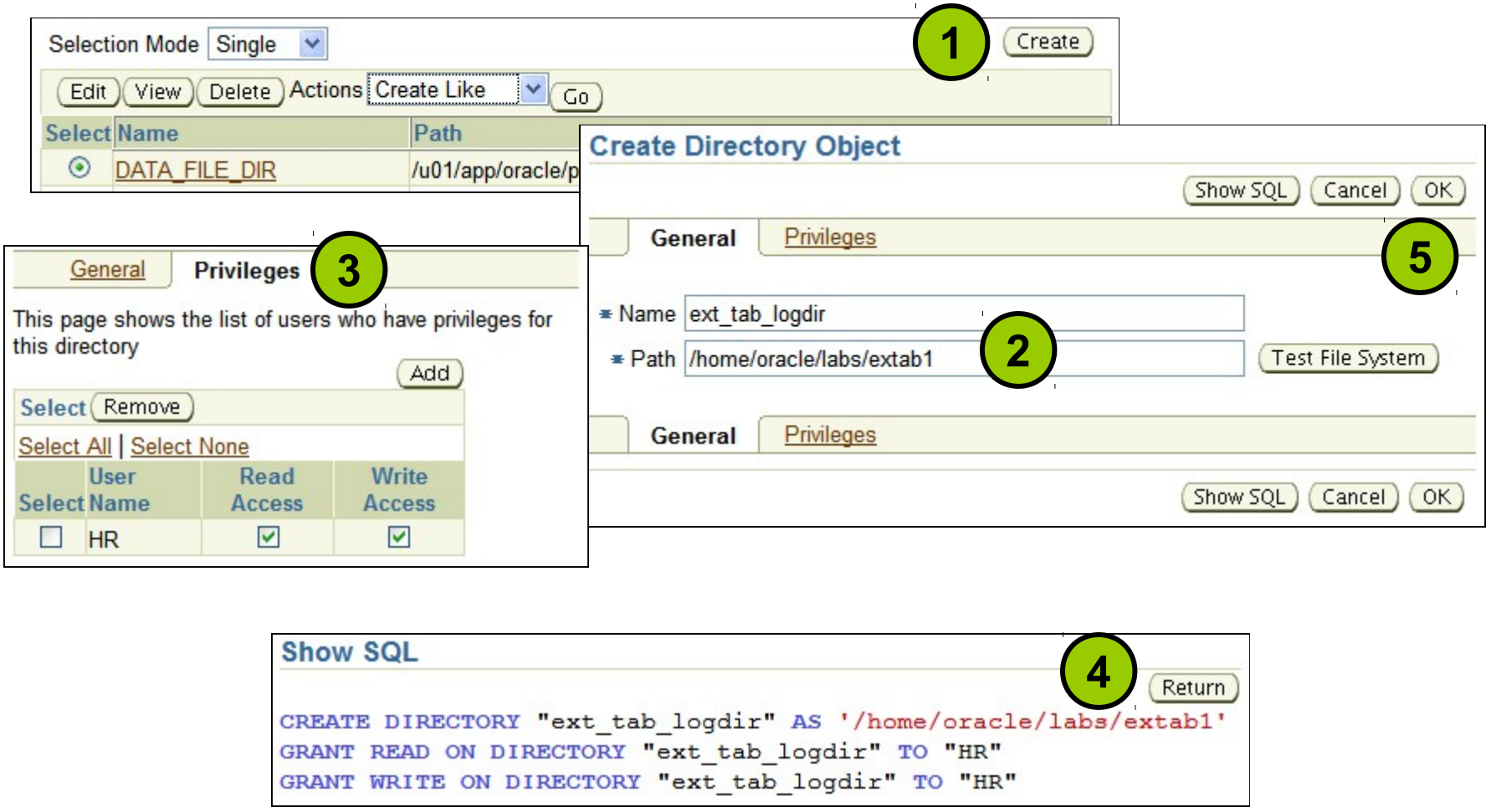
Data Pump offers many benefits and some new features over earlier data movement tools, such as:

- Fine-grained object and data selection
- Explicit specification of database version
- Parallel execution
- Estimation of export job space consumption
- Network mode in a distributed environment
- Remapping capabilities
- Data sampling and metadata compression
- Compression of data during a Data Pump export
- Security through encryption
- Ability to export XMLType data as CLOBs
- Legacy mode to support old import and export files

