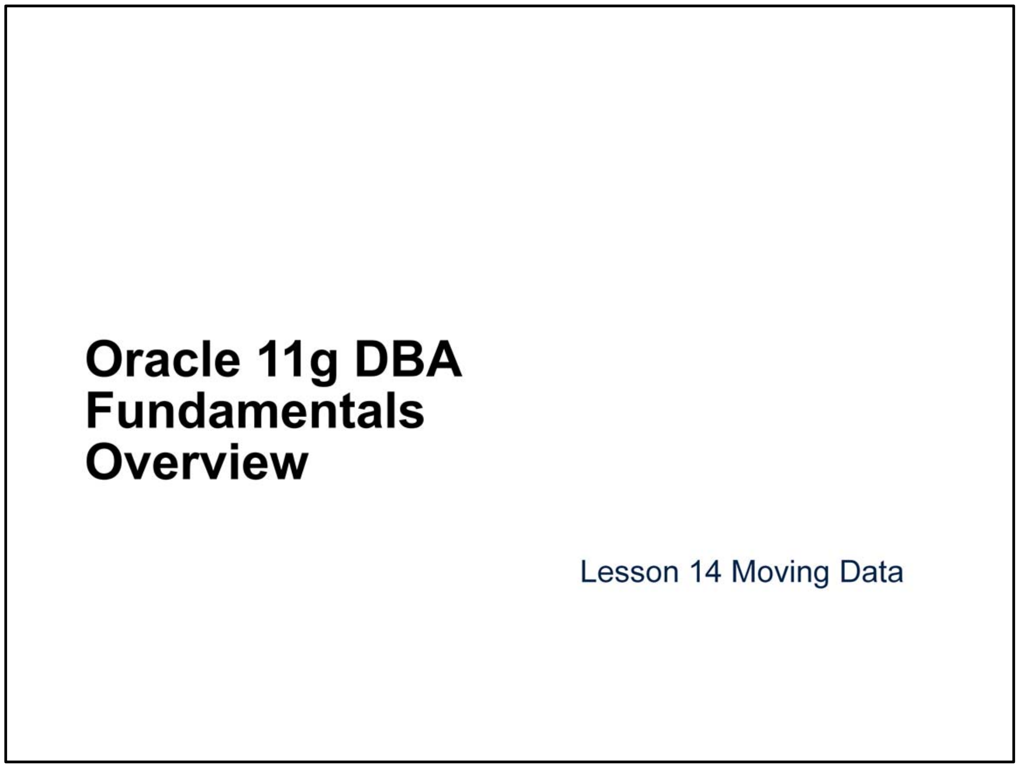
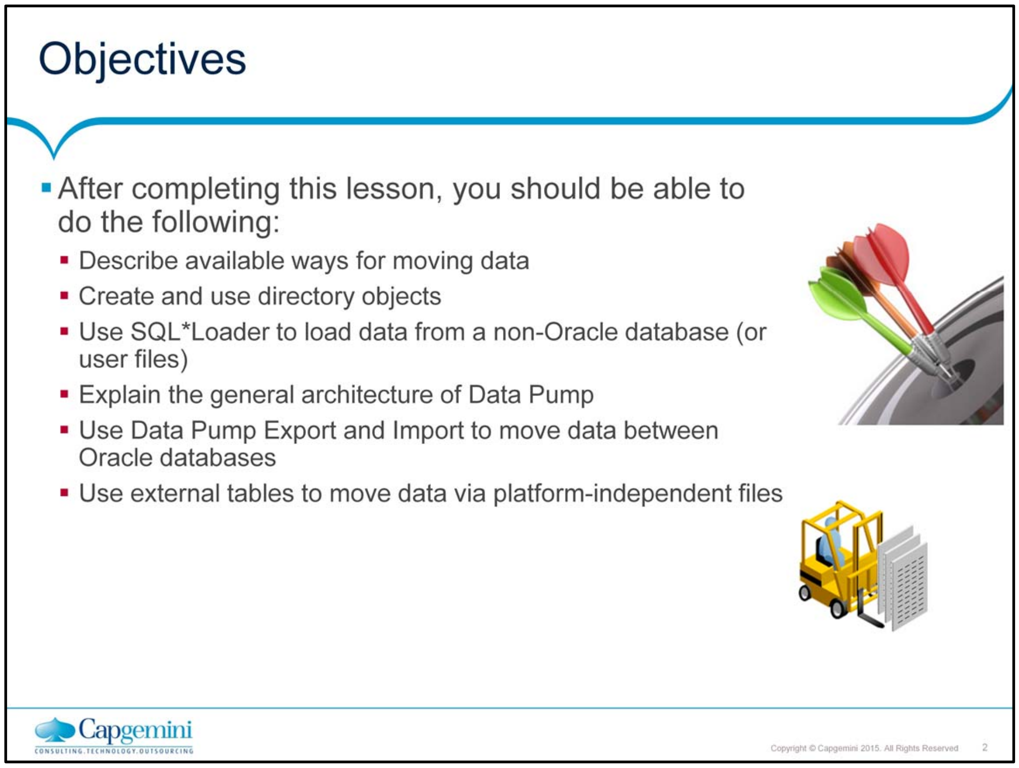


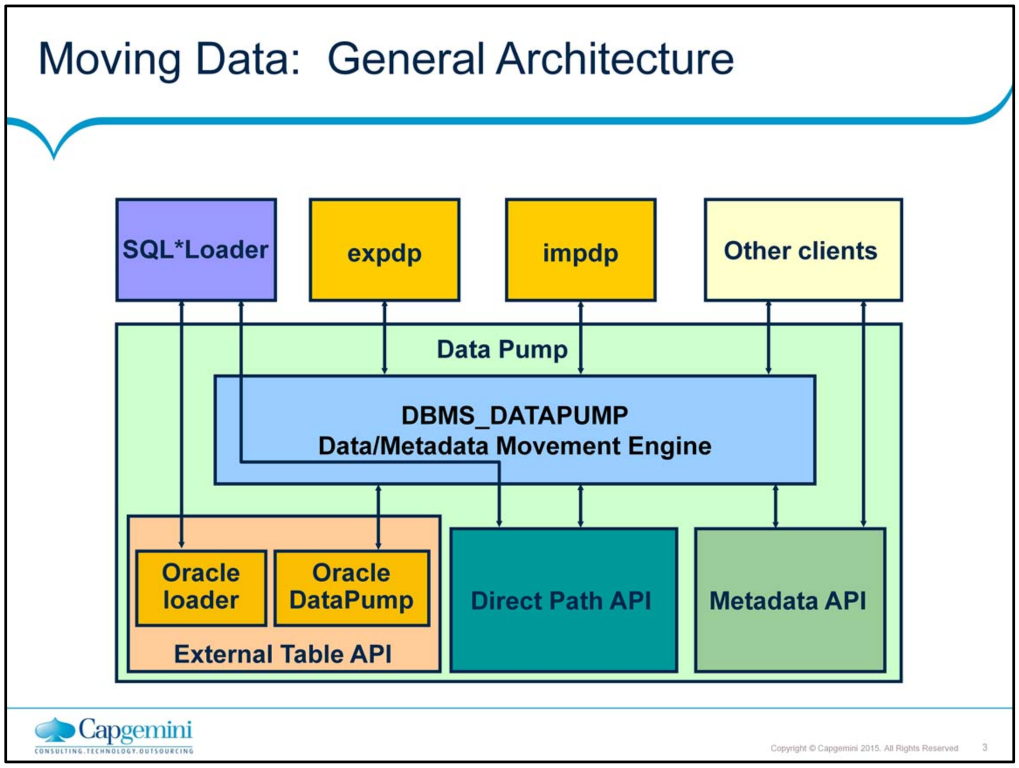
Oracle 11g DBA Fundamentals Overview

Lesson 14 Moving Data



Objectives

- After completing this lesson, you should be able to do the following:
  - Describe available ways for moving data
  - Create and use directory objects
  - Use SQL*Loader to load data from a non-Oracle database (or user files)
  - Explain the general architecture of Data Pump
  - Use Data Pump Export and Import to move data between Oracle databases
  - Use external tables to move data via platform-independent files



Moving Data: General Architecture

Moving Data: General Architecture

This is a block diagram of the major functional components:

DBMS_DATAPUMP: This package embodies the API for high-speed export and import utilities for bulk data and metadata movement.

Direct Path API (DPAPI): Oracle Database 11g supports a direct path API interface that minimizes data conversion and parsing at both unload and load time.

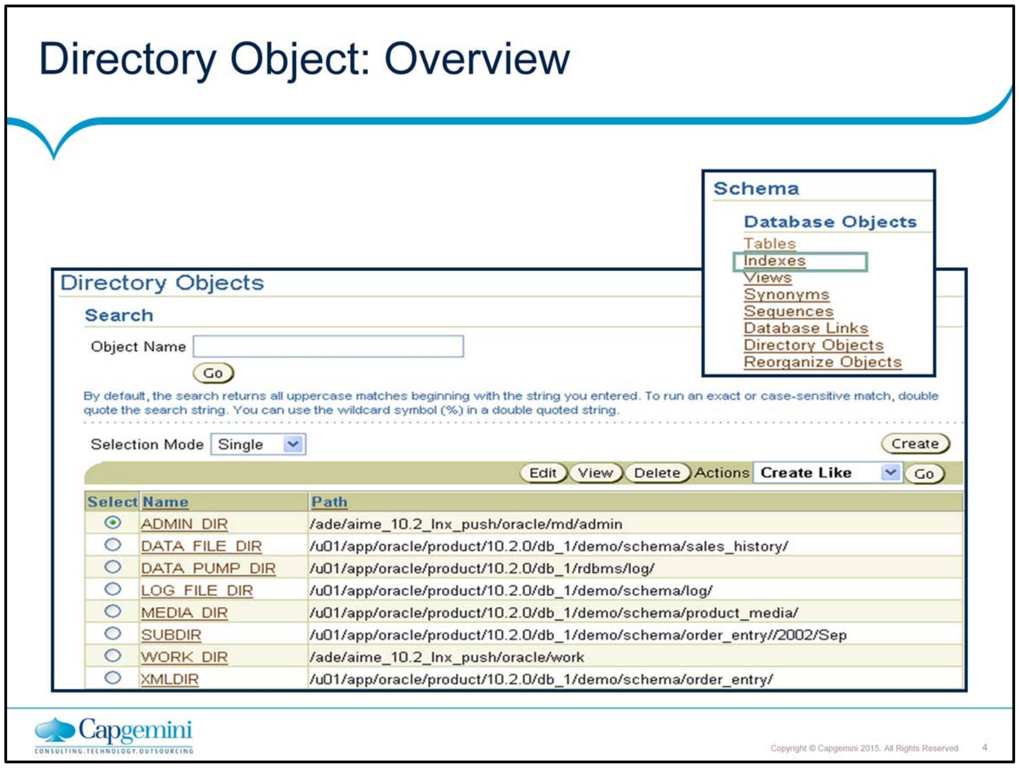
DBMS_METADATA: This package is used by worker processes for all metadata unloading and loading. Database object definitions are stored using XML rather than SQL.

External Table API: With the ORACLE_DATAPUMP and ORACLE_LOADER access drivers, you can store data in external tables (that is, in platform-independent files). The SELECT statement reads external tables as though they were stored in an Oracle database.

SQL*Loader: The SQL*Loader client has been integrated with external tables, thereby providing automatic migration of loader control files to external table access parameters.

expdp and impdp: The expdp and impdp clients are thin layers that make calls to the DBMS_DATAPUMP package to initiate and monitor Data Pump operations.

Other clients: They are applications, such as Database Control, replication, transportable tablespaces, and user applications, that benefit from this infrastructure. SQL*Plus may also be used as a client of DBMS_DATAPUMP for simple status queries against ongoing operations.



Directory Object: Overview

Directory Objects

Search

Object Name [ ]

Go

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Selection Mode: Single

Select	Name	Path
●	ADMIN_DIR	/ade/aime_10.2_lnx_push/oracle/md/admin
○	DATA_FILE_DIR	/u01/app/oracle/product/10.2.0/db_1/demo/schema/sales_history/
○	DATA_PUMP_DIR	/u01/app/oracle/product/10.2.0/db_1/rdbms/log/
○	LOG_FILE_DIR	/u01/app/oracle/product/10.2.0/db_1/demo/schema/log/
○	MEDIA_DIR	/u01/app/oracle/product/10.2.0/db_1/demo/schema/product_media/
○	SUBDIR	/u01/app/oracle/product/10.2.0/db_1/demo/schema/order_entry//2002/Sep
○	WORK_DIR	/ade/aime_10.2_lnx_push/oracle/work
○	XMLDIR	/u01/app/oracle/product/10.2.0/db_1/demo/schema/order_entry/

Directory Object: Overview

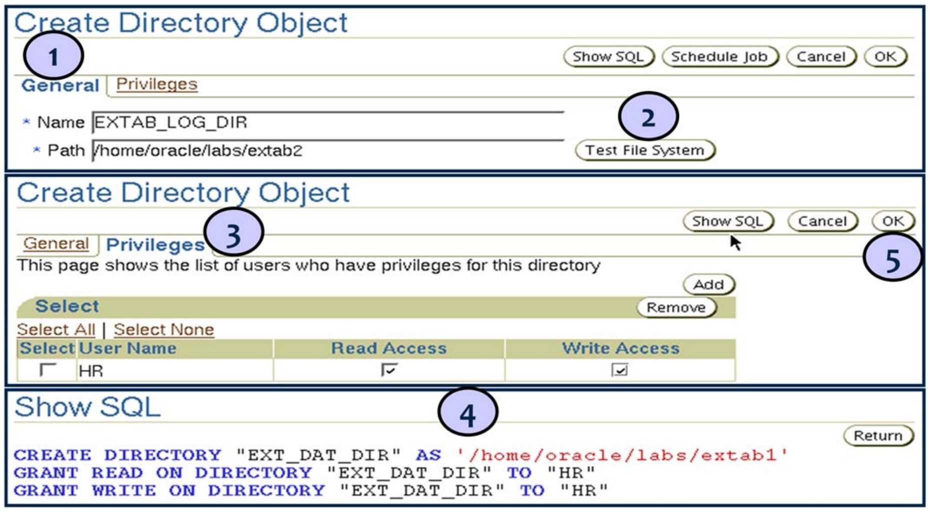
Directory objects are logical structures that represent a physical directory on the server’s file system. They contain the location of a specific operating system directory. This directory object name can be used in Enterprise Manager, so you do not need to hard-code directory path specifications. Therefore, you get greater file management flexibility. Directory objects are owned by the SYS user. Directory names are unique across the database because all the directories are located in a single name space (that is, SYS).

