A Framework for Managing Imports through External Tables

By Michael McLaughlin ♠

http://blog.mclaughlinsoftware.com/2012/03/05/oracle-csv-imports/

Objectives

- Review push or pull imports
- Review Oracle CSV file import options
- Implement Oracle import user/schema
- Implement external tables and logs
- Protect read errors

Push/Pull Imports - Defined

- Push import
  - Originates somewhere else
  - Holds rights to upload a particular file
  - Uploads (pushes) the file to a remote server
- Pull import
  - Originates somewhere else
  - Grants read permissions to an external source
  - Uploads (pulls) the file from a remote server

Push/Pull Imports – Host Hardening

- Host hardening
  - Limits user access to minimum directories
  - Limits user access to minimum files
  - Limits connection types, duration, and origin
- File system management
  - Restrict directory permissions
  - Restrict file permissions
  - Organize, control, and manage user access

Oracle CSV File Import Options

- Comma-separated Values (CSV) file
  - Comma-delimited string fields
  - Double quote enclosed string fields when they include a white space
- Programming solutions
  - C-callable language, Java, PHP, Perl, et cetera
- Non-programming solutions
  - SQL*Loader or External Files

Implement Import User/Schema

- Create the user/schema:

```
CREATE USER import IDENTIFIED BY import
DEFAULT TABLESPACE users QUOTA 1000M ON users
TEMPORARY TABLESPACE temp;
```

- Grant permissions to the user/schema:

```
GRANT create cluster, create indextype, create operator
,     create procedure, create sequence, create session
,     create table, create trigger, create type
,     create view TO import;
```

Implement Virtual Directories

- Create the physical directories in the file system.
  - Windows: Absolute directory path from a logical drive
  - Linux/Unix: Absolute directory path from a mount point
- Create the Virtual Directories:

```
CREATE DIRECTORY upload_files AS 'C:\Imports\ImportFiles';
CREATE DIRECTORY upload_logs AS 'C:\Imports\ImportLogs';
```

- Grant permissions to the user on the virtual directories:

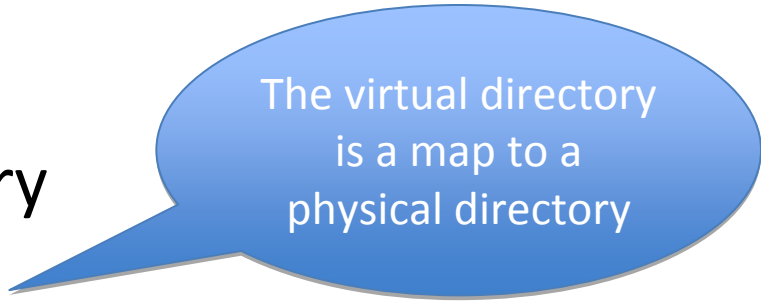
```
GRANT READ ON DIRECTORY upload_files TO import;
GRANT READ, WRITE ON DIRECTORY upload_logs TO import;
```

Implement External Table

```
CREATE TABLE item_import_ext_table
( asin_number         VARCHAR2(10) ... , ...
, item_release_date   DATE) ORGANIZATION EXTERNAL
( TYPE oracle_loader DEFAULT DIRECTORY upload_files
  ACCESS PARAMETERS
  ( RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    BADFILE     'UPLOAD_LOGS':'item_import_ext_table.bad’
    DISCARDFILE 'UPLOAD_LOGS':'item_import_ext_table.dis’
    LOGFILE     'UPLOAD_LOGS':'item_import_ext_table.log’
    FIELDS TERMINATED BY ','
    OPTIONALLY ENCLOSED BY "'”
    MISSING FIELD VALUES ARE NULL )
  LOCATION ('item_import.csv'))
REJECT LIMIT UNLIMITED;
```

Query Missing External Files

- They work when the file is
  - Found in the physical directory
  - Found in the virtual directory
- They raise this exception without a file:




```
SELECT * FROM item_import_ext_table
*
ERROR at line 1:
ORA-29913: error in executing ODCIEXTTABLEOPEN callout
ORA-29400: data cartridge error
KUP-04040: file item_import.csv in UPLOAD_FILES not found
```

Framework Infrastructure

- A function to get the physical directory
- An Oracle Attribute Data Type (ADT)
- A Java library to read external directory
- A DBMS_JAVA security grant
- A PL/SQL wrapper for the Java library
- A mapping query of table to source file
- A function leveraging file existence
- A view to encapsulate the framework logic

Grant from SYS to SYSTEM

- A direct gran of the select privilege on the administrative view to the SYSTEM schema is required to include it in a function:

```
GRANT SELECT
 ON sys.dba_directories
 TO system;
```