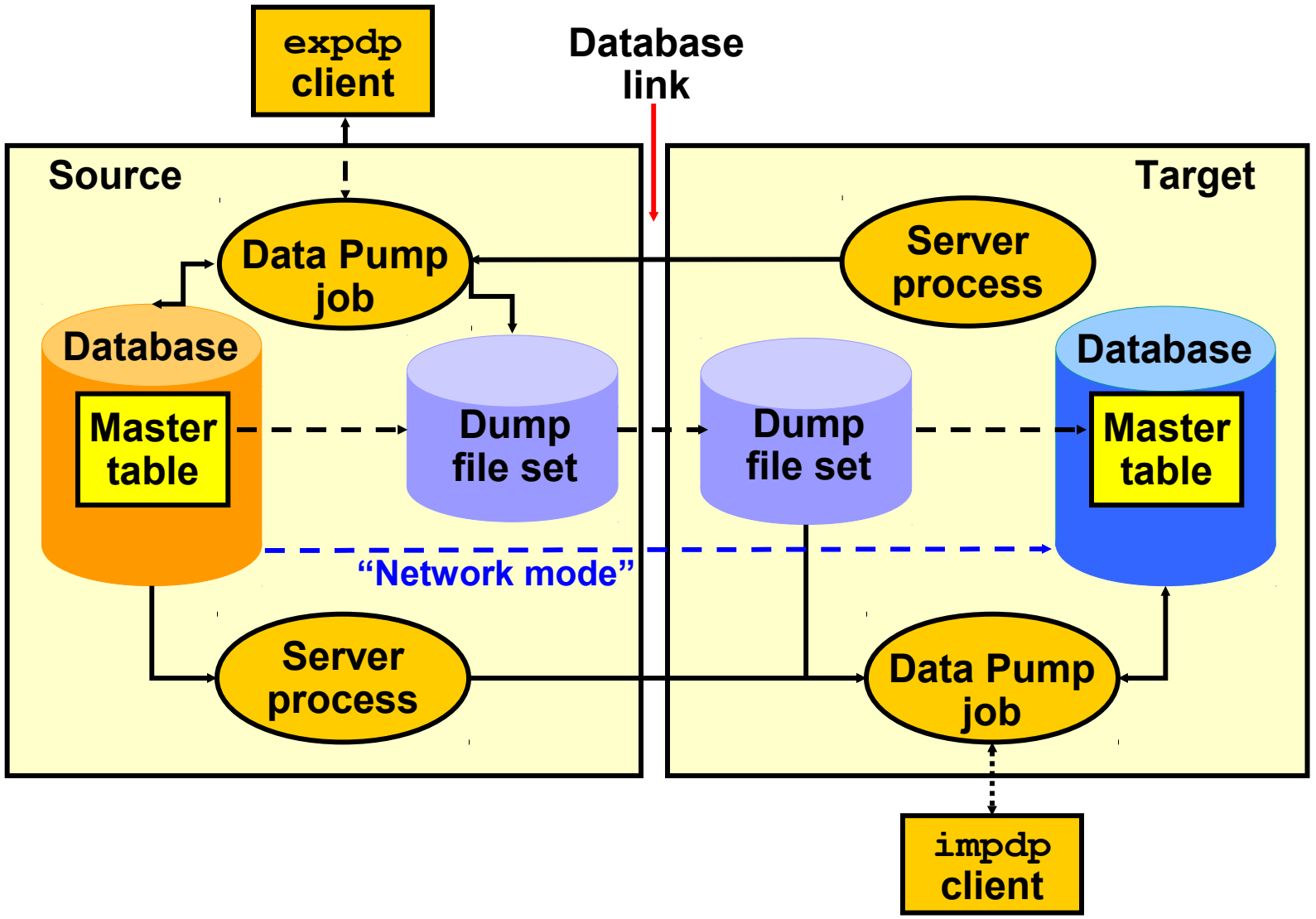
Directory Objects for Data Pump



Creating Directory Objects



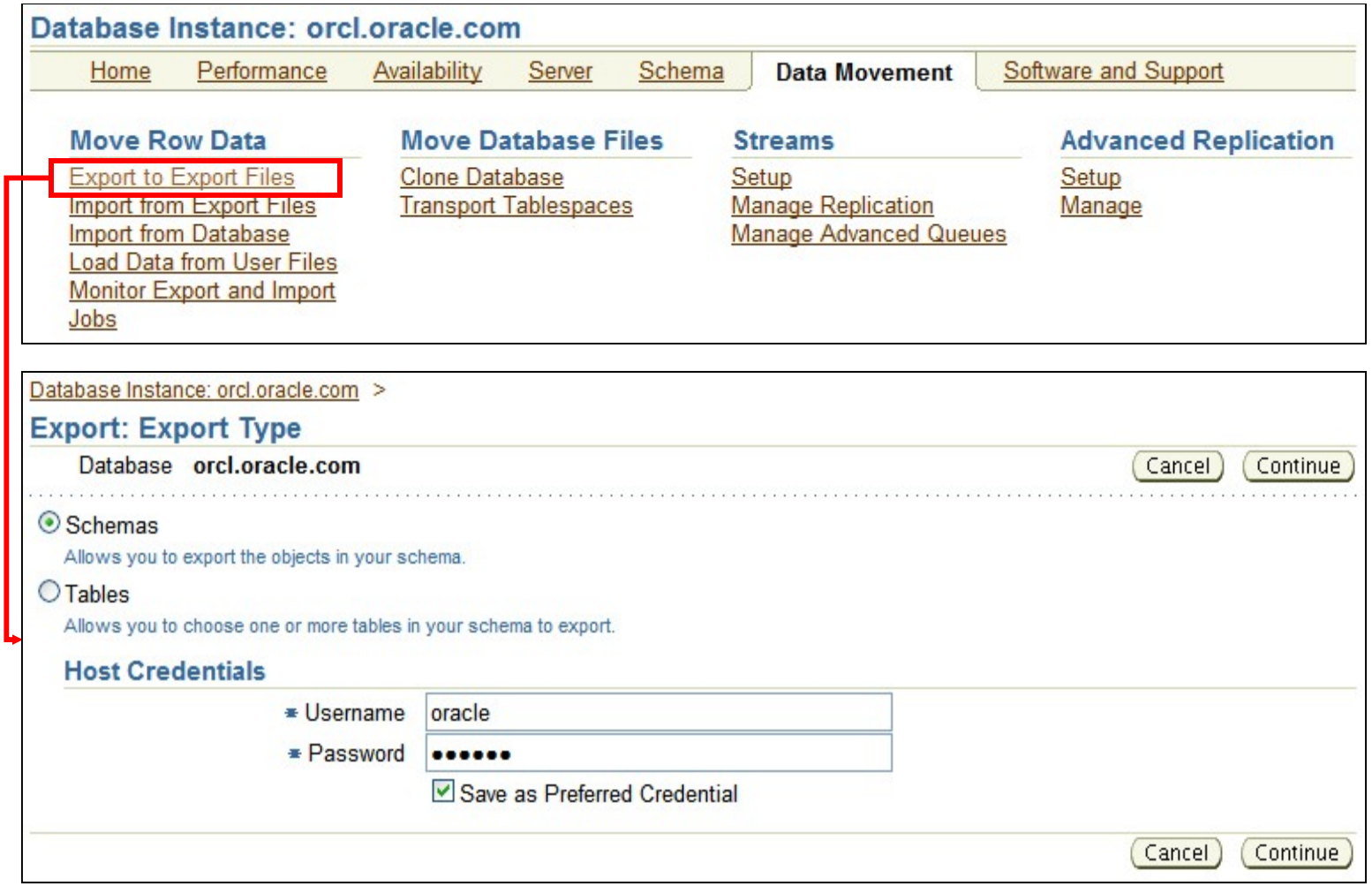
Data Pump Export and Import Clients: Overview



