ORACLE DATABASE 11g

An Oracle White Paper
July 2011

Using the Preprocessor Feature with External Tables in Oracle Database 11*g* Release *2*

ORACLE

Introduction

The preprocessor feature in Oracle Database 11*g* Release 2 (first released in Oracle Database 11.1.0.7 and Oracle Database 10.2.0.5) allows users to preprocess input data before it is sent to the access driver. The ability to manipulate input data with a preprocessor program results in additional loadable data formats, which greatly enhances the flexibility and processing power of external tables.

The types of preprocessor programs that can be used are versatile, ranging from system commands, user-generated binaries (for example, a C program), or user-supplied shell scripts. Because the user supplies the program to preprocess the data, it can be tailored to meet the user's specific needs. This means that the number of loadable formats is restricted only by the ability to manipulate the original data set. In addition to the compression example used in this white paper, additional examples of using the preprocessor include converting character data from upper to lower case or vice-versa, and excluding certain data. The preprocessor feature reads as input the standard output of the preprocessor program. Preprocessor programs must write to standard output for the feature to work correctly.

The New PREPROCESSOR Access Parameter

To support the new preprocessor feature, a new PREPROCESSOR parameter has been added to the RECORDS clause of the ORACLE_LOADER access driver. The syntax of the PREPROCESSOR parameter is as follows:

directory_spec – specifies the directory object containing the name of the preprocessor program to execute for every data file. If *directory_spec* is omitted, then the default directory specified for the external table is used. The user trying to access the external table must have the EXECUTE privilege for the specified directory object.

file-spec - the name of the preprocessor program. It is appended to the path name associated with the directory object (either the *directory_spec* or the default directory for the external table). The *file-spec* cannot contain an absolute or relative directory path.

Note that if *file-spec* requires any arguments, then *file-spec* along with the arguments must be placed in an executable shell script (or on Windows systems, in a batch (.bat) file), and it must reside in *directory_spec*. (See Example 2.)

Examples Using the New PREPROCESSOR Access Parameter

The following two examples demonstrate use of the preprocessor feature. In both examples, input data in compressed format is preprocessed to uncompress it and then sent to the ORACLE_LOADER access driver in uncompressed format. Example 1 specifies the preprocessor program directly on the PREPROCESSOR clause. Example 2 specifies the preprocessor program within a shell script, because the program uses additional arguments.

Example 1

This example requires you to take several distinct steps, including: supplying a preprocessor program; creating a simple data file, directory objects, and an external table; and then querying from that table to verify the data was successfully uncompressed.

Step 1 – Supply a preprocessor program

From within the shell, use the following command to copy the preprocessor program (in this case, zcat) to some other directory on your system.

% /bin/cp /bin/zcat /somedirectory/bin/zcat

Step 2 – Create a simple data file in compressed format

In another directory, create a simple data file named foo.dat containing the words "Hello World" and then use the gzip executable to compress the foo.dat file into a file named foo.dat.gz. Verify that the compressed file has been created by issuing an ls -l command.

```
% echo "Hello World" > foo.dat

% gzip foo.dat

% ls -l foo.dat.gz

-rw-rw-r--  1 oracle dba 40 Oct  2 15:10 foo.dat.gz
```

Step 3 – Create directory objects and grant required privileges

Directory objects must be created for the directories that hold the preprocessor programs and data files. For this example, the necessary privileges on those directories are granted to user scott. The creation of directory objects and granting of privileges on them to only certain users is necessary for security reasons (see BEST PRACTICES). In the following examples, replace the name 'somedirectory' with the name of the directory to which you copied the zcat program in step 1. Replace the name 'somedirectory1' with the name of the directory in which you created the foo.dat.gz file in step 2.

```
SQL> create or replace directory execdir as '/somedirectory/bin';

Directory created.

SQL> create or replace directory data_dir as '/somedirectory1';

Directory created.

SQL> grant read, execute on directory execdir to scott;

Grant succeeded.

SQL> grant read, write on directory data_dir to scott;

Grant succeeded.
```

Step 4 – Create an external table named xtab

```
SQL> CONNECT scott/tiger

Connected.

SQL> CREATE TABLE xtab (COL1 varchar2(2000))
  2  ORGANIZATION EXTERNAL (
  3   TYPE ORACLE_LOADER
  4  DEFAULT DIRECTORY data_dir
  5  ACCESS PARAMETERS (
```

```
 6  RECORDS DELIMITED BY NEWLINE
 7  PREPROCESSOR execdir:'zcat'
 8  FIELDS (COL1 char(2000)))
 9  LOCATION ('foo.dat.gz'))
10  REJECT LIMIT UNLIMITED
11  PARALLEL 2;
```

Table created.

Step 5 – Select from the external table to verify the data is uncompressed

SQL> SELECT * FROM xtab;

COL1

Hello World

1 row selected.