Directory objects are required when you specify file locations for Data Pump because it accesses files on the server rather than on the client.

In Enterprise Manager, select Administration > Directory Objects.

To edit or delete a directory object, select the directory object and click the appropriate button.

Creating Directory Objects



* Name EXTAB_LOG_DIR

* Path /home/oracle/labs/extab2

This page shows the list of users who have privileges for this directory

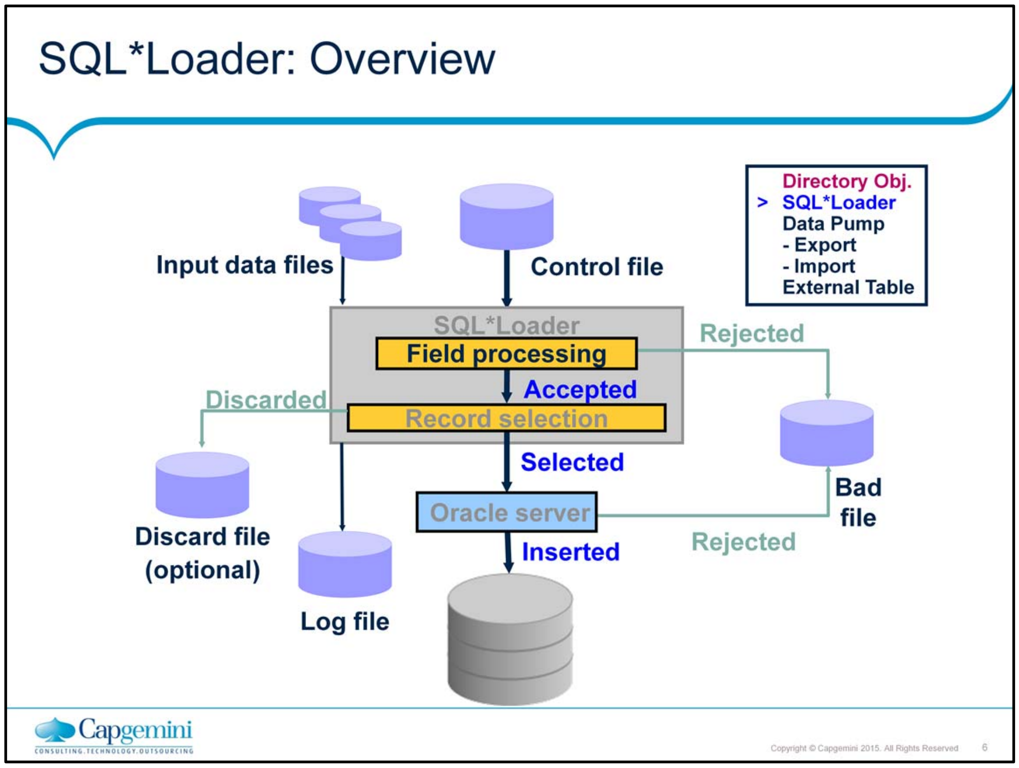
Select	User Name	Read Access	Write Access
☐	HR	✓	✓

```
CREATE DIRECTORY "EXT_DAT_DIR" AS '/home/oracle/labs/extab1'
GRANT READ ON DIRECTORY "EXT_DAT_DIR" TO "HR"
GRANT WRITE ON DIRECTORY "EXT_DAT_DIR" TO "HR"
```

Creating Directory Objects

On the Directory Objects page, click the Create button.

2. Enter the name of the directory object and the OS path to which it maps. OS directories should be created before they are used. You can test this by clicking the “Test File System” button. For the test, provide the host login credentials (that is, the OS user who has privileges on this OS directory).
3. Permissions for the directory objects are not the same as the OS permissions on the physical directory on the server file system. You can manage user privileges on individual directory objects. This increases the level of security and gives you granular control over these objects. On the Privileges tabbed page, click Add to select the user to which you give read or write privileges or both.
4. Click Show SQL to view the underlying statements.
5. Click OK to create the object.



SQL*Loader: Overview

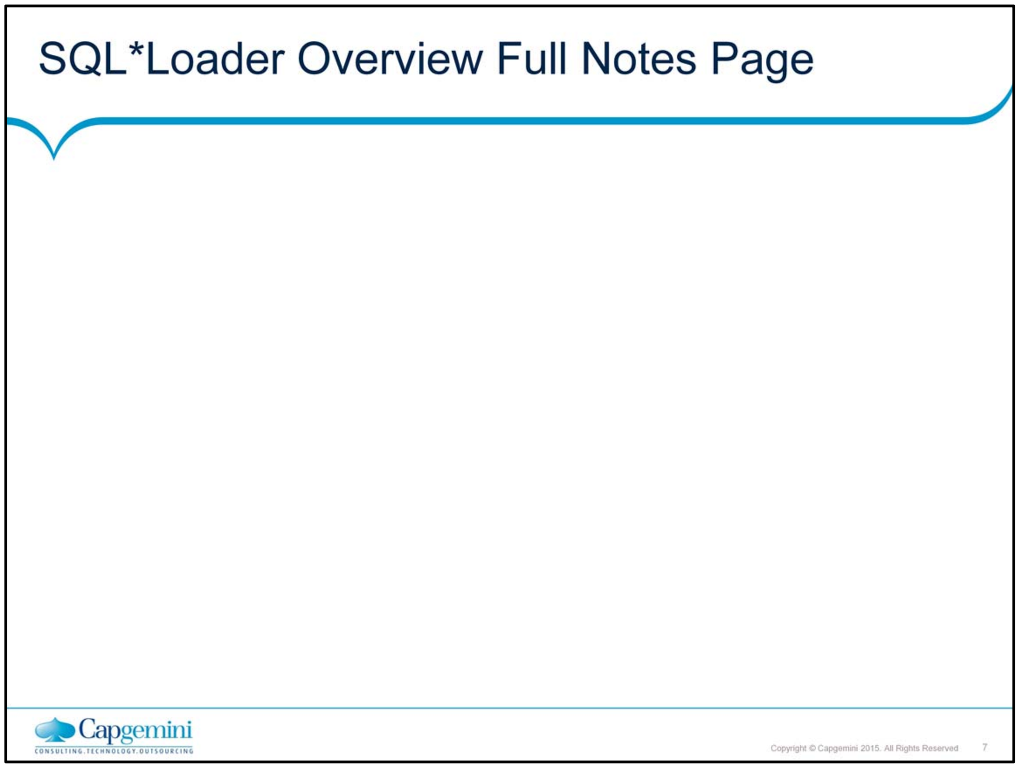
SQL*Loader: Overview

SQL*Loader loads data from external files into tables of an Oracle database. It has a powerful data parsing engine that puts little limitation on the format of the data in the data file. The files that are used by SQL*Loader are as follows:

Input data files: SQL*Loader reads data from one or more files (or operating system–equivalents of files) that are specified in the control file. From SQL*Loader's perspective, the data in the data file is organized as records. A particular data file can be in fixed record format, variable record format, or stream record format. The record format can be specified in the control file with the INFILE parameter. If no record format is specified, the default is stream record format.

Control file: The control file is a text file that is written in a language that SQL*Loader understands. The control file indicates to SQL*Loader where to find the data, how to parse and interpret the data, where to insert the data, and so on. Although not precisely defined, a control file can be said to have three sections.

- The first section contains sessionwide information, for example:
  - Global options, such as the input data file name, and records to be skipped.
  - INFILE clauses to specify where the input data is located
  - Data to be loaded



SQL*Loader Overview Full Notes Page

SQL*Loader: Overview (continued)

The second section consists of one or more INTO TABLE blocks. Each of these blocks contains information about the table (such as the table name and the columns of the table) into which the data is to be loaded.

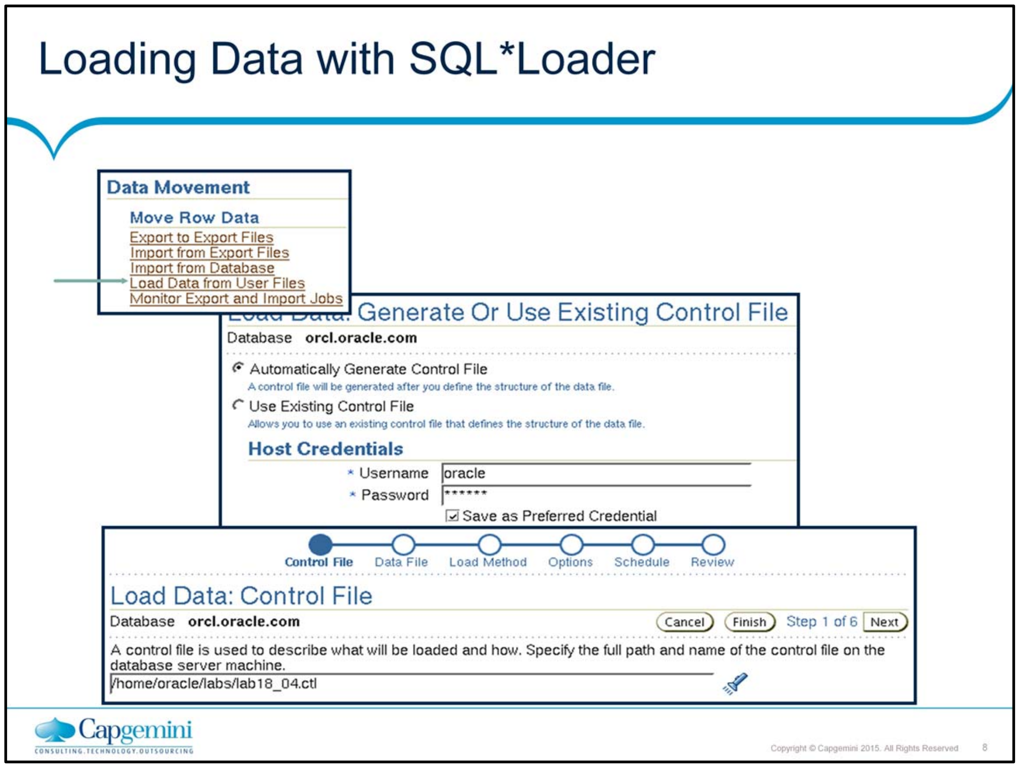
The third section is optional and, if present, contains input data.

Log file: When SQL*Loader begins execution, it creates a log file. If it cannot create a log file, execution terminates. The log file contains a detailed summary of the load, including a description of any errors that occurred during the load.

Bad file: The bad file contains records that are rejected, either by SQL*Loader or by the Oracle database. Data file records are rejected by SQL*Loader when the input format is invalid. After a data file record is accepted for processing by SQL*Loader, it is sent to the Oracle database for insertion into a table as a row. If the Oracle database determines that the row is valid, then the row is inserted into the table. If the row is determined to be invalid, then the record is rejected and SQL*Loader puts it in the bad file.

Discard file: This file is created only when it is needed, and only if you have specified that a discard file should be enabled. The discard file contains records that are filtered out of the load because they do not match any record-selection criteria specified in the control file.

For more information about SQL*Loader, refer to the Oracle Database Utilities documentation.



Loading Data with SQL*Loader

Load Data: Generate Or Use Existing Control File

Database **orcl.oracle.com**

(● Automatically Generate Control File)
A control file will be generated after you define the structure of the data file.

○ Use Existing Control File
Allows you to use an existing control file that defines the structure of the data file.

Host Credentials

* Username	oracle
* Password	******

☑ Save as Preferred Credential

Load Data: Control File

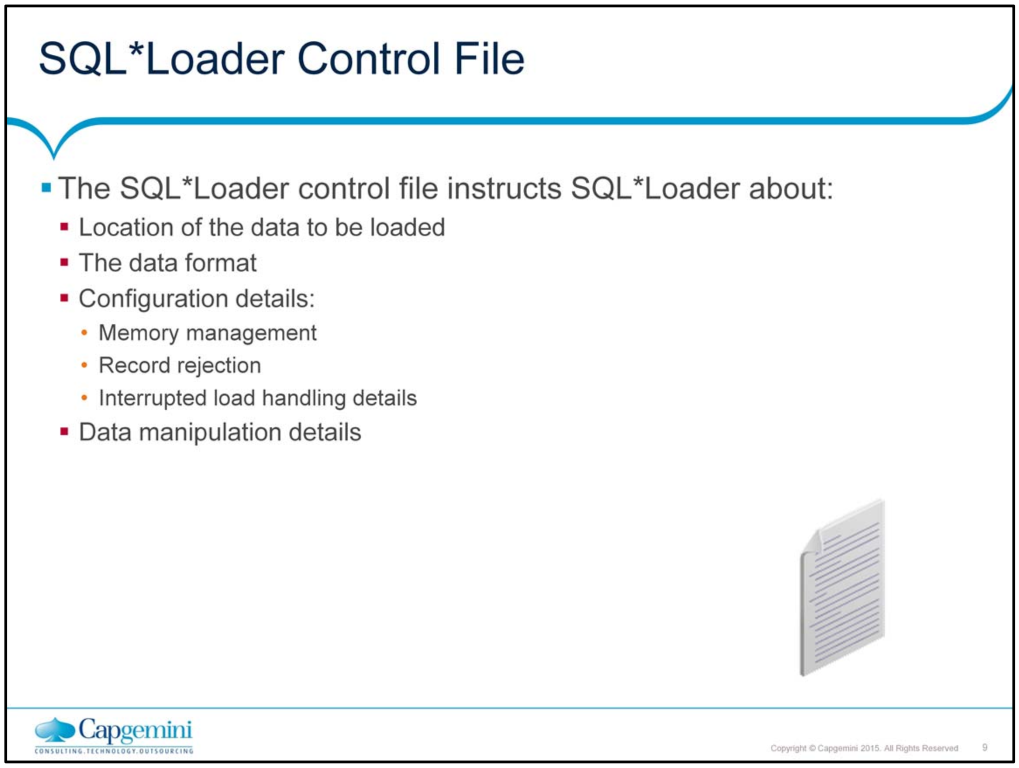
Database **orcl.oracle.com**

A control file is used to describe what will be loaded and how. Specify the full path and name of the control file on the database server machine.