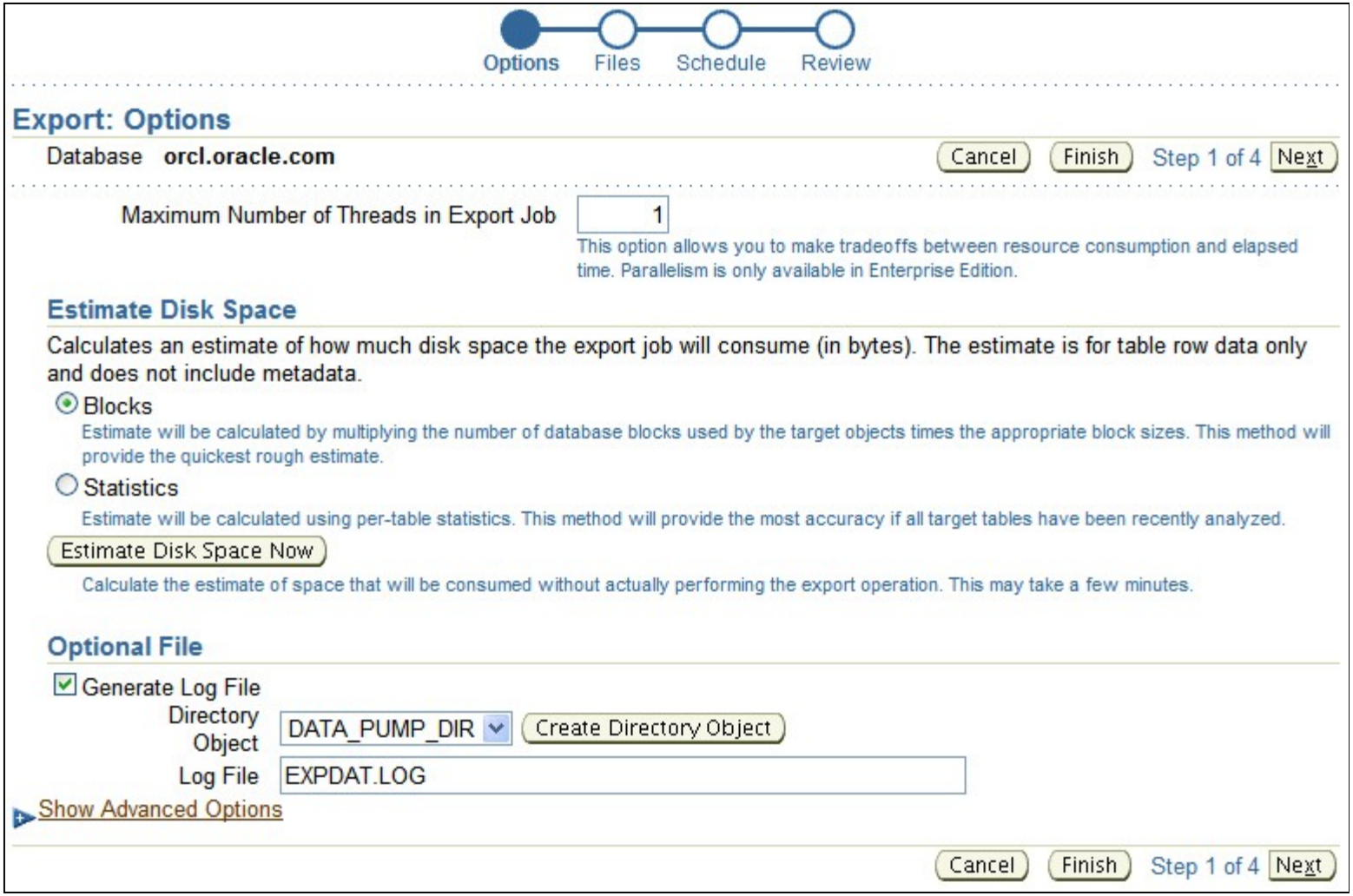
Data Pump Utility: Interfaces and Modes

- Data Pump Export and Import interfaces:
  - Command line
  - Parameter file
  - Interactive command line
  - Enterprise Manager
- Data Pump Export and Import modes:
  - Full
  - Schema
  - Table
  - Tablespace
  - Transportable tablespace