Example 2

The following example demonstrates using a shell script to uncompress the data. Shell scripts are necessary when preprocessor programs require additional arguments. Note that /bin/zcat and /bin/gunzip –c are functionally equivalent.

Step 1 – Create the preprocessor shell script

% echo '/bin/gunzip –c $1' > uncompress.sh

% chmod +x uncompress.sh

% cp uncompress.sh /somedirectory/bin/uncompress.sh

Note the following when creating a preprocessor shell script:

- The full path name must be specified for system commands (for example, gunzip)
- The data file listed in the external table LOCATION clause should be referred to by $1. (On Windows systems, the LOCATION clause should be referred to by %1.)
- On Windows systems, the first line in the .bat file must be the following:
- @echo off
- Otherwise, by default, Windows will echo the contents of the batch file (which will be treated as input by the external table access driver).

- Make sure the preprocessor shell script has EXECUTE permissions

Steps 2 and 3 remain the same as in Example 1.

Step 4 – Create an external table named xtab

```
SQL> CONNECT scott/tiger

Connected.

SQL> drop table xtab;

Table dropped.

SQL> CREATE TABLE xtab (COL1 varchar2(2000))
  2  ORGANIZATION EXTERNAL (
  3  TYPE ORACLE_LOADER
  4  DEFAULT DIRECTORY data_dir
  5  ACCESS PARAMETERS (
  6  RECORDS DELIMITED BY NEWLINE
  7  PREPROCESSOR execdir:'uncompress.sh'
  8  FIELDS (COL1 char(2000)))
  9  LOCATION ('foo.dat.gz'))
 10  REJECT LIMIT UNLIMITED
 11  PARALLEL 2;

Table created.
```

Step 5 – Select from the external table to verify the data is uncompressed

```
SQL> SELECT * FROM xtab;

COL1

Hello World

1 row selected.
```

Best Practices

Because a DBA does not necessarily know what actions a preprocessor program performs, precautions must be taken when allowing preprocessors to be used with external tables.

The OS system manager should create a separate directory to hold the preprocessors. Multiple directories may need to be created if different sets of Oracle users will use different preprocessors. The OS-user ORACLE must have appropriate OS-permissions to execute the image.

The OS system manager and the DBA need to ensure that the correct version of the preprocessor program is placed in the proper directory. The OS system manager should protect the preprocessor program from write access by any OS user.

The DBA should create a directory object for each directory that contains preprocessor programs. The DBA should grant the EXECUTE privilege to only those users who require access to the preprocessor programs.

To prevent a database user from accidentally or maliciously overwriting the preprocessor program, the DBA should NOT grant write access on the directory object containing the preprocessor programs to anyone.

Any data files required for an external table should be kept in a directory separate from the directory containing preprocessor programs. The DBA and the OS system manager should work together to ensure that only the appropriate OS users have access to the directory. Because the data files are in a different directory, a directory object needs to be created. The DBA should allow users only read access to the directory object in order to keep the data files from being accidentally or maliciously overwritten by the database user.

Any files generated by the access driver, including log files, bad files, and discard files, should be written to a directory object that is separate from the directory objects containing data files and directory objects containing the preprocessor. The DBA and the OS system manager need to work together to create a directory and ensure it has the proper protections. The database user may need to access these files to resolve problems in data files, so the DBA and OS system manager might want to determine a way for the database user to read those files.

Note that any DBA, as well as any user with the CREATE ANY DIRECTORY or DROP ANY DIRECTORY privilege also has full access to all directory objects. Therefore, those privileges should be used sparingly, if at all.

When the preprocessor option is specified, each file in the LOCATION clause is a single granule and the number of granules limits the degree of parallelism achievable. Therefore, the number of files in the LOCATION clause should be a multiple of the degree of parallelism. At lease one granule per slave is needed to exploit parallel operation across all slaves. In practice, 10 granules per slave is a good choice to avoid tail processing skew.

Conclusion

The simple example in this white paper illustrates how the power and flexibility of external tables has increased with the new preprocessing feature. The ability to use a preprocessor program to manipulate the input data in ways not previously possible offers more loadable data formats to users.

Other benefits include the potential reduction of hardware and developer resources. Additional disk space is not required to first uncompress the input data before it is read by the external table. For users with very large compressed input data and a shortage of available disk space this represents huge savings in terms of both time and resources. Being able to reformat the input data through a preprocessor program, rather than manually, also saves administrative resources.

References

For more information, see the following Oracle Database 11*g* Release 2 documentation:
Oracle Database Utilities for more information about the ORACLE_LOADER access driver.
Oracle Database Security Guide for guidelines about securing the ORACLE_LOADER Access Driver.
Oracle Database Administrator's Guide for detailed information about creating and managing external tables.

Using the Preprocessor Feature with External Tables in Oracle Database 11*g Release 2*
Author: Michael Sakayeda]
Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together