Physical Directory PL/SQL Function

```
CREATE OR REPLACE FUNCTION get_directory_path
( virtual_directory IN VARCHAR2 ) RETURN VARCHAR2 IS
  -- Define RETURN variable.
  directory_path VARCHAR2(256) := '';
  --Define dynamic cursor.
  CURSOR get_directory (virtual_directory VARCHAR2) IS
    SELECT   directory_path
    FROM     sys.dba_directories
    WHERE    directory_name = virtual_directory;
  -- Define a LOCAL exception FOR name violation.
  directory_name EXCEPTION;
  PRAGMA EXCEPTION_INIT(directory_name,-22284);
  ...
```

Physical Directory PL/SQL Function (continue)

```
CREATE OR REPLACE FUNCTION get_directory_path
( virtual_directory IN VARCHAR2 ) RETURN VARCHAR2 IS
  ...
BEGIN
  OPEN  get_directory (virtual_directory);
  FETCH get_directory
  INTO  directory_path;
  CLOSE get_directory;
  -- RETURN file name.
  RETURN directory_path;
EXCEPTION
  WHEN directory_name THEN
    RETURN NULL;
END get_directory_path;
/
```

Java Library to Read External Files

- Deploy the function in the SYSTEM schema
- Grant execute privilege to the IMPORT schema

```
GRANT EXECUTE ON get_directory_path TO import;
```

- Create a synonym for the function in the IMPORT schema

```
CREATE SYNONYM get_directory_path
 FOR system.get_directory_path;
```

- A call to the function maps a virtual directory to a physical directory name:

```
SELECT   get_directory_path('UPLOAD_FILES')
FROM     dual;
```

Attribute Data Type

- An ADT is a varray or nested table of a scalar data type
- It always returns a single column_value column
- Create an ADT for a list of files

```
CREATE OR REPLACE TYPE file_list
 AS TABLE OF VARCHAR2(255);
```

- A SQL ADT is available in the data catalog for calls by Java libraries

Java Library to Read Files

```
CREATE OR REPLACE AND COMPILE JAVA SOURCE NAMED "ListVirtualDirectory" AS
   … import statements …
public class ListVirtualDirectory {
  public static ARRAY getList(String path) throws SQLException {
    ARRAY listed = null;
    Connection conn =
      DriverManager.getConnection("jdbc:default:connection:");
    try {
      File directory = new File(path);
      ArrayDescriptor arrayDescriptor =
        new ArrayDescriptor("FILE_LIST",conn);
      listed =
        new ARRAY(arrayDescriptor,conn,((Object[]) directory.list())); }
    catch (AccessControlException e) {}
  return listed; }}
/
```

Attribute Data Type

- Grant *READ* permissions to the IMPORT schema in the Java permissions file:

```
BEGIN
  DBMS_JAVA.GRANT_PERMISSION('IMPORT’
                             ,'SYS:java.io.FilePermission’
                             ,'C:\Imports\ImportFiles’
                             ,'read');
  END;
/
```

PL/SQL Wrapper for Java

- Create a PL/SQL Wrapper for the Java library

```
CREATE OR REPLACE FUNCTION list_files(path VARCHAR2)
     RETURN FILE_LIST IS
     LANGUAGE JAVA
     NAME 'ListVirtualDirectory.getList(java.lang.String)
     return oracle.sql.ARRAY';
/
```

- Query the wrapper file

```
SELECT   column_value
   FROM     TABLE(
              list_files(
                get_directory_path('UPLOAD_FILES')));
```

Mapping Query

- The query uses administrative views to map an external table to a physical file

```
SELECT   xt.table_name
,        xt.file_name
FROM    (SELECT   uxt.table_name
         ,        ixt.column_value AS file_name
         FROM     user_external_tables uxt CROSS JOIN
         TABLE(list_files(
           get_directory_path(
             uxt.default_directory_name))) ixt) xt
JOIN     user_external_locations xl
ON       xt.table_name = xl.table_name
AND      xt.file_name = xl.location;
/
```

Mapping Query Function

- The query uses the administrative views to map an external table to a physical directory

```
CREATE OR REPLACE FUNCTION external_file_found
( table_in VARCHAR2 ) RETURN NUMBER IS
  retval NUMBER := 0;
  CURSOR c (table_in VARCHAR2) IS
    ... mapping query ...
    AND xt.TABLE_NAME = UPPER(table_in);
BEGIN
  FOR i IN c(table_in) LOOP
    retval := 1;
  END LOOP;
  RETURN retval;
END;
/
```

Error Proof View to External Table

- The view is query safe, with or without the physical file

```
CREATE OR REPLACE VIEW item_import AS
  SELECT   *
  FROM     item_import_ext_table
  WHERE    external_file_found('ITEM_IMPORT_EXT_TABLE') = 1;
/
```

Review

- Reviewed push or pull imports
- Reviewed Oracle CSV file import options
- Implemented Oracle import user/schema
- Implemented external tables and logs
- Protected read errors

Questions & Answers