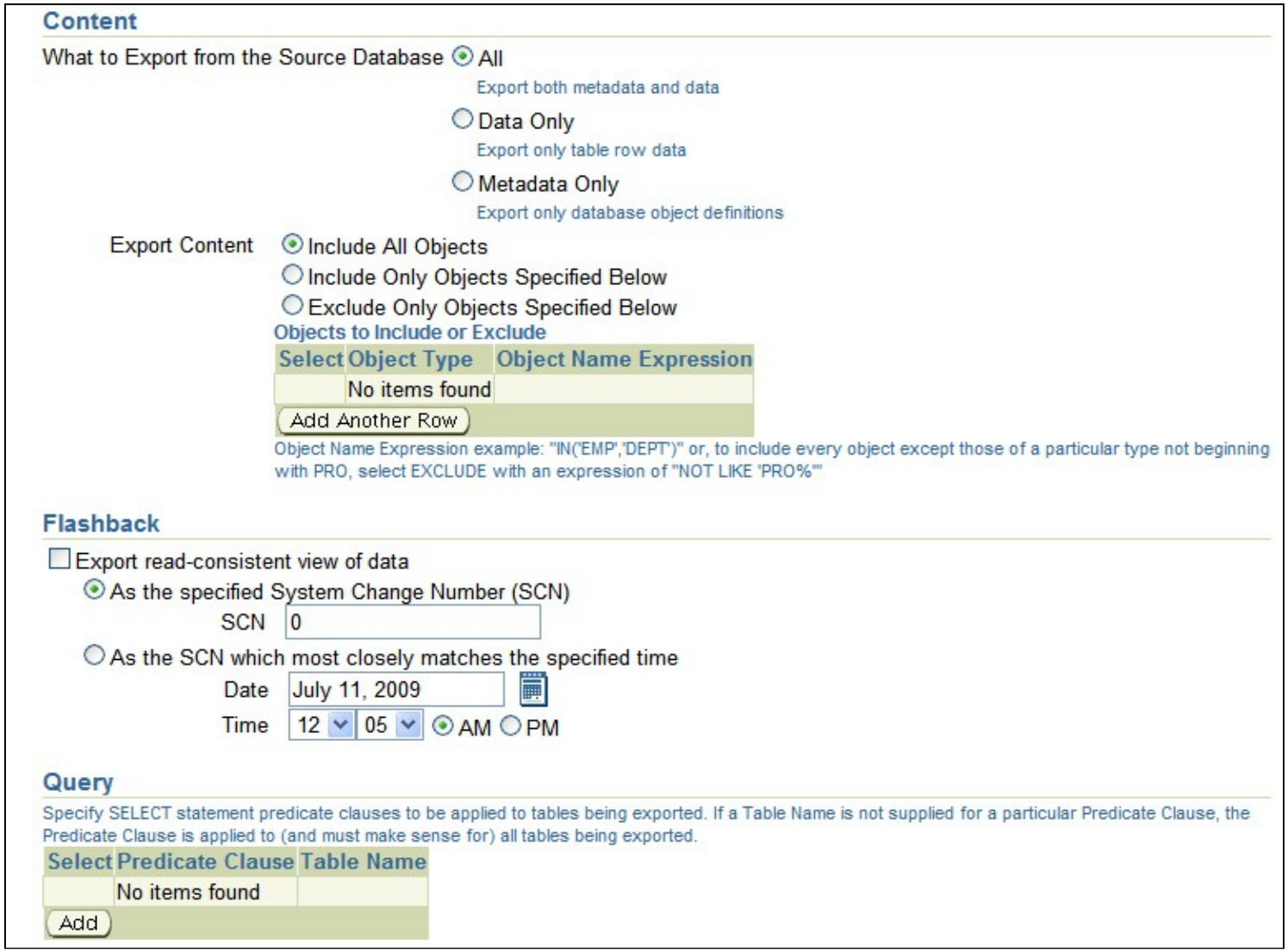
Data Pump Export using Database Control



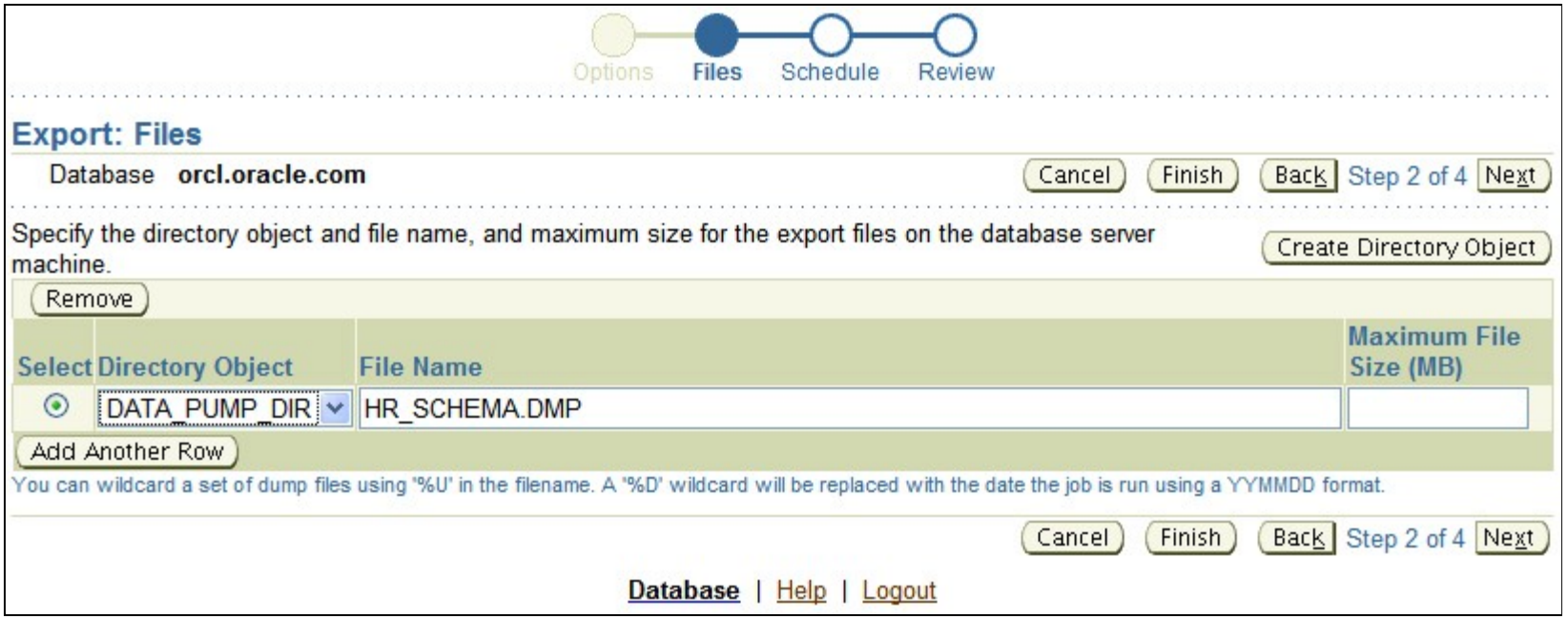
Data Pump Export Example: Basic Options



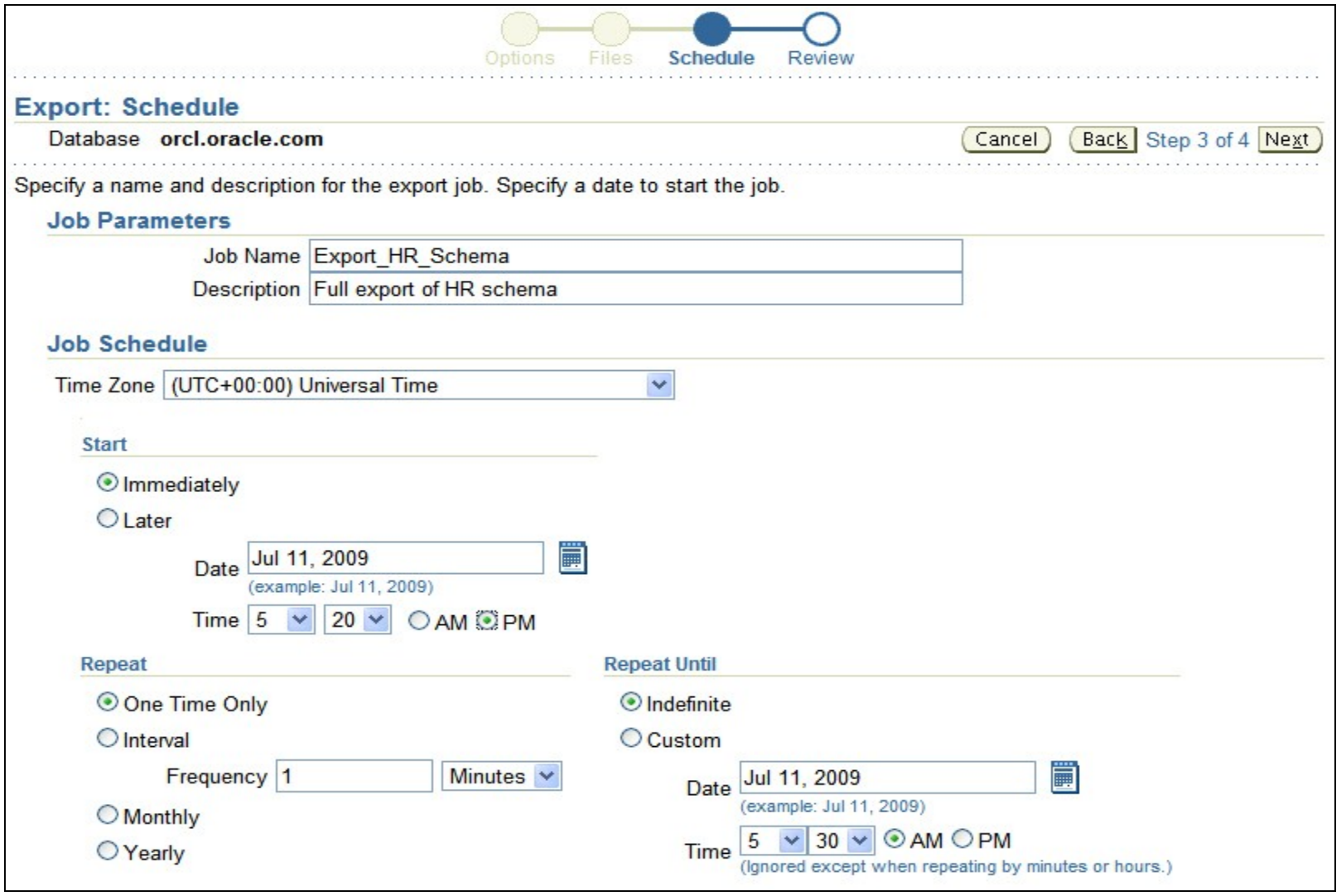
Data Pump Export Example: Advanced Options