/home/oracle/labs/lab18_04.ctl

Loading Data with SQL*Loader

Use the Load Data from User Files Wizard to load data from a flat file into an Oracle database. To display the wizard, select Enterprise Manager Maintenance > Data Movement > Move Row Data > Load Data from User Files. The wizard guides you through the required steps.



SQL*Loader Control File

- The SQL*Loader control file instructs SQL*Loader about:
  - Location of the data to be loaded
  - The data format
  - Configuration details:
    - Memory management
    - Record rejection
    - Interrupted load handling details
  - Data manipulation details

SQL*Loader Control File

The SQL*Loader control file is a text file that contains data definition language (DDL) instructions. DDL is used to control the following aspects of a SQL*Loader session:

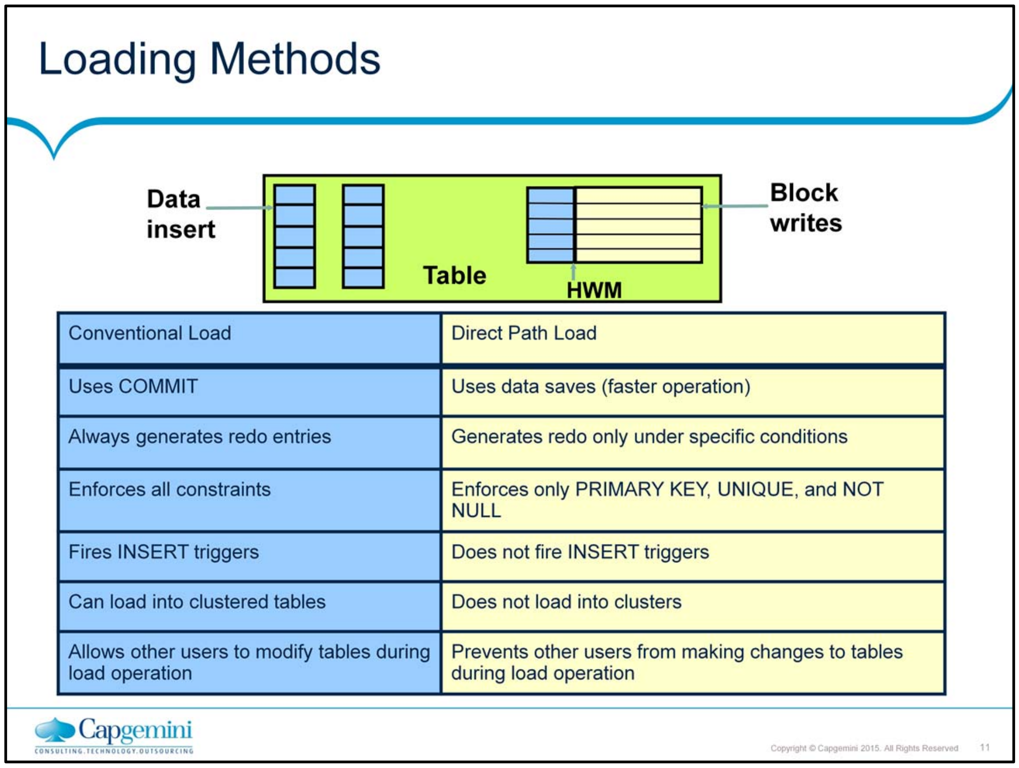
Where SQL*Loader finds the data to load
How SQL*Loader expects that data to be formatted
How SQL*Loader is being configured (including memory management, selection and rejection criteria, interrupted load handling, and so on) as it loads the data
How SQL*Loader manipulates the data being loaded

SQL*Loader Control File (continued)

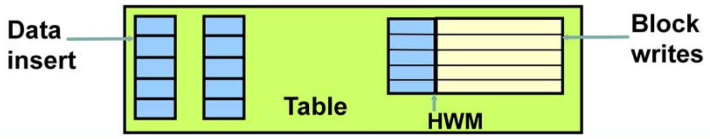
```
1     -- This is a sample control file
2 LOAD DATA
3 INFILE 'SAMPLE.DAT'
4 BADFILE 'sample.bad'
5 DISCARDFILE 'sample.dsc'
6 APPEND
7 INTO TABLE emp
8 WHEN (57) = '.'
9 TRAILING NULLCOLS
10 (hiredate SYSDATE,
              deptno POSITION(1:2) INTEGER EXTERNAL(3)
        NULLIF deptno=BLANKS,
              job POSITION(7:14) CHAR TERMINATED BY WHITESPACE
              NULLIF job=BLANKS "UPPER(:job)",
              mgr POSITION(28:31) INTEGER EXTERNAL
              TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
        ename POSITION(34:41) CHAR
              TERMINATED BY WHITESPACE "UPPER(:ename)",
              empno POSITION(45) INTEGER EXTERNAL
              TERMINATED BY WHITESPACE,
              sal POSITION(51) CHAR TERMINATED BY WHITESPACE
              "TO_NUMBER(:sal,'$99,999.99')",
              comm INTEGER EXTERNAL ENCLOSED BY '(' AND '%'
              ":comm * 100"
      )
```

The explanation of a sample control file by line numbers is as follows:

1. Comments can appear anywhere in the command section of the file, but they must not appear within the data. Precede any comment with two hyphens. All text to the right of the double hyphen is ignored, until the end of the line.
2. The LOAD DATA statement indicates to SQL*Loader that this is the beginning of a new data load. If you are continuing a load that has been interrupted in progress, use the CONTINUE LOAD DATA statement.
3. The INFILE keyword specifies the name of a data file containing data that you want to load.
4. The BADFILE keyword specifies the name of a file into which rejected records are placed.
5. The DISCARDFILE keyword specifies the name of a file into which discarded records are placed.
6. The APPEND keyword is one of the options that you can use when loading data into a table that is not empty. To load data into a table that is empty, use the INSERT keyword.
7. The INTO TABLE keyword enables you to identify tables, fields, and data types. It defines the relationship between records in the data file and tables in the database.
8. The WHEN clause specifies one or more field conditions that each record must match before SQL*Loader loads the data. In this example, SQL*Loader loads the record only if the 57th character is a decimal point. That decimal point delimits dollars and cents in the field and causes records to be rejected if SAL has no value.
9. The TRAILING NULLCOLS clause prompts SQL*Loader to treat any relatively positioned columns that are not present in the record as null columns.
10. The remainder of the control file contains the field list, which provides information about column formats in the table that is being loaded.



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

Comparing Direct and Conventional Path Loads

Method of Saving Data

Conventional path loads use SQL processing and a database COMMIT operation for saving data. The insertion of an array of records is followed by a COMMIT operation. Each data load may involve several transactions. Direct path loads use data saves to write blocks of data to Oracle data files. This is why the direct path loads are faster than the conventional ones. The following features differentiate a data save from COMMIT:

- During a data save, only full database blocks are written to the database.
- The blocks are written after the high-water mark (HWM) of the table.
- After a data save, the high-water mark (HWM) is moved.
- Internal resources are not released after a data save.
- A data save does not end the transaction.
- Indexes are not updated at each data save.

Note: Direct path and parallel direct path loads are so similar regarding DML activities that they are not separated in this comparison.

Comparing Direct and Conventional Path Loads (continued)

Logging Changes

Conventional path loading generates redo entries similar to any DML statement. When using a direct path load, redo entries are not generated if:

- The database is in NOARCHIVELOG mode
- The database is in ARCHIVELOG mode, but logging is disabled. Logging can be disabled by setting the NOLOGGING attribute for the table or by using the UNRECOVERABLE clause in the control file.

Enforcing Constraints

During a conventional path load, all enabled constraints are enforced in the same way that they are during any DML operation.

During direct path loads, the constraints are handled as follows:

- NOT NULL constraints are checked when arrays are built.
- FOREIGN KEY and CHECK constraints are disabled, and they can be enabled at the end of the load by using the appropriate commands in the control file. The FOREIGN KEY constraints are disabled because they reference other rows or tables, and the CHECK constraints are disabled because they may use SQL functions. If only a small number of rows are to be inserted into a large table, then use conventional loads.
- PRIMARY KEY and UNIQUE constraints are checked during and at the end of the load, and they may be disabled if they are violated.

Firing the INSERT Triggers

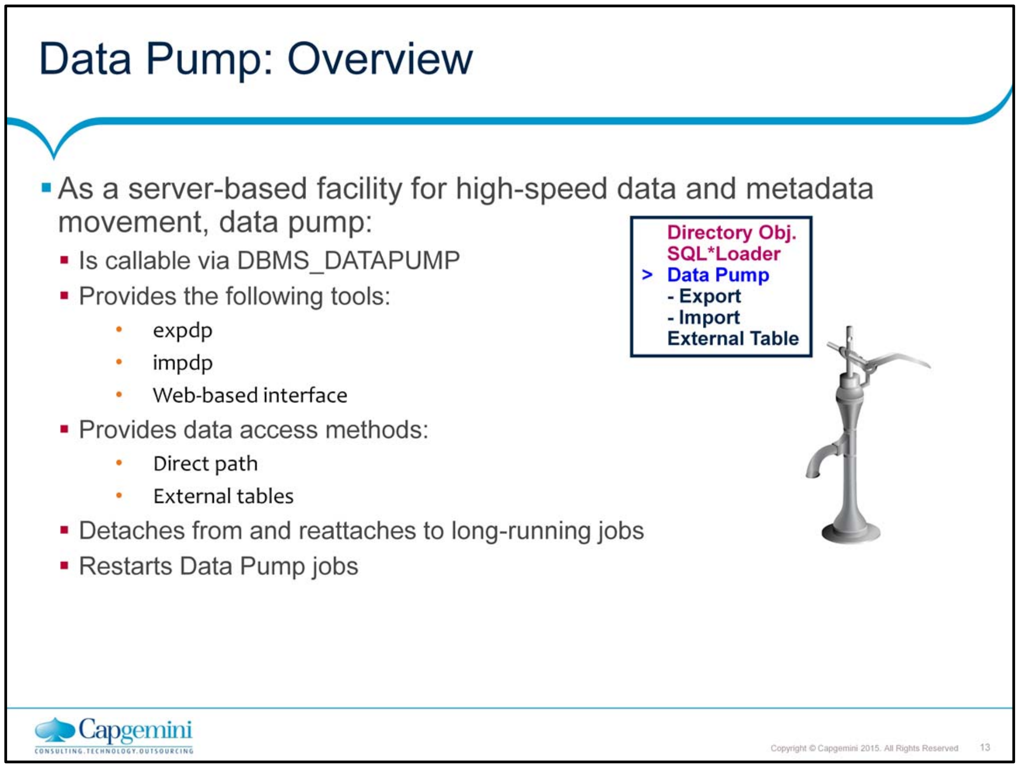
The WHILE INSERT triggers are fired during conventional loads; they are disabled before a direct path load and reenabled at the end of the load. They may remain disabled if a referenced object is not accessible at the end of the run. Consider using conventional path loads to load data into tables with the INSERT triggers.

Loading into Clustered Tables

Direct Loads cannot be used to load rows into clustered tables. Clustered tables can be loaded with conventional path loads only.

Locking

While a direct path load is in progress, other transactions cannot make changes to the tables that are being loaded. The only exception to this rule is when several parallel direct load sessions are used concurrently.



Data Pump: Overview

- As a server-based facility for high-speed data and metadata movement, data pump:
  - Is callable via DBMS_DATAPUMP
  - Provides the following tools:
    - expdp
    - impdp
    - Web-based interface
  - Provides data access methods:
    - Direct path
    - External tables
  - Detaches from and reattaches to long-running jobs
  - Restarts Data Pump jobs

Directory Obj.
SQL*Loader
> Data Pump
- Export
- Import
External Table

Data Pump: Overview

Data Pump enables very high-speed data and metadata loading and unloading of Oracle databases. The Data Pump infrastructure is callable via the DBMS_DATAPUMP PL/SQL package. Thus, custom data movement utilities can be built by using Data Pump.

Oracle Database 11g provides the following tools:

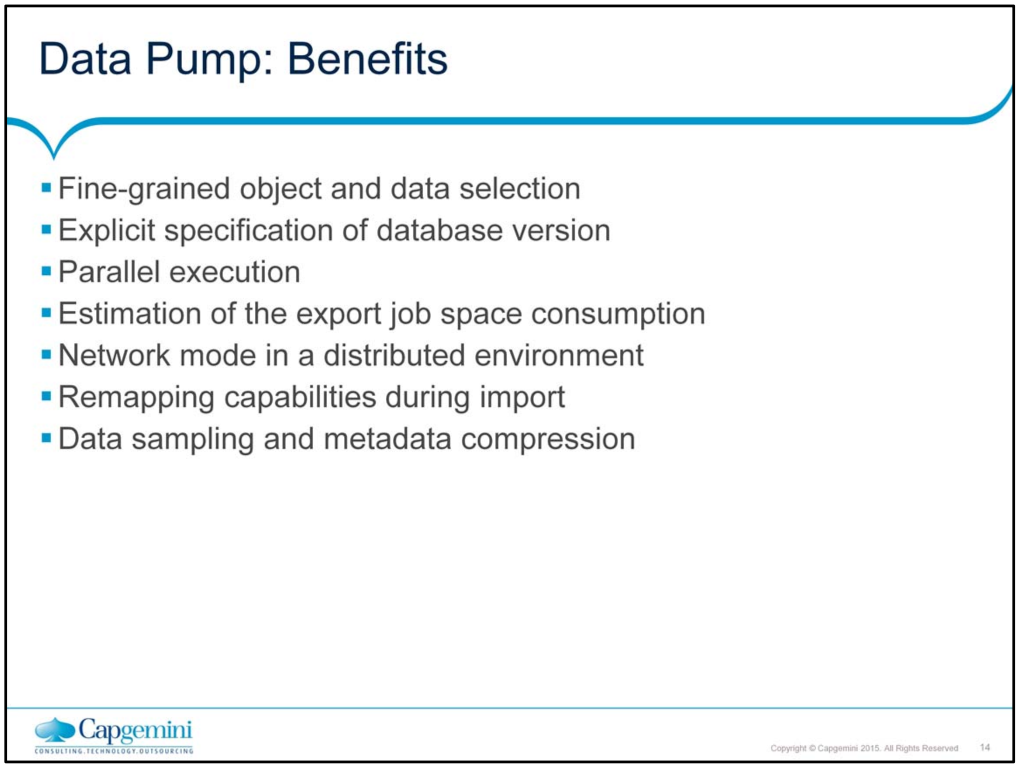
Command-line export and import clients called expdp and impdp respectively

A Web-based export and import interface that is accessible from Database Control

Data Pump automatically decides the data access methods to use; these can be either direct path or external tables. Data Pump uses direct path load and unload when a table’s structure allows it and when maximum single-stream performance is desired. However, if there are clustered tables, referential integrity constraints, encrypted columns, or a number of other items, Data Pump uses external tables rather than direct path to move the data.