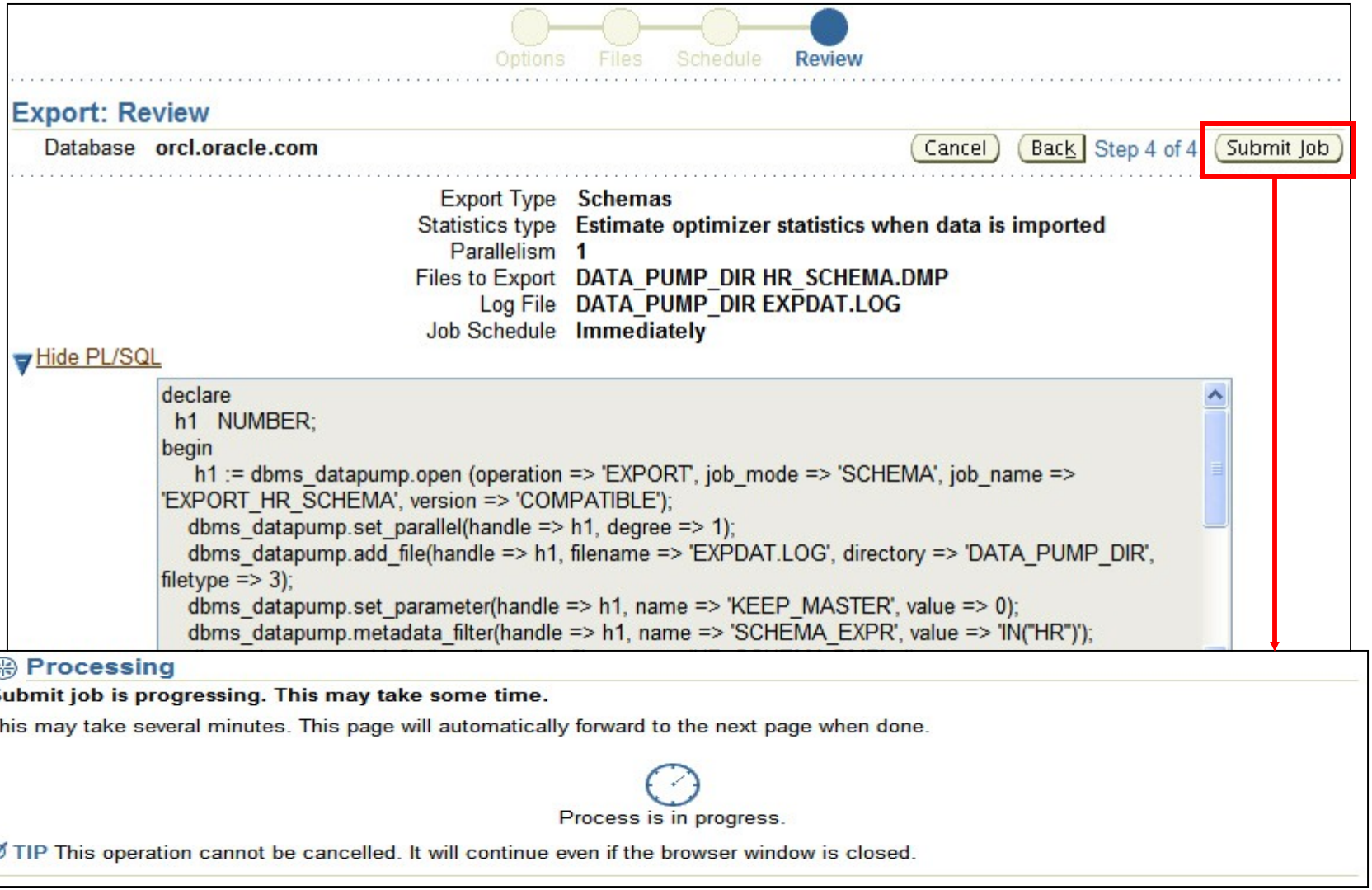
Data Pump Export Example: Files



Data Pump Export Example: Schedule



Data Pump Export Example: Review



Data Pump Import Example: `impdp`

Data Pump can be invoked on the command line to allow further command line options to be specified.

```
$ impdp hr DIRECTORY=DATA_PUMP_DIR \
DUMPFILE=HR_SCHEMA.DMP \
PARALLEL=1 \
CONTENT=ALL \
TABLES="EMPLOYEES" \
LOGFILE=DATA_PUMP_DIR:import_hr_employees.log \
JOB_NAME=importHR \
TRANSFORM=STORAGE:n
```

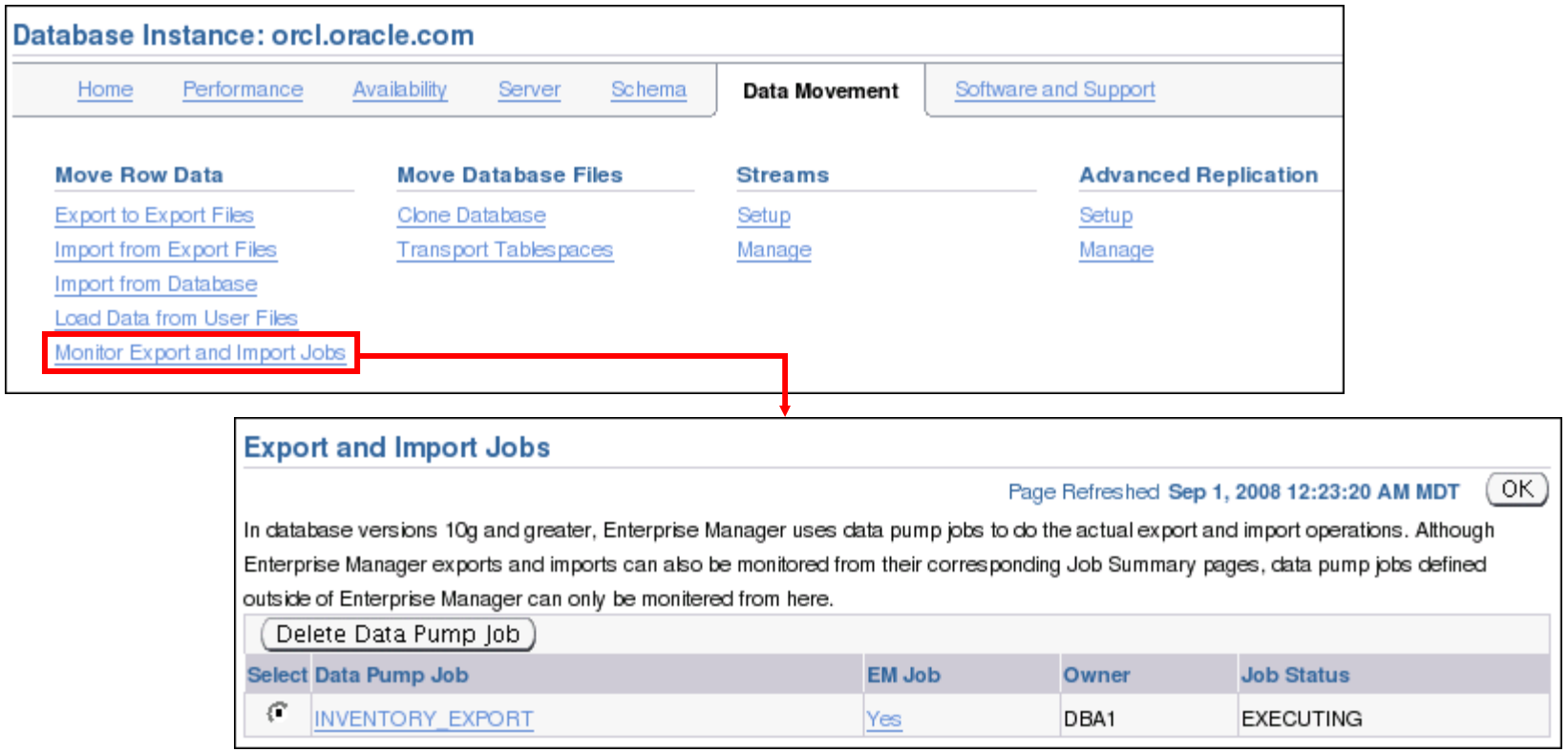
Data Pump Import: Transformations

You can remap:

- Data files by using `REMAP_DATAFILE`
- Tablespaces by using `REMAP_TABLESPACE`
- Schemas by using `REMAP_SCHEMA`
- Tables by using `REMAP_TABLE`
- Data by using `REMAP_DATA`

```
REMAP_TABLE = 'EMPLOYEES':'EMP'
```

Using Enterprise Manager to Monitor Data Pump Jobs



Migration with Data Pump Legacy Mode

- Assistance in transitioning from `imp` and `exp` utilities to `impdp` and `expdp` utilities
- Data Pump in legacy mode:
  1. Encounters unique `imp` or `exp` parameter and enters legacy mode
  2. Attempts to map the old syntax to the new syntax
  3. Displays new syntax
  4. Exits legacy mode

Best practice tip: Oracle strongly recommends that you view the new syntax and make script changes as time permits.

Data Pump Legacy Mode

The Data Pump export and import utilities:

- Read and write files only in Data Pump format
- Accept `exp` and `imp` utility commands in legacy mode
- Include legacy mode parameters that:
  - Can be identical to the new syntax:

    ```
    FILESIZE=integer[B | K | M | G]
    ```

  - Can be similar:

    ```
    QUERY= query_clause
    ```

  - Are ignored, when the command is superceded by Data Pump defaults

    ```
    BUFFER=integer
    COMPRESS={y|n}
    DIRECT={y|n}
    ```

  - Cause an error when old and new syntax is mixed

Data Pump Legacy Mode

- Legacy mode parameters:
  - Are mapped to Data Pump parameters, if possible:

```
consistent={y|n} -> FLASHBACK_TIME
GRANTS=n -> EXCLUDE=CONSTRAINTS
INDEXES=n -> EXCLUDE=INDEX
LOG=filename -> LOGFILE=filename
FILE=filename -> dumpfile=directory-object:filename
```

  - Can be similar, but not identical:

```
FEEDBACK=integer -> STATUS
```

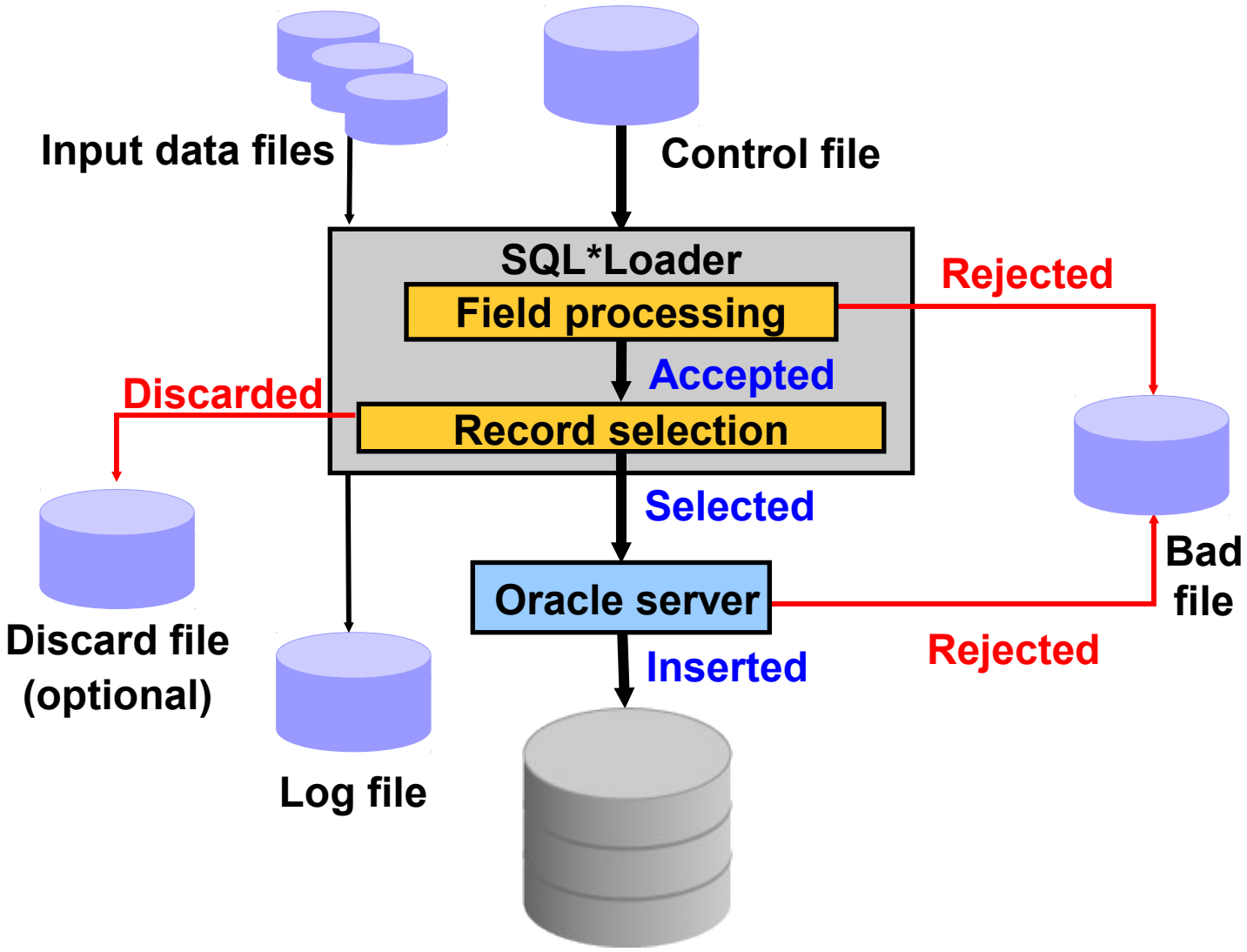
  - Cause an error when incompatible with new Data Pump:

```
VOLSIZE=integer
```

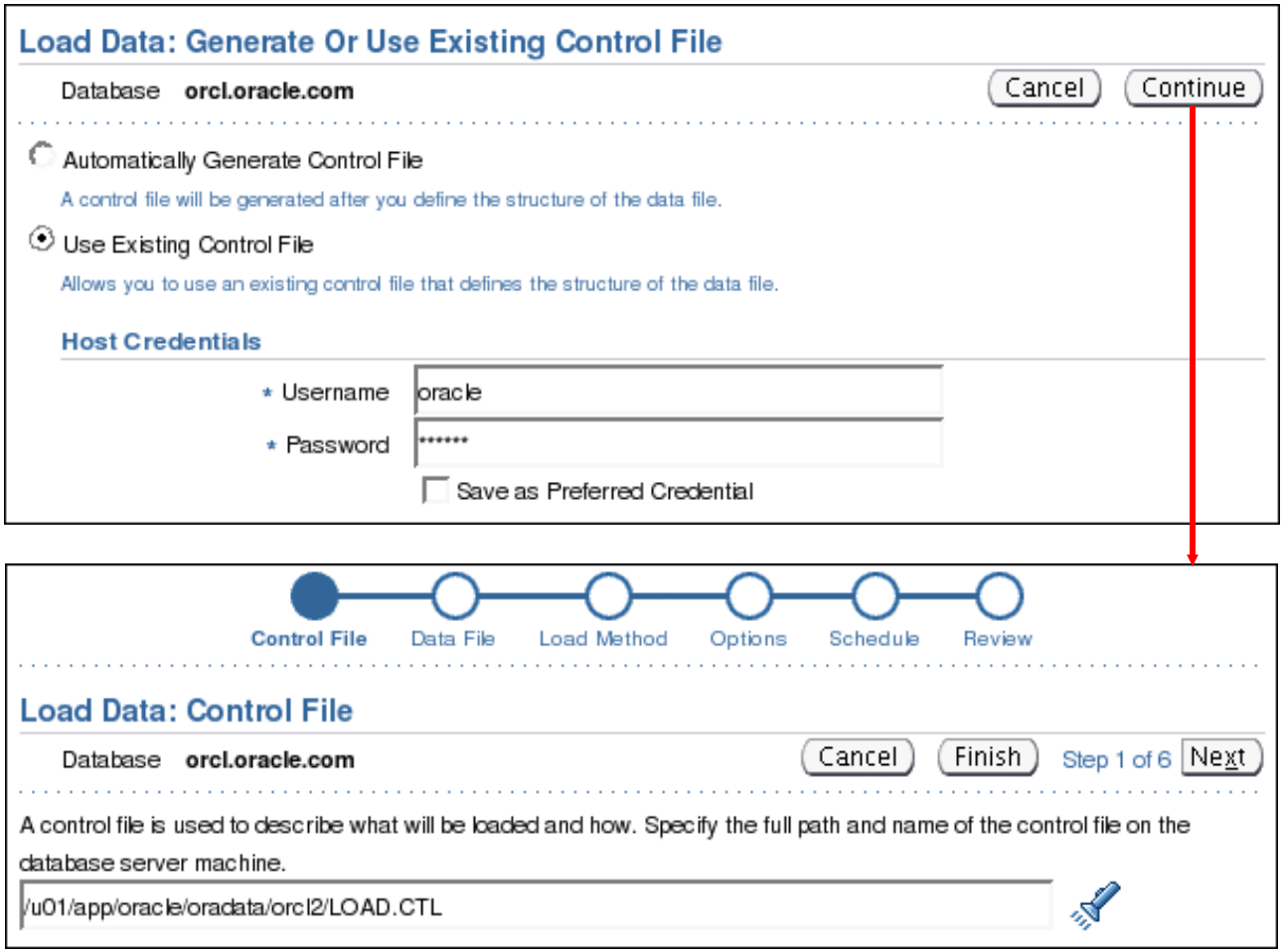
Managing File Locations

- Original `exp` and `imp` utilities: Fully qualified file names
- Data Pump directory object for file locations
  - Default (in prior versions): `DATA_PUMP_DIR` parameter
  - New optional `DATA_PUMP_DIR_`*`schema-name`* directory object
  - Managed with the `CREATE DIRECTORY` and `GRANT` SQL commands
  - Default location (independent of legacy mode), when:
    - Command line without `DIRECTORY` parameter
    - User without `EXP_FULL_DATABASE` privilege

SQL*Loader: Overview



Loading Data with SQL*Loader

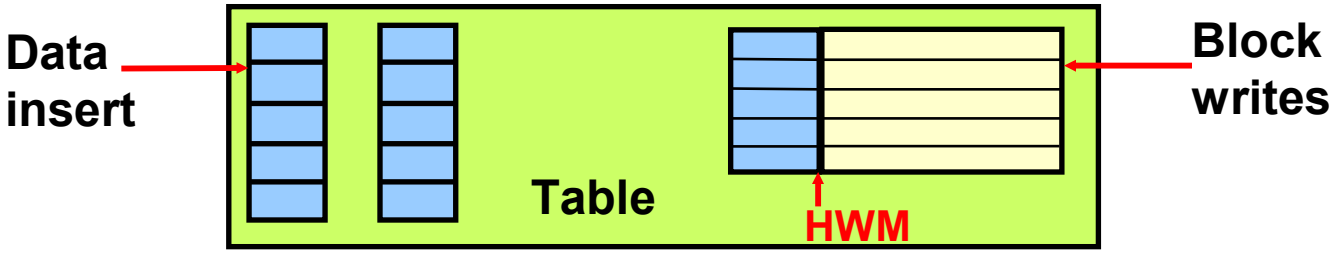


SQL*Loader Control File

The SQL*Loader control file instructs SQL*Loader about:

- Location of the data to be loaded
- Data format
- Configuration details:
  - Memory management
  - Record rejection
  - Interrupted load handling details
- Data manipulation details