The ability to detach from and reattach to long-running jobs without affecting the job itself enables you to monitor jobs from multiple locations while they are running. All stopped Data Pump jobs can be restarted without loss of data as long as the metainformation remains undisturbed. It does not matter whether the job is stopped voluntarily or involuntarily due to a crash.

Note: Data Pump is an integral feature of Oracle Database 11g and is, therefore, available in all configurations. However, parallelism is available in Enterprise Edition only.



Data Pump: Benefits

- Fine-grained object and data selection
- Explicit specification of database version
- Parallel execution
- Estimation of the export job space consumption
- Network mode in a distributed environment
- Remapping capabilities during import
- Data sampling and metadata compression

Data Pump: Benefits

The EXCLUDE, INCLUDE, and CONTENT parameters are used for fine-grained object and data selection.

You can specify the database version for objects to be moved (using the VERSION parameter) to create a dump file set that is compatible with a previous release of the Oracle database that supports Data Pump.

You can use the PARALLEL parameter to specify the maximum number of threads of active execution servers operating on behalf of the export job.

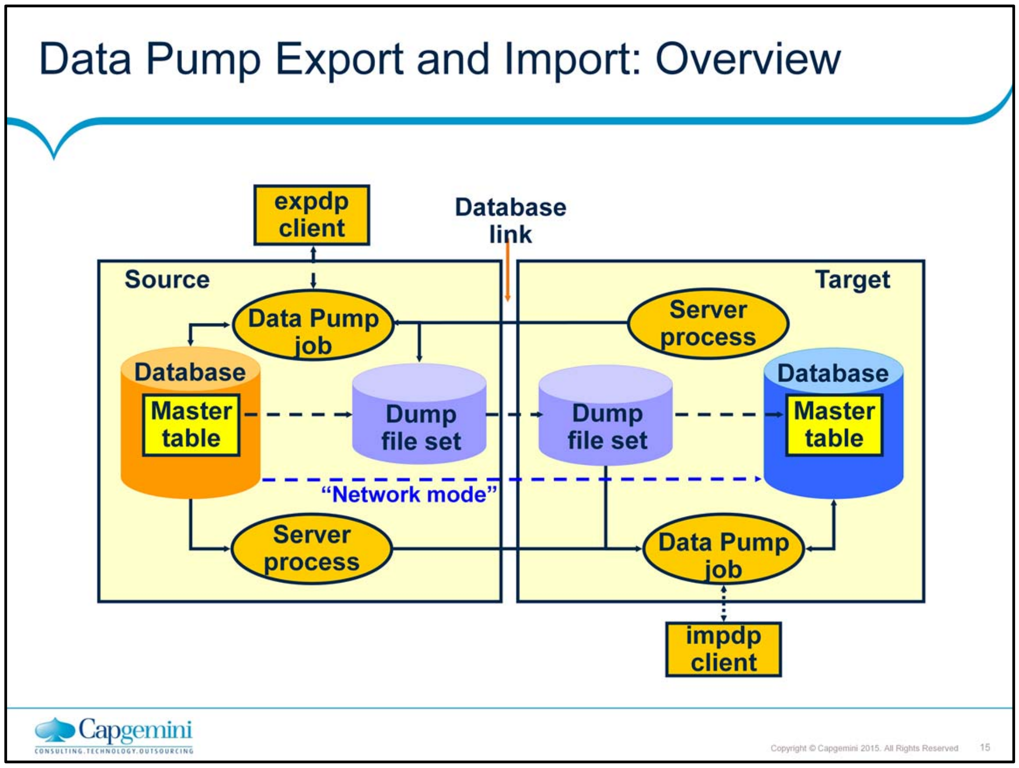
You can estimate how much space an export job would consume (without actually performing the export) by using the ESTIMATE_ONLY parameter.

Network mode enables you to export from a remote database directly to a dump file set. This can be done by using a database link to the source system.

During import, you can change the target data file names, schemas, and tablespaces.

In addition, Oracle Database 11g enables you to specify a percentage of data to be sampled and unloaded from the source database when performing a Data Pump export. This can be done by specifying the SAMPLE parameter.

You can use the COMPRESSION parameter to indicate whether the metadata should be compressed in the export dump file so that it consumes less disk space. If you compress the metadata, it is automatically uncompressed during import.



Data Pump Export and Import: Overview

Data Pump Export and Import: Overview

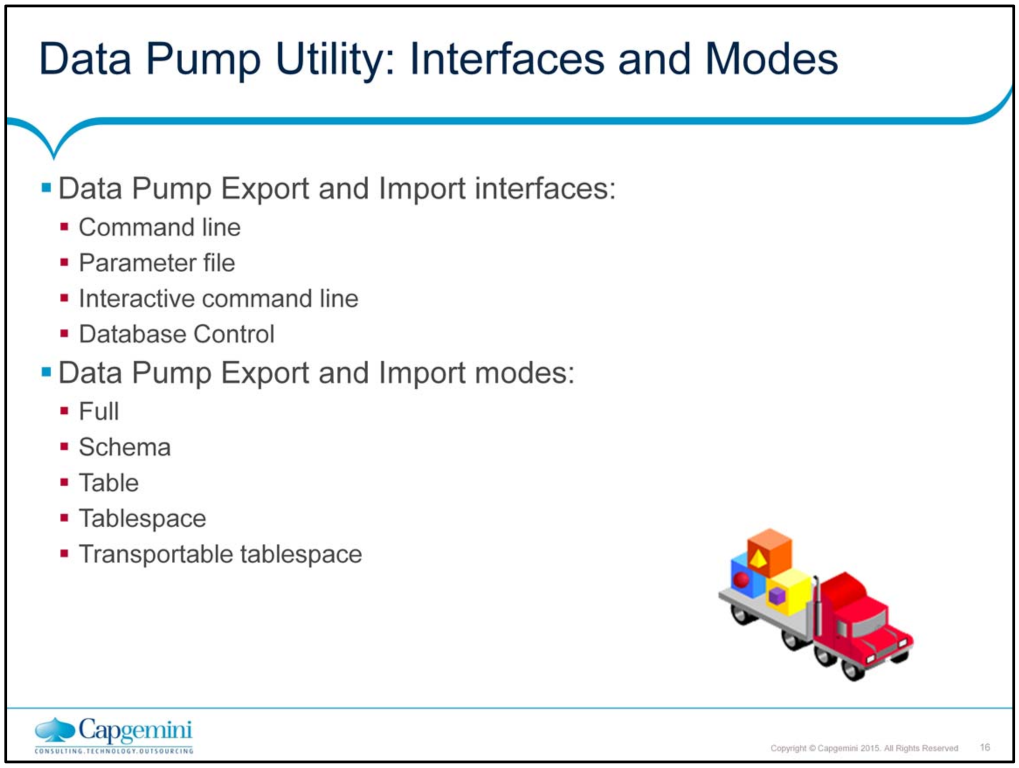
Data Pump Export is a utility for unloading data and metadata into a set of operating system files called dump file sets. Data Pump Import is used to load metadata and data stored in an export dump file set into a target system.

The Data Pump API accesses its files on the server rather than on the client.

These utilities can also be used to export from a remote database directly to a dump file set, or to load the target database directly from a source database with no intervening files. This is known as network mode. This mode is particularly useful to export data from a read-only source database.

At the center of every Data Pump operation is the master table (MT), which is a table created in the schema of the user running the Data Pump job. The MT maintains all aspects of the job. The MT is built during a file-based export job and is written to the dump file set as the last step. Conversely, loading the MT into the current user's schema is the first step of a file-based import operation and is used to sequence the creation of all objects imported.

Note: The MT is the key to Data Pump's restart capability in the event of a planned or unplanned stopping of the job. The MT is dropped when the Data Pump job finishes normally.



Data Pump Utility: Interfaces and Modes

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

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Data Pump Utility: Interfaces and Modes

You can interact with Data Pump Export and Import by using one of the following interfaces:

The command-line interface enables you to specify most of the export parameters directly on the command line.

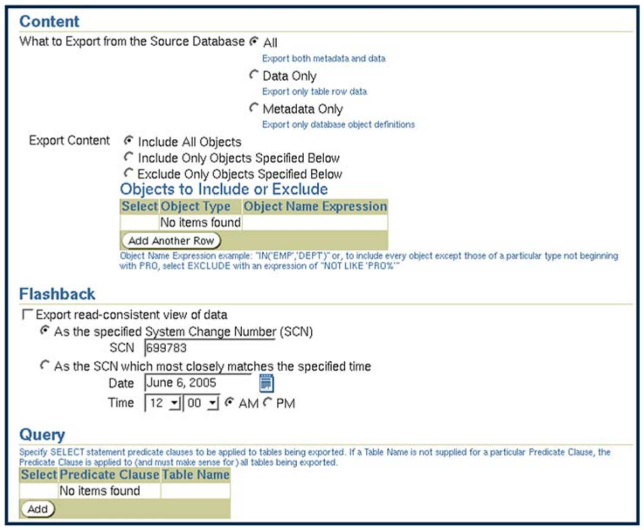
The parameter file interface enables you to specify all command-line parameters in a parameter file. The only exception is the PARFILE parameter.

The interactive-command interface stops logging to the terminal and displays the export or import prompts, where you can enter various commands. This mode is enabled by pressing [Ctrl] + [C] during an export operation that is started with the command-line interface or the parameter file interface. Interactive-command mode is also enabled when you attach to an executing or stopped job.

You can also access the Web interface. On the Database Control home page, click the Maintenance tab, and then select one of the following links from the Utilities region: Export to Files, Import from Files, or Import from Database.

Data Pump Export and Import provide different modes for unloading and loading different portions of the database. The mode is specified on the command line by using the appropriate parameter. The available modes are listed in the slide and are the same as with original export and import utilities.

Fine-Grained Object Selection



Content

What to Export from the Source Database [✓ All] Data Only Metadata Only

Export both metadata and data

Export only table row data

Export only database object definitions

Export Content [✓ Include All Objects] Include Only Objects Specified Below Exclude Only Objects Specified Below

Objects to Include or Exclude

Select	Object Type	Object Name Expression
	No items found	

Add Another Row

Object Name Expression example: "IN('EMP','DEPT')" or, to include every object except those of a particular type not beginning with PRO, select EXCLUDE with an expression of "NOT LIKE 'PRO%'"

Flashback

☐ Export read-consistent view of data

[✓ As the specified System Change Number (SCN)]

SCN 699783

As the SCN which most closely matches the specified time

Date June 6, 2005

Time 12 00 [✓ AM] PM

Query

Specify SELECT statement predicate clauses to be applied to tables being exported. If a Table Name is not supplied for a particular Predicate Clause, the Predicate Clause is applied to (and must make sense for) all tables being exported.

Select	Predicate Clause	Table Name
	No items found	

Add

Directory Obj.
SQL*Loader
Data Pump
> - Export
- Import
External Table

Fine-Grained Object Selection

The Data Pump job can include or exclude virtually any type of object.

The EXCLUDE parameter enables any database object type to be excluded from an export or import operation. The optional name qualifier enables you to have finer selectivity within each object type that is specified. Examples:

```
EXCLUDE=VIEW
EXCLUDE=PACKAGE
EXCLUDE=INDEX:"LIKE 'EMP%'"
```

The INCLUDE parameter includes only the specified object types and objects in an operation. Syntax:

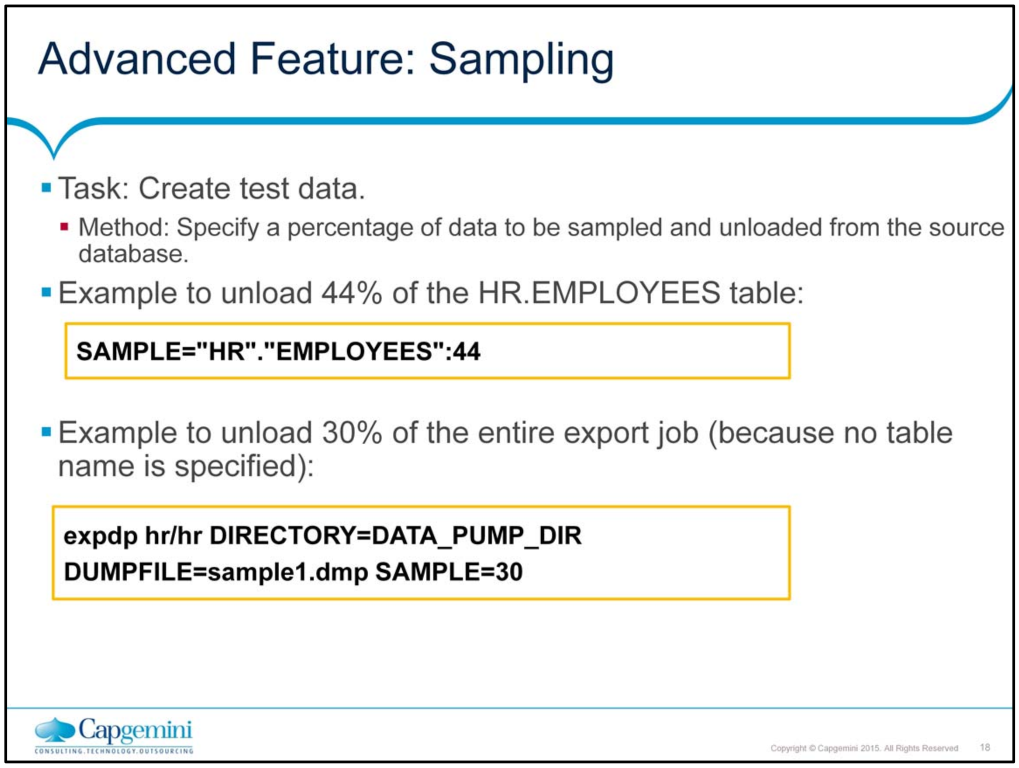
```
INCLUDE  = object_type[:"name_expr"]
```

The CONTENT parameter enables you to request for the current operation, only the metadata, only the data, or both. Syntax:

```
CONTENT = ALL | METADATA_ONLY |
DATA_ONLY
```

The QUERY parameter operates in a similar manner as the original export utility, with two significant enhancements: It can be qualified with a table name so that it applies to only that table, and it can be used during import as well. Example:

```
QUERY=hr.employees:"WHERE department_id in
(10,20) and salary < 1600 ORDER BY
department_id"
```



Advanced Feature: Sampling

- Task: Create test data.
  - Method: Specify a percentage of data to be sampled and unloaded from the source database.
- Example to unload 44% of the HR.EMPLOYEES table:

```
SAMPLE="HR"."EMPLOYEES":44
```

- Example to unload 30% of the entire export job (because no table name is specified):

```
expdp hr/hr DIRECTORY=DATA_PUMP_DIR
DUMPFILE=sample1.dmp SAMPLE=30
```

Advanced Feature: Sampling

With the SAMPLE parameter, you can specify a percentage of data to be sampled and unloaded from the source database when performing a Data Pump export.