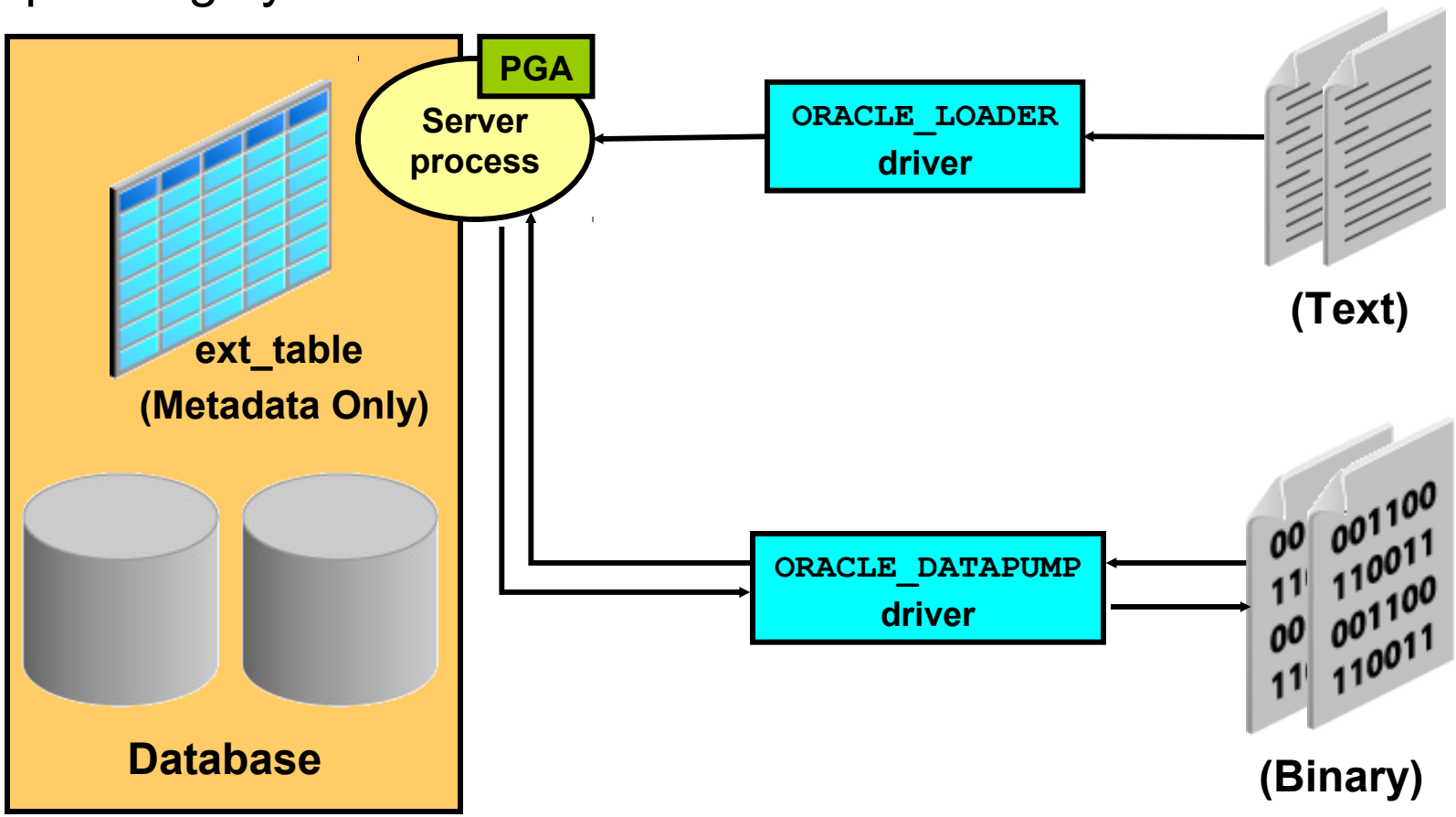
Loading Methods



Conventional Load	Direct Path Load
Uses `COMMIT`	Uses data saves (faster operation)
Always generates redo entries	Generates redo only under specific conditions
Enforces all constraints	Enforces only `PRIMARY KEY`, `UNIQUE`, and `NOT NULL`
Fires `INSERT` triggers	Does not fire `INSERT` triggers
Can load into clustered tables	Does not load into clusters
Allows other users to modify tables during load operation	Prevents other users from making changes to tables during load operation
Maintains index entries on each insert	Merges new index entries at the end of the load

External Tables

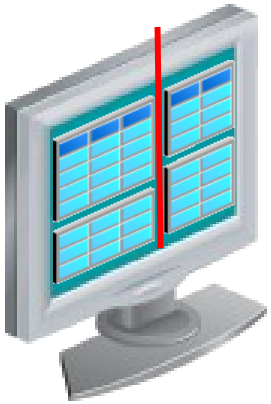
External tables are read-only tables stored as files on the operating system outside of the Oracle database.



External Table Benefits

- Data can be used directly from the external file or loaded into another database.
- External data can be queried and joined directly in parallel with tables residing in the database, without requiring it to be loaded first.
- The results of a complex query can be unloaded to an external file.
- You can combine generated files from different sources for loading purposes.

From Oracle Database



From external file

Defining an External Tables with ORACLE_LOADER

```
CREATE TABLE extab_employees
                (employee_id       NUMBER(4),
                 first_name        VARCHAR2(20),
                 last_name         VARCHAR2(25),
                 hire_date         DATE)
ORGANIZATION EXTERNAL
    ( TYPE ORACLE_LOADER DEFAULT DIRECTORY extab_dat_dir
      ACCESS PARAMETERS
      ( records delimited by newline
        badfile extab_bad_dir:'empxt%a_%p.bad'
        logfile extab_log_dir:'empxt%a_%p.log'
        fields terminated by ','
        missing field values are null
    ( employee_id, first_name, last_name,
     hire_date char date_format date mask "dd-mon-yyyy"))
    LOCATION ('empxt1.dat', 'empxt2.dat') )
    PARALLEL  REJECT LIMIT UNLIMITED;
```

External Table Population with ORACLE_DATAPUMP

```
CREATE TABLE ext_emp_query_results
  (first_name, last_name, department_name)
ORGANIZATION EXTERNAL
  (
    TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY ext_dir
    LOCATION ('emp1.exp','emp2.exp','emp3.exp')
  )
PARALLEL
AS
SELECT e.first_name,e.last_name,d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id AND
       d.department_name in
                    ('Marketing', 'Purchasing');
```

Using External Tables

- Querying and external table:

```
SQL> SELECT * FROM extab employees;
```

- Querying and joining an external table with an internal table

```
SQL> SELECT e.employee_id, e.first_name, e.last_name,
                          d.department_name FROM
departments d, extab_employees e
WHERE d.department id = e.department id;
```

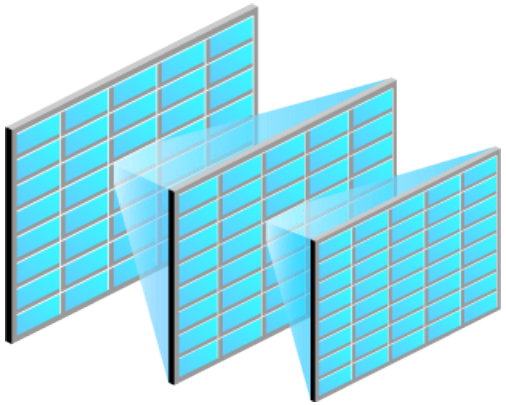
- Appending data to an internal table from an external table

```
SQL> INSERT /*+ APPEND */ INTO hr.employees SELECT * FROM
extab_employees;
```

Data Dictionary

View information about external tables in:

- [DBA| ALL| USER]_EXTERNAL_TABLES
- [DBA| ALL| USER]_EXTERNAL_LOCATIONS
- [DBA| ALL| USER]_TABLES
- [DBA| ALL| USER]_TAB_COLUMNS
- [DBA| ALL]_DIRECTORIES



Quiz

Like other database objects, Directory objects are owned by the user that creates them unless another schema is specified during creation.

1. True
2. False

Quiz

An index can be created on an external table.

1. True
2. False

Summary

In this lesson, you should have learned how to:

- Describe ways to move data
- Create and use directory objects
- Use SQL*Loader to load data from a non-Oracle database (or user files)
- Use external tables to move data via platform-independent files
- Explain the general architecture of Oracle Data Pump
- Use Data Pump Export and Import to move data between Oracle databases

Practice 17 Overview: Moving Data

This practice covers the following topics:

- Using the Data Pump Export Wizard to select database objects to be exported
- Monitoring a Data Pump Export job
- Using the Data Pump Import Wizard to import tables to your database
- Using the Load Data Wizard to load data into your database
- Loading data by using the command line