Syntax:

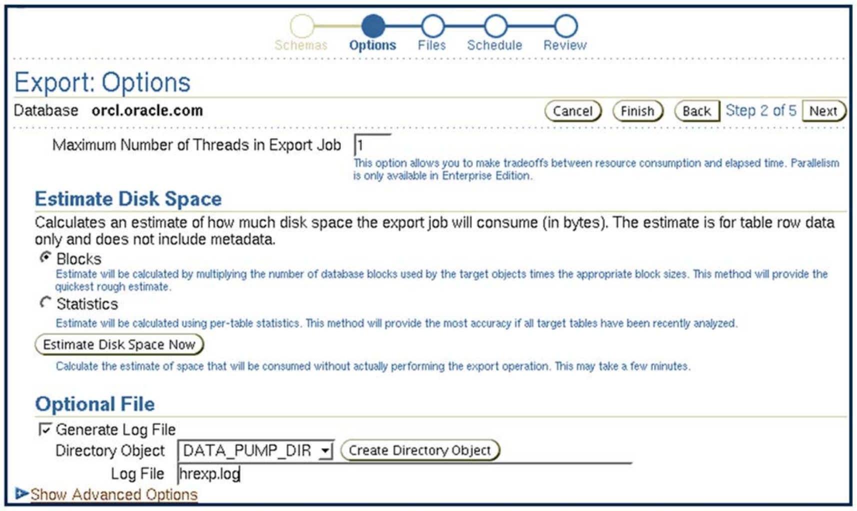
SAMPLE=[[schema_name.]table_name:]sample_percent

Range for sample_percent: .000001 to (but not including) 100

Sample percentage indicates the likelihood that a block of rows will be included.

Note: The SAMPLE parameter is not valid for network exports.

Export Options: Files



Export: Options

Database **orcl.oracle.com**

Maximum Number of Threads in Export Job 1

This option allows you to make tradeoffs between resource consumption and elapsed time. Parallelism is only available in Enterprise Edition.

Estimate Disk Space

Calculates an estimate of how much disk space the export job will consume (in bytes). The estimate is for table row data only and does not include metadata.

● Blocks

Estimate will be calculated by multiplying the number of database blocks used by the target objects times the appropriate block sizes. This method will provide the quickest rough estimate.

○ Statistics

Estimate will be calculated using per-table statistics. This method will provide the most accuracy if all target tables have been recently analyzed.

Calculate the estimate of space that will be consumed without actually performing the export operation. This may take a few minutes.

Optional File

✓ Generate Log File

Directory Object	DATA_PUMP_DIR
Log File	hrexp.log

Export Options: Files

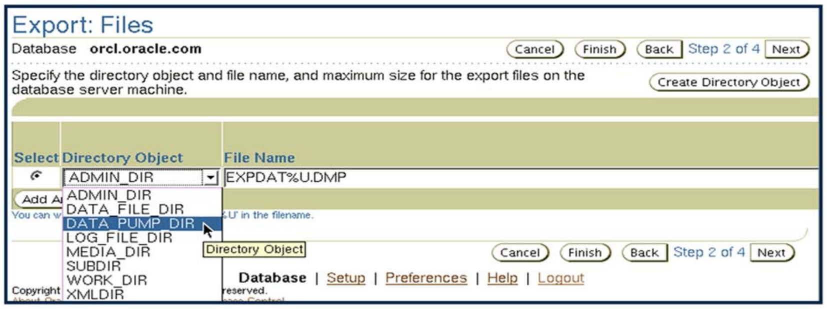
There are three types of files that are managed by Data Pump jobs:

- Dump files for data and metadata that is to be moved
- Log files for messages
- SQL files for the output of a SQLFILE operation

Because Data Pump is server based and not client based, Data Pump files are accessed relative to Oracle directory paths. Absolute paths are not supported for security reasons.

Data Pump File Locations

- The order of precedence of file locations:
  - Per-file directory
  - The DIRECTORY parameter
  - The DATA_PUMP_DIR environment variable
  - DATA_PUMP_DIR directory object


Export: Files

Database orcl.oracle.com

Specify the directory object and file name, and maximum size for the export files on the database server machine.

Select	Directory Object	File Name
●	ADMIN_DIR	EXPDAT%U.DMP

Data Pump File Locations

The slide shows you the order of precedence used by Data Pump clients to locate these files:

Per-file directory objects may be specified for each dump file, log file, and SQL file. If specified, they are separated from the file name by a colon (:).

The Data Pump Export and Import clients provide a DIRECTORY parameter, which specifies the name of a directory object. These directory objects describe the location in which the files are accessed.

You can alternatively define an environment variable, DATA_PUMP_DIR, to specify the directory object name rather than use the DIRECTORY parameter. The Data Pump clients look for this environment variable if no explicit directory object is specified.

There is a default directory object created for every database. This directory object is named DATA_PUMP_DIR. Access to the DATA_PUMP_DIR directory is granted automatically to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles.

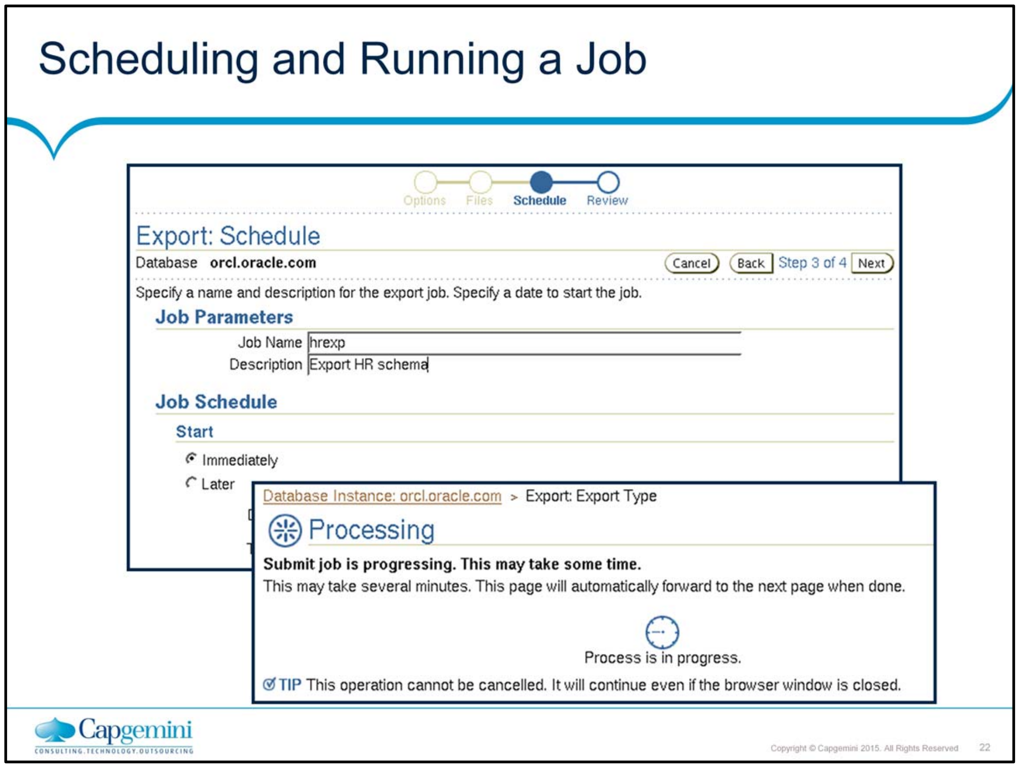
Data Pump File Locations (continued)

You do not need to create a directory object manually before using Data Pump Export. There is a default directory object created for every database, whether newly created or upgraded by a script on UNIX or Windows platforms. This directory object is named DATA_PUMP_DIR. Access to the DATA_PUMP_DIR directory is granted automatically to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles. The DATA_PUMP_DIR directory is created in one of the following locations:

<ORACLE_BASE>/admin/DB_UNIQUE_NAME
<ORACLE_HOME>/admin/DB_UNIQUE_NAME

The exact directory path specification for DATA_PUMP_DIR varies, depending on the value of the ORACLE_BASE and ORACLE_HOME system environment variables and on the existence of the DATA_PUMP_DIR subdirectory. If ORACLE_BASE is defined on the target system, then that value is used. Otherwise, the value of ORACLE_HOME is used. If, for some reason, the DATA_PUMP_DIR subdirectory is not found, the following default path is used: ORACLE_HOME/rdbms/log.

Note: In all cases, you must have the appropriate access privileges to the directory object for the operation attempted. For export, you need write access for all files; for import, you need read access for dump files and write access for log files and SQL files.



Scheduling and Running a Job

Export: Schedule

Database **orcl.oracle.com**

Specify a name and description for the export job. Specify a date to start the job.

Job Parameters

Job Name: hrexp

Description: Export HR schema

Job Schedule

Start

(●) Immediately

( ) Later

Database Instance: orcl.oracle.com > Export: Export Type

Processing

Submit job is progressing. This may take some time.

This may take several minutes. This page will automatically forward to the next page when done.

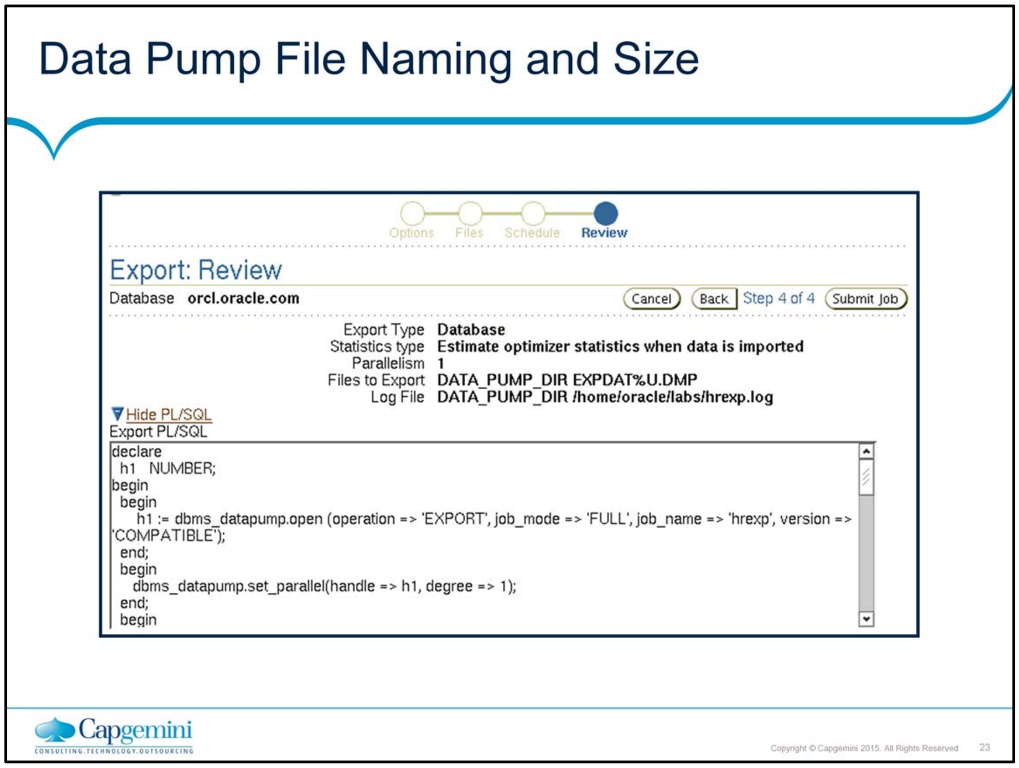
Process is in progress.

TIP This operation cannot be cancelled. It will continue even if the browser window is closed.

22

Scheduling and Running a Job

Data Pump jobs (created through this wizard) are scheduled as repeatable jobs by Enterprise Manager Database Control.



Data Pump File Naming and Size

Export: Review

Database orcl.oracle.com

Export Type	Database
Statistics type	Estimate optimizer statistics when data is imported
Parallelism	1
Files to Export	DATA_PUMP_DIR EXPDAT%U.DMP
Log File	DATA_PUMP_DIR /home/oracle/labs/hrexp.log

Export PL/SQL

```
declare
 h1   NUMBER;
begin
 begin
    h1 := dbms_datapump.open (operation => 'EXPORT', job_mode => 'FULL', job_name => 'hrexp', version =>
'COMPATIBLE');
 end;
 begin
    dbms_datapump.set_parallel(handle => h1, degree => 1);
 end;
 begin
```

Data Pump File Naming and Size

The DUMPFILE parameter specifies the names and (optionally) directories of disk-based dump files. Multiple file specifications may be provided as a comma-separated list or in separate DUMPFILE parameter specifications. File names may contain the substitution variable %U, which implies that multiple files may be generated. %U is expanded in the resulting file names into a two-character, fixed-width, monotonically increasing integer starting at 01. If no DUMPFILE is specified, expdat.dmp is used by default. By default, created dump files are autoextensible.

If FILESIZE is specified, each file is FILESIZE bytes in size and nonextensible. If more dump space is required and a template with %U has been supplied, then a new file is automatically created with FILESIZE bytes; otherwise, the client receives a message to add a new file.

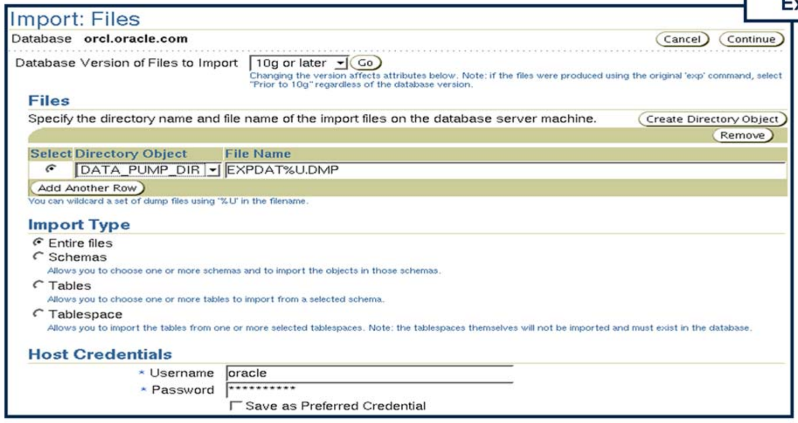
If a template with %U is specified, the number of files initially created is equal to the PARALLEL parameter.

Preexisting files that match the resulting file names are not overwritten; they result in an error and cause the job to be aborted.

Note: If multiple dump file templates are provided, then they are used to generate dump files in a circular fashion.

Data Pump Import

Directory Obj.
SQL*Loader
Data Pump
- Export
> - Import
External Table



Import: Files

Database **orcl.oracle.com**

Database Version of Files to Import: 10g or later

Changing the version affects attributes below. Note: if the files were produced using the original 'exp' command, select "Prior to 10g" regardless of the database version.

Files

Specify the directory name and file name of the import files on the database server machine.

Select	Directory Object	File Name
●	DATA_PUMP_DIR	EXPDAT%U.DMP

You can wildcard a set of dump files using '%U' in the filename.

Import Type

[✓ Entire files]
Schemas
Allows you to choose one or more schemas and to import the objects in those schemas.
Tables
Allows you to choose one or more tables to import from a selected schema.
Tablespace
Allows you to import the tables from one or more selected tablespaces. Note: the tablespaces themselves will not be imported and must exist in the database.

Host Credentials

* Username: oracle
* Password: **********
☐ Save as Preferred Credential

Data Pump Import

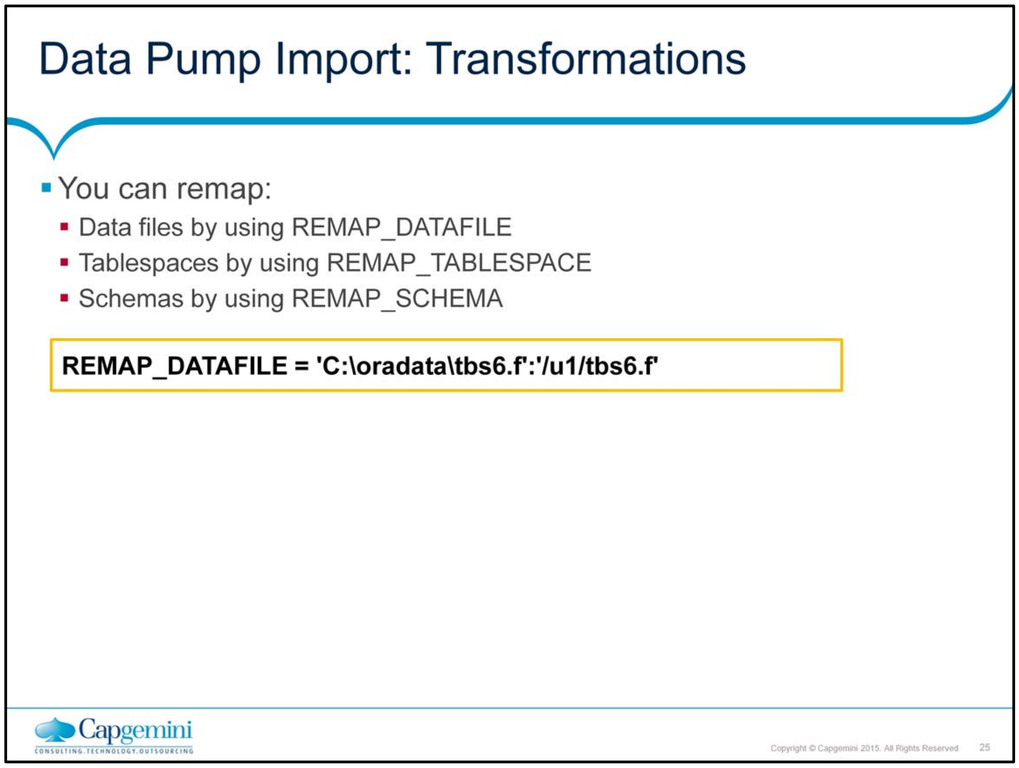
Data Pump Import is a utility for loading an export dump file set into a target system. The dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in a proprietary, binary format. During an import operation, the Data Pump Import utility uses these files to locate each database object in the dump file set.

You can interact with Data Pump Import by using a command line, a parameter file, or an interactive-command mode:

You can use the impdp command and specify parameters directly on the command line.

You can enter command-line parameters in a file (the PARFILE parameter is excluded because parameter files cannot be nested).

In interactive-command mode, the current job continues running, but logging to the terminal is stopped and the Import prompt is displayed. You can, for example, attach additional jobs to an executing or stopped job.



Data Pump Import: Transformations

- You can remap:
  - Data files by using REMAP_DATAFILE
  - Tablespaces by using REMAP_TABLESPACE
  - Schemas by using REMAP_SCHEMA

```
REMAP_DATAFILE = 'C:\oradata\tbs6.f':'/u1/tbs6.f'
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

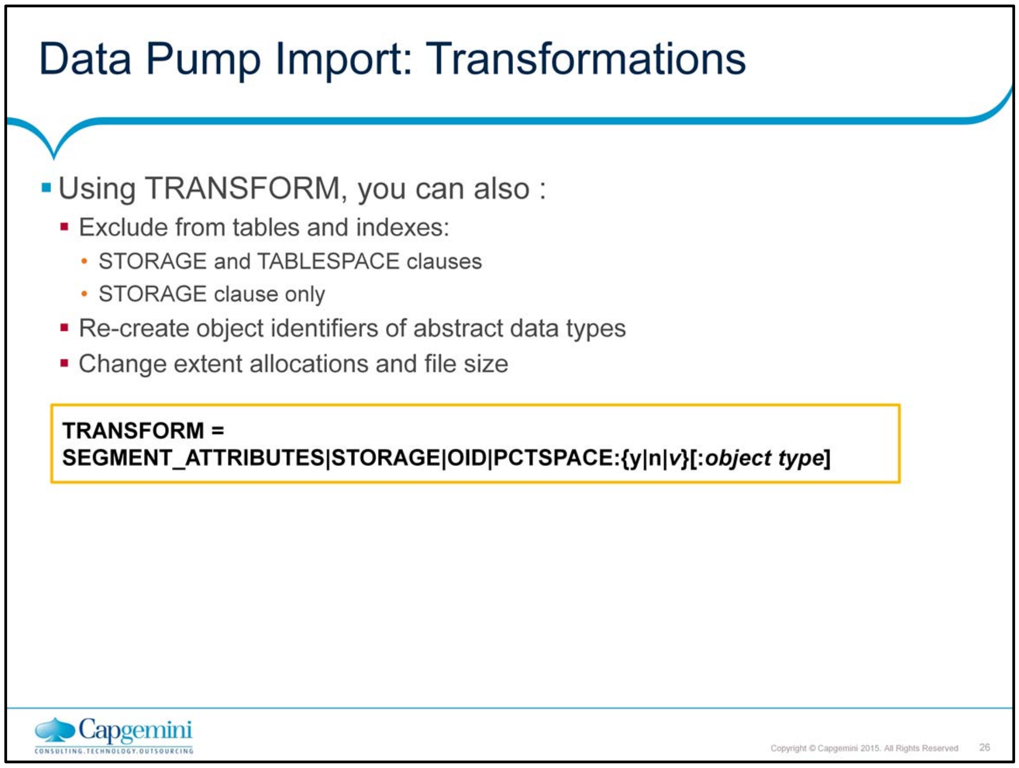
Data Pump Import: Transformations

Because object metadata is stored as XML in the dump file set, it is easy to apply transformations when DDL is being formed during import. Data Pump Import supports several transformations:

REMAP_DATAFILE is useful when moving databases across platforms that have different file-system semantics.

REMAP_TABLESPACE allows objects to be moved from one tablespace to another.

REMAP_SCHEMA provides the old FROMUSER /TOUSER capability to change object ownership.



Data Pump Import: Transformations

- Using TRANSFORM, you can also :
  - Exclude from tables and indexes:
    - STORAGE and TABLESPACE clauses
    - STORAGE clause only
  - Re-create object identifiers of abstract data types
  - Change extent allocations and file size

TRANSFORM =
SEGMENT_ATTRIBUTES|STORAGE|OID|PCTSPACE:{y|n|*v*}[:*object type*]

Data Pump Import: Transformations (continued)

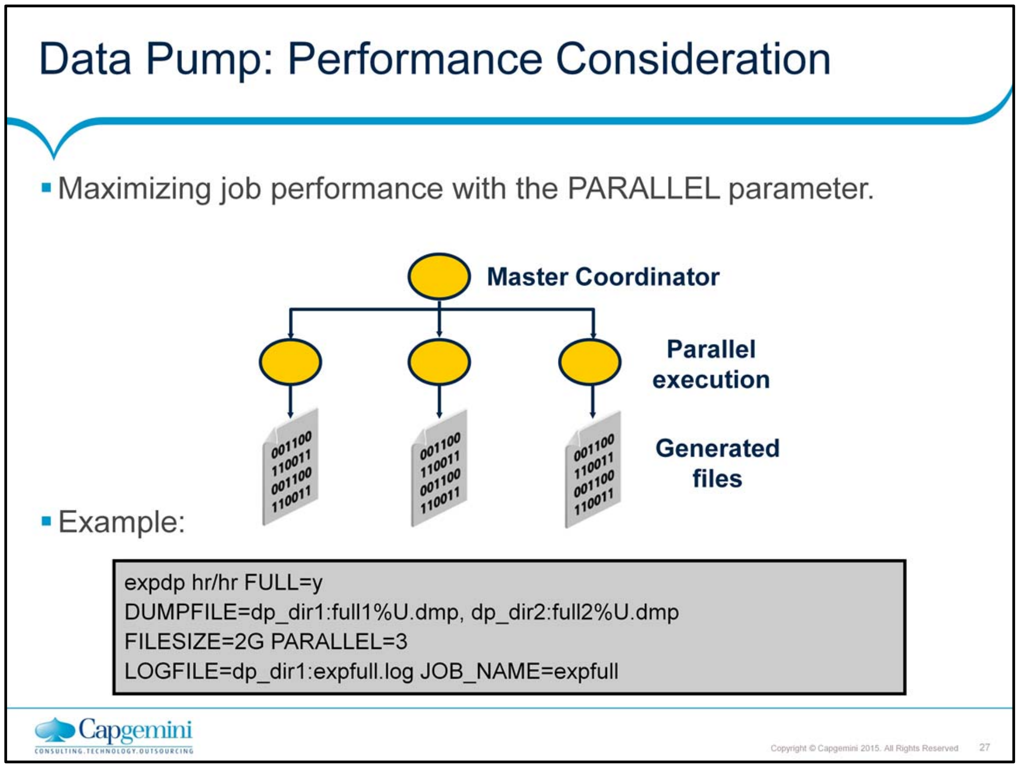
The TRANSFORM parameter enables you to alter the object creation DDL for specific objects or for all applicable objects being loaded. Specify the TRANSFORM parameter as shown in the slide. The following are possible options:

SEGMENT_ATTRIBUTES: If the value is specified as Y, segment attributes (physical attributes, storage attributes, tablespaces, and logging) are included.

STORAGE: If the value is specified as Y, the STORAGE clauses are included.

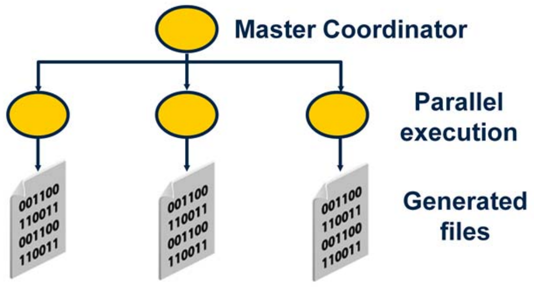
OID: You can use this parameter to determine whether the object ID (OID) of abstract data types is reused or created anew. If the value is specified as N, then the generation of the export OID clause for object types is suppressed. This is useful when you need to duplicate schemas across databases by using export and import, but you cannot guarantee that the object types will have identical OID values in those databases.

PCTSPACE: You can use the PCTSPACE parameter to reduce the amount of space required for tablespaces by performing a shrink operation on tablespace storage allocation. The value supplied for this transformation must be a number greater than zero. It represents the percentage multiplier used to alter extent allocations and the size of data files.



Data Pump: Performance Consideration

- Maximizing job performance with the PARALLEL parameter.



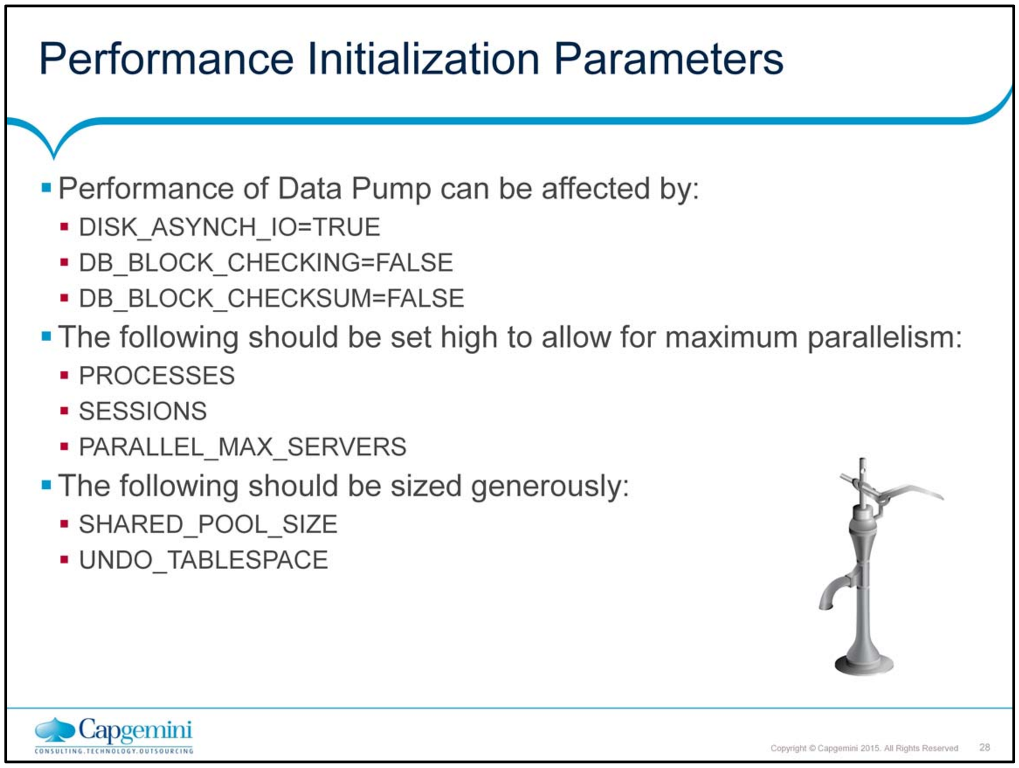
- Example:

```
expdp hr/hr FULL=y
DUMPFILE=dp_dir1:full1%U.dmp, dp_dir2:full2%U.dmp
FILESIZE=2G PARALLEL=3
LOGFILE=dp_dir1:expfull.log JOB_NAME=expfull
```

Data Pump: Performance Consideration

You can improve throughput of a job with the PARALLEL parameter. The parallelism setting is enforced by the master process, which allocates work to be executed to worker processes that perform the data and metadata processing within an operation. These worker processes operate in parallel. In general, the degree of parallelism should be set to more than twice the number of CPUs on an instance. To maximize parallelism, you must supply at least one file for each degree of parallelism. If there are not enough dump files, the performance will not be optimal because multiple threads of execution will be trying to access the same dump file. The degree of parallelism can be reset at any time during a job.

The example in the slide shows a full database export. All data and metadata in the database will be exported. Dump files (full101.dmp, full201.dmp, full102.dmp, and so on) will be created in a round-robin fashion in the directories pointed to by the dp_dir1 and dp_dir2 directory objects. For best performance, these should be on separate I/O channels. Each file will be up to 2 gigabytes in size, as necessary. Initially, up to three files will be created. More files will be created, if needed. The job and master table have the same name: expfull. The log file will be written to expfull.log in the dp_dir1 directory.

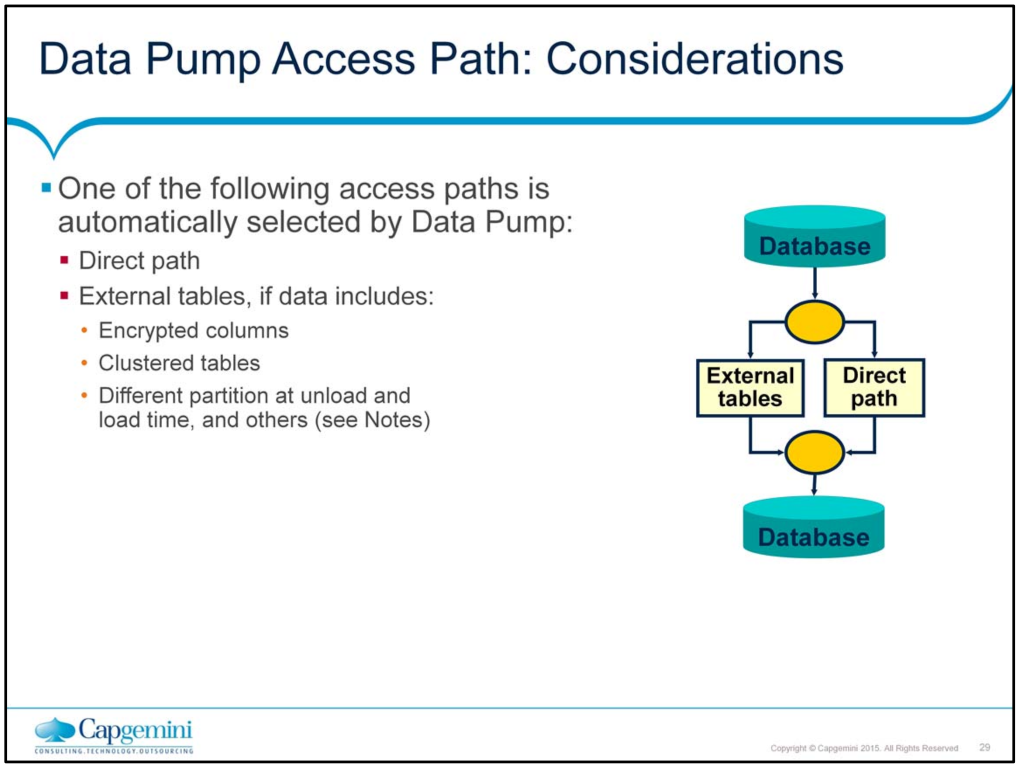


Performance Initialization Parameters

- Performance of Data Pump can be affected by:
  - DISK_ASYNCH_IO=TRUE
  - DB_BLOCK_CHECKING=FALSE
  - DB_BLOCK_CHECKSUM=FALSE
- The following should be set high to allow for maximum parallelism:
  - PROCESSES
  - SESSIONS
  - PARALLEL_MAX_SERVERS
- The following should be sized generously:
  - SHARED_POOL_SIZE
  - UNDO_TABLESPACE

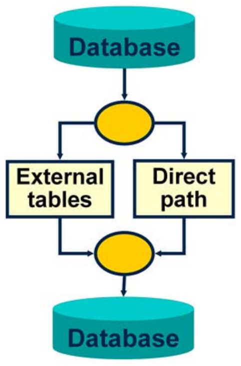
Performance Initialization Parameters

You can try using the parameters (shown in the slide) to improve performance, although the effect may not be the same on all platforms. Additionally, the SHARED_POOL_SIZE and UNDO_TABLESPACE initialization parameters should be generously sized. The exact values will depend upon the size of your database.



Data Pump Access Path: Considerations

- One of the following access paths is automatically selected by Data Pump:
  - Direct path
  - External tables, if data includes:
    - Encrypted columns
    - Clustered tables
    - Different partition at unload and load time, and others (see Notes)



Data Pump Direct Path: Considerations

Data Pump automatically selects the most appropriate access method for each table.

Data Pump uses direct path load and unload when a table's structure allows it and when maximum single-stream performance is desired.

Data Pump uses external tables, if any of the conditions exist:

Tables with fine-grained access control enabled in insert and select modes

Domain index, which exists for a LOB column

Tables with active triggers defined

Global index on partitioned tables with a single-partition load

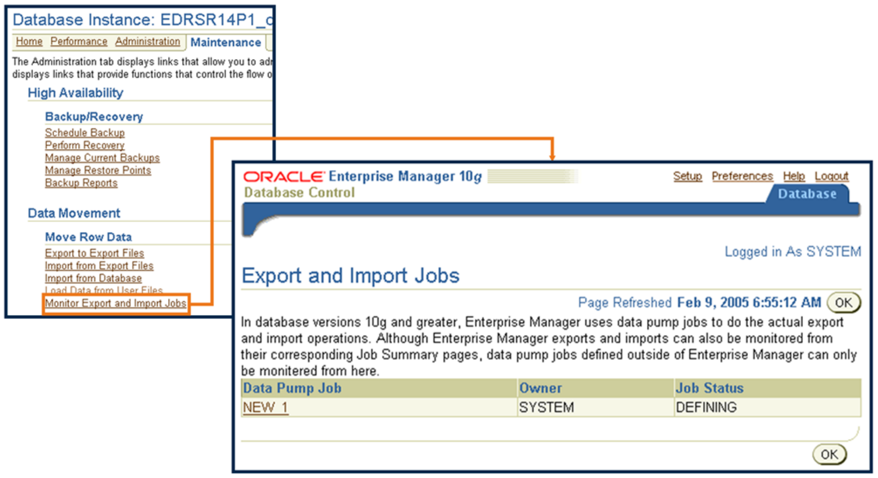
BFILE or opaque type columns

Referential integrity constraint

VARRAY columns with an embedded opaque type

Note: Because both methods support the same external data representation, data that is unloaded with one method can be loaded using the other method.

Using Enterprise Manager to Monitor Data Pump Jobs



Database Instance: EDRSR14P1_c

Home Performance Administration Maintenance

The Administration tab displays links that allow you to adr
displays links that provide functions that control the flow o

High Availability

Backup/Recovery

Schedule Backup
Perform Recovery
Manage Current Backups
Manage Restore Points
Backup Reports

Data Movement

Move Row Data

Export to Export Files
Import from Export Files
Import from Database
Load Data from User Files
Monitor Export and Import Jobs

ORACLE Enterprise Manager 10*g* Database Control

Setup Preferences Help Logout

Database

Logged in As SYSTEM

Export and Import Jobs

Page Refreshed **Feb 9, 2005 6:55:12 AM** OK

In database versions 10g and greater, Enterprise Manager uses data pump jobs to do the actual export and import operations. Although Enterprise Manager exports and imports can also be monitored from their corresponding Job Summary pages, data pump jobs defined outside of Enterprise Manager can only be monitered from here.

Data Pump Job	Owner	Job Status
NEW_1	SYSTEM	DEFINING

OK

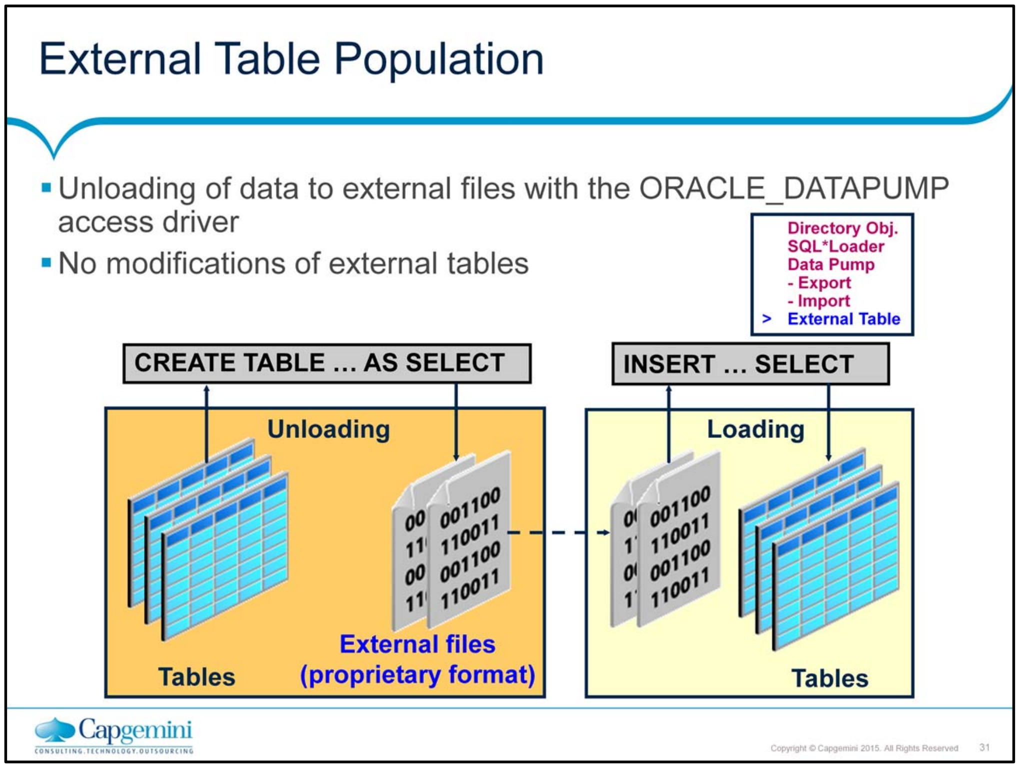
30

Using Enterprise Manager to Monitor Data Pump Jobs

You can use the Enterprise Manager graphical user interface (GUI) to monitor all Data Pump jobs, including those created by using the expdp or impdp command-line interfaces or by using the DBMS_DATAPUMP package.

You can view the current status of the job and also change the status to EXECUTE, STOP, or SUSPEND.

To access the Export and Import Jobs page, click the Monitor Export and Import Jobs link in the Move Row Data region of the Maintenance page.



External Table Population

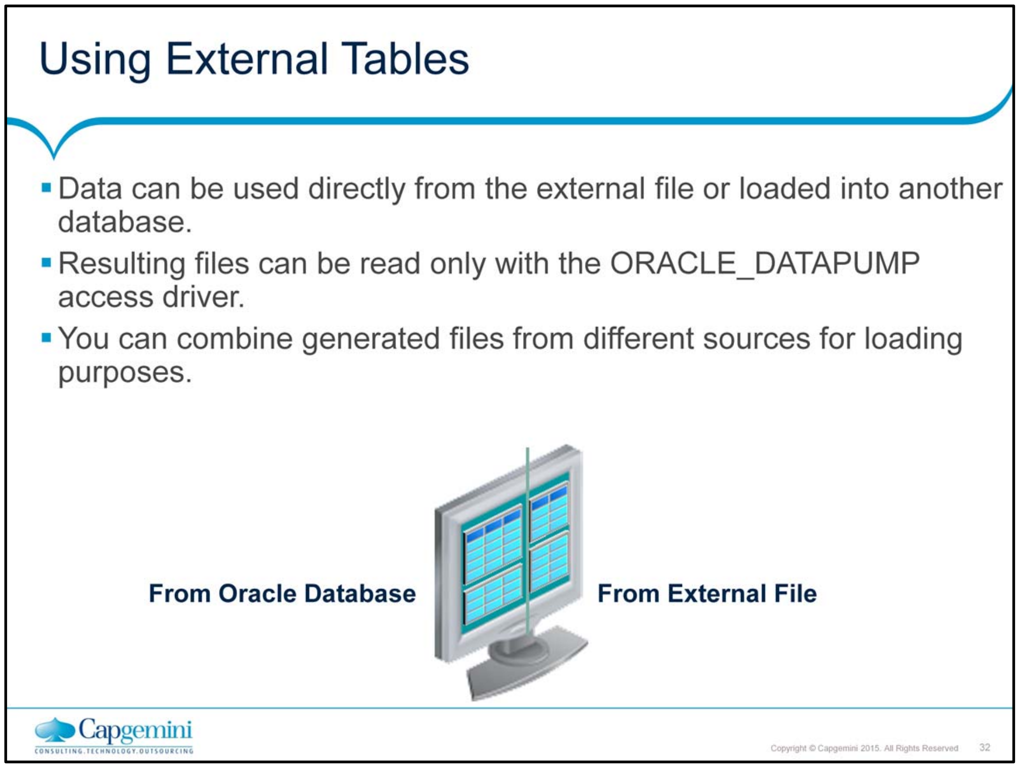
- Unloading of data to external files with the ORACLE_DATAPUMP access driver
- No modifications of external tables

External Table Population

An "external table" is composed of proprietary format (that is, Direct Path API) flat files that are operating system independent. As data is extracted from the Oracle database and "unloaded" into files, it is transparently converted from its Oracle internal representation into an equivalent Oracle native external representation (that is, DPAPI).

You may use the CREATE TABLE AS SELECT command to populate an external table. After an external table has been created and populated, no rows may be added, updated, or deleted from the external table. Any attempt to modify the data in the external table fails. An external table may not have indexes.

The Data Pump access driver enables the unloading and loading operations for external tables.



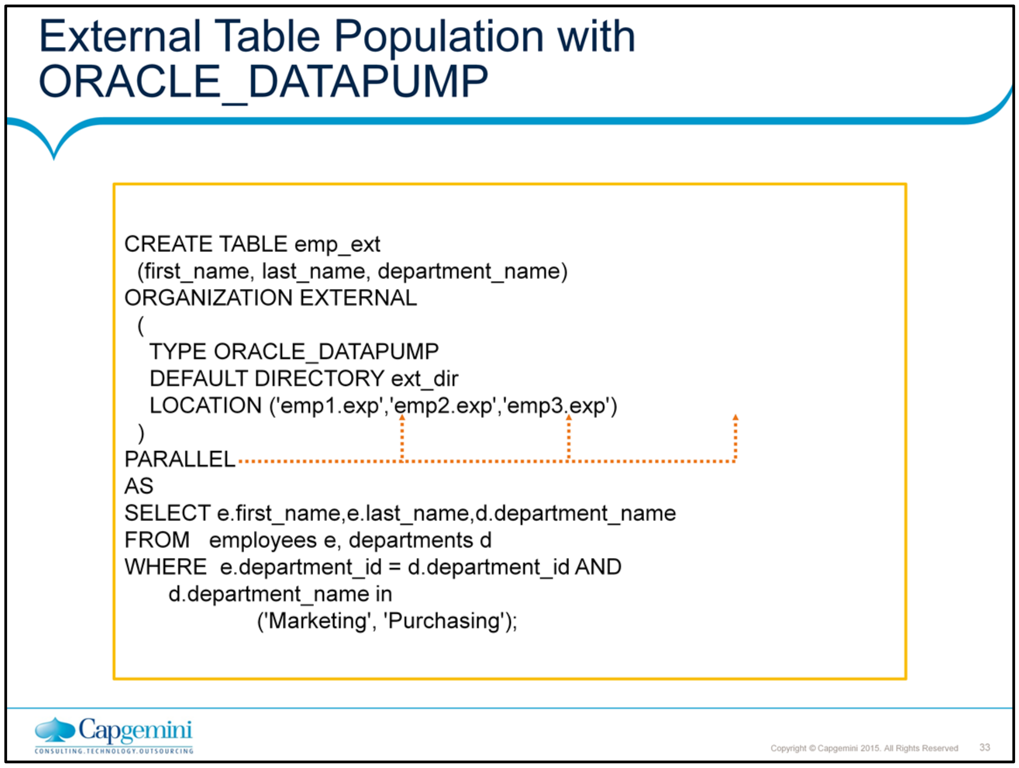
Using External Tables

- Data can be used directly from the external file or loaded into another database.
- Resulting files can be read only with the ORACLE_DATAPUMP access driver.
- You can combine generated files from different sources for loading purposes.

Using External Tables

The data files created for the external table can be moved and used as the data files for another external table in the same database or different database. They can be read only by the ORACLE_DATAPUMP access driver. You can choose whether your applications should directly access external tables with the SELECT command, or if the data should first be loaded into a target database.

Data files populated by different external tables can all be specified in the LOCATION clause of another external table. This provides an easy way of aggregating data from multiple sources. The only restriction is that the metadata for all the external tables must be exactly the same.



External Table Population with ORACLE_DATAPUMP

```
CREATE TABLE emp_ext
  (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');
```

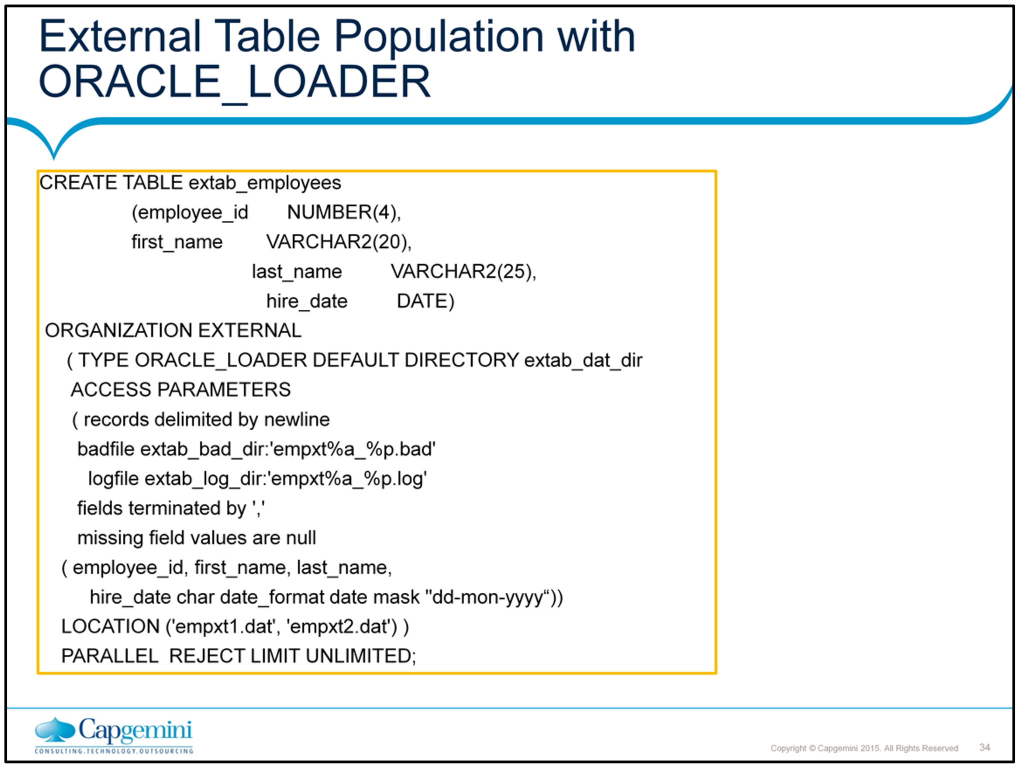
External Table Population with ORACLE_DATAPUMP

This example shows you how the new external table population operation can help to export a selective set of records resulting from the join of the EMPLOYEES and DEPARTMENTS tables.

Because the external table can be large, you can use a parallel populate operation to unload your data to an external table. As opposed to a parallel query from an external table, the degree of parallelism of a parallel populate operation is constrained by the number of concurrent files that can be written to by the access driver. There is never more than one parallel execution server writing into one file at a particular point in time.

The number of files in the LOCATION clause must match the specified degree of parallelism because each input/output (I/O) server process requires its own file. Any extra files that are specified are ignored. If there are not enough files for the specified degree of parallelism, the degree of parallelization is lowered to match the number of files in the LOCATION clause.

Note: For more information about the ORACLE_DATAPUMP access driver parameters, see the Oracle Database Utilities guide.



External Table Population 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;
```

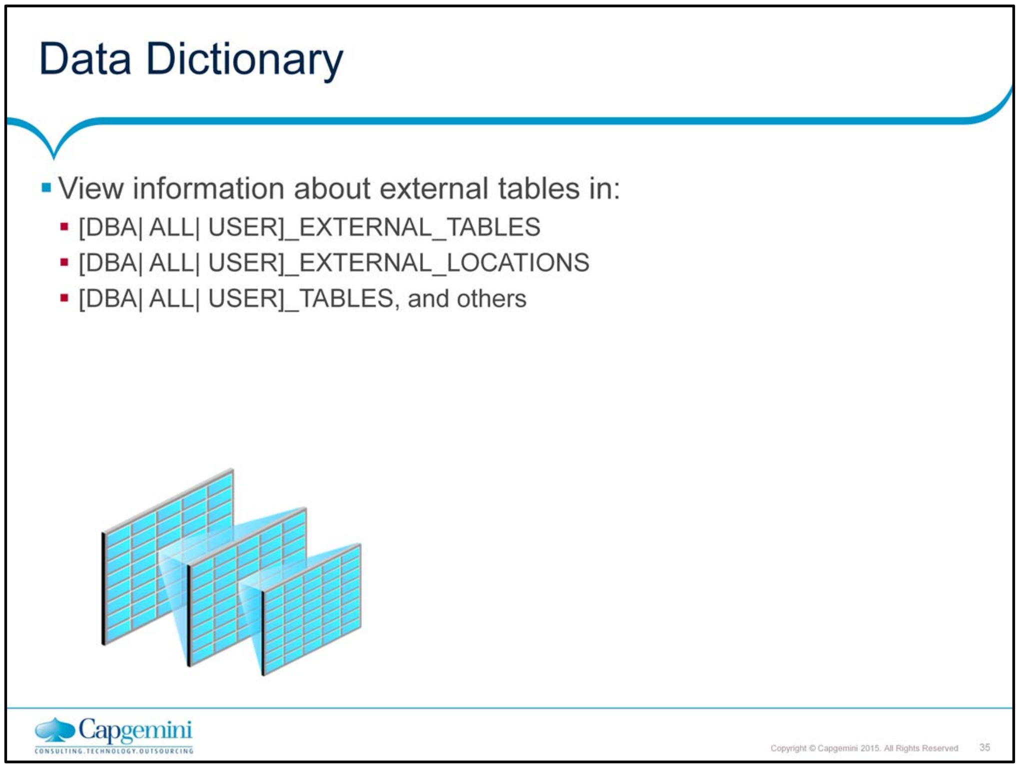
External Table Population with ORACLE_LOADER

The ORACLE_LOADER access driver uses the SQL*Loader syntax to create external tables.

The example shown in the slide assumes that three directory objects (extab_dat_dir, extab_bad_dir, and extab_log_dir) are created and mapped to existing OS directories, to which the user is granted access.

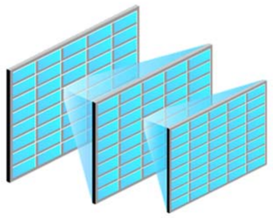
Tip: If you have a lot of data to load, enable PARALLEL for the load operation:

ALTER SESSION ENABLE PARALLEL DML;



Data Dictionary

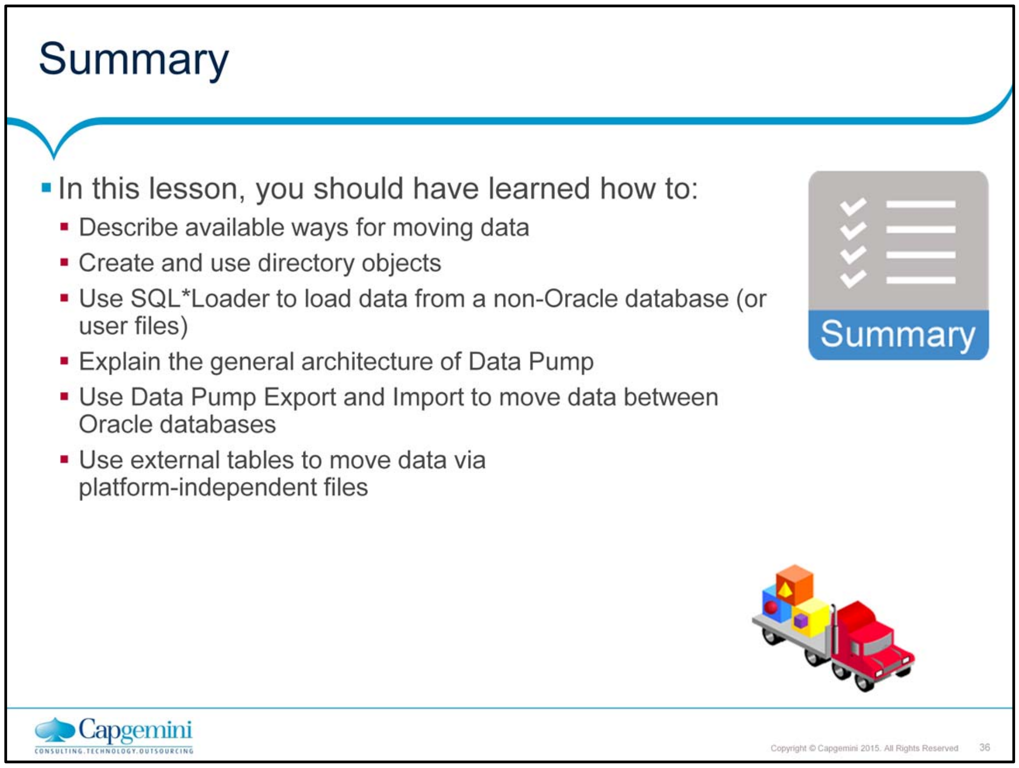
- View information about external tables in:
  - [DBA| ALL| USER]_EXTERNAL_TABLES
  - [DBA| ALL| USER]_EXTERNAL_LOCATIONS
  - [DBA| ALL| USER]_TABLES, and others



Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Data Dictionary

[DBA| ALL| USER]_EXTERNAL_TABLES list the specific attributes of external tables in the database.
[DBA| ALL| USER]_EXTERNAL_LOCATIONS list the data sources for external tables.
[DBA| ALL| USER]_TABLES describe relational tables in the database.
[DBA| ALL| USER]_TAB_COLUMNS describe the columns of tables, views, and clusters in the database.



Summary

- In this lesson, you should have learned how to:
  - Describe available ways for moving data
  - Create and use directory objects
  - Use SQL*Loader to load data from a non-Oracle database (or user files)
  - Explain the general architecture of Data Pump
  - Use Data Pump Export and Import to move data between Oracle databases
  - Use external tables to move data via platform-independent files