

Accessing Remote Data Files Using Oracle External Tables

Problem Definition: Typically external tables are created with data stored on the local machine, the problem at hand is to provide a mechanism to the users that enable them to access data files stored on a remote machine via external tables.

Overview of Solution:

The basic idea is to use the preprocessor directive of external tables to ssh into the remote machine and use the local data file specified in the access parameters of the external table as a proxy for the remote data file.

Background:

Generally, when users have to access remote data they would move the data files to some kind of distributed file system (example: NFS) and access this data as if it were local to the machine. One constraint imposed by this mechanism is that the user needs to have sufficient privileges to move the data and store it in the network file system. The mechanism provided below does not require the user to move the data or even have a network file system. However, ssh will need to be set up to allow the OS user which oracle executes as, to access the remote hosts(s).

Instructions:

Setting up 'ssh':

Since we will be using a shell script to ssh into the remote machine, we would like to perform ssh without using a password. This requires the user to generate the key file on the local machine and copy it to the remote machine so that the remote machine identifies our local machine as a trusted known host. Note that the external table preprocessor script executes with the OS credentials of the oracle shadow process. The private key file (.ssh/id_rsa) needs to reside in the HOME directory of the user which Oracle executes under (typically Oracle). The public key file (.ssh/id_rsa.pub) needs to reside in the HOME directories of the user on the remote machine.

The following commands will help you achieve this.

Generating the keyfile:

```
$ ssh-keygen -t rsa
```

After you enter the above command, you should see something like this:

Generating public/private rsa key pair.

```
Enter file in which to save the key (/home/oracle/.ssh/id_rsa):
```

Just hit Enter there. It will then ask you for a pass phrase; just hit enter twice. Here's what the results should look like:

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oracle/.ssh/id_rsa.
Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.
The key fingerprint is:
a6:5c:c3:eb:18:94:0b:06:a1:a6:29:58:fa:80:0a:bc oracle@localhost
```

This would create two files on your system (keypair) ~/.ssh/id_rsa and ~/.ssh/id_rsa.pub.

Now we need to copy the *.pub file to the HOME directory of the user on the remote machine that we want to access data from via ssh.

```
$ ssh remote_machine "mkdir .ssh; chmod 0700 .ssh"
$ scp .ssh/id_rsa.pub remote_machine:~/.ssh/authorized_keys2
```

Once the file is copied to the remote system you should be able to ssh into the remote system without requiring a password.

Setting up local machine:

Now we would need to setup our local machines to have oracle directories and the data files that would act as a proxy to fetch data from the remote file. Note that local files are necessary as a proxy since the external table access driver checks for and requires that the files exist.

Creating Oracle directories.

Assume that our directory structure for the local machine is the following

```
'/scratch/ora_xtra/data': directory where the proxy data file will reside
'/scratch/ora_xtra/log':  directory where the log files will reside
'/scratch/ora_xtra/exec': directory where the preprocessor script will reside
```

```
$sqlplus / as sysdba
```

```
SQL> CREATE OR REPLACE DIRECTORY data_dir AS '/scratch/ora_xtra/data';
```

```
Directory created.
```

```
SQL>GRANT READ on DIRECTORY data_dir to SCOTT;
```

```
Grant succeeded.
```

```
SQL> CREATE OR REPLACE DIRECTORY log_dir AS '/scratch/ora_xtra/log'
```

Directory created.

```
SQL>GRANT READ, WRITE on DIRECTORY log_dir to SCOTT;
```

Grant succeeded.

```
SQL> CREATE OR REPLACE DIRECTORY exec_dir AS '/scratch/ora_xtra/exec';
```

Directory created.

```
SQL>GRANT READ, EXECUTE on DIRECTORY exec_dir to SCOTT;
```

Grant succeeded.

Before we can query the external table we must create a Map file. This is the file that helps the preprocessor to identify the host name, user to connect to the remote machine, remote directory and the remote command to be executed on the file. The format of the file is

file:rusr:rhost:rdir:rcmd

wher,

file: name of the local proxy file.

rusr: name of the remote user used to ssh into the remote system.

rhost: name of the remote host where actual data file is stored.

rdir: name of the directory where the data file resides on the remote machine (excluding file name).

rcmd: command to be executed on the file mentioned (example 'cat', 'zcat').

Example

1. foo.dat:abrumm:abrumm-dev:/home/abrumm/xt_data:cat

2. bar.dat:zshuntu:zshuntu-dev:/home/zshuntu/xt_data:zcat

Note: The map file should be stored in the data directory in the same location where the local proxy data file is stored. Name of the file must be "ora_xtra_map.txt".

Note: The remote directory path is completely independent of how the local oracle directory is created. The only constraint is that the name of the data file (proxy file) created on the local machine is the same on the remote machine.

In this example, two files are accessed remotely

1. foo.dat on host abrumm-dev via cat

2. bar.dat on host zhuntu-dev via zcat.

In each case a different user is used to connect to a different host. Since we are connecting to two different hosts it would be required to have the rsa.pub file in the HOME directory of both the users of the respective host machines, i.e. two separate scp's need to be executed as follows.

```
$ scp .ssh/id_rsa.pub abrumm-dev:~/.ssh/authorized_keys2
$ scp .ssh/id_rsa.pub zshuntu-dev:~/.ssh/authorized_keys2
```

Once the public key files are in place and the map file is created querying the external table will fetch data from the remote files.

Note: Once the public key (rsa.pub) is copied to the remote system it might be a good idea to ssh to the remote host from terminal to ensure that ssh works fine without requiring the user to enter a password. If you still require to enter a password make sure the permissions on the .ssh folder are correct. Being able to ssh into the remote system without requiring a password is critical for this preprocessor to function.

Appendix:

External Table DDL:

```
create table xtab (id number,name varchar2(2000))
organization external(
type oracle_loader
default directory datadir
access parameters(
records delimited by newline
PREPROCESSOR execdir:'ora_xtra_pp.sh'
fields terminated by ','
missing field values are null (id , name))
LOCATION('foo.dat'))
reject limit unlimited
;
```

Preprocessor Script:

```
#!/bin/sh
PATH=/bin:/usr/bin export PATH

# ora_xtra_pp: ORAcle eXternal Table Remote file Access Pre- Processor

# Format for the Map File
# Consists of five fields separated using a :
# Syntax
# file:rusr:rhost:rdir:rcmd
#
# Example
# foo.dat:abrumm:abrumm-dev:/home/abrumm/xt_data:cat
# get filename component of LOCATION, the access driver
# provides the LOCATION as the first argument to the preprocessor

proxy_file_name=`basename $1`
data_dir_name=`dirname $1`
```

**# Flag is set if the file name in the Map file matches the proxy file name
where our data file is stored.**

flag_dirf=0

**# loops through the map file and fetches details for ssh
username, hostname and remote directory
file_not_found_err='ora_xtra_pp:
Map file missing. Needed ora_xtra_map.txt in data directory.'**

```
if [ -e $data_dir_name/ora_xtra_map.txt ]  
then  
  while read line  
  do  
    map_file_name=`echo $line | cut -d : -f1`  
    if [ $map_file_name = $proxy_file_name ]  
    then  
      rdir=`echo $line | cut -d : -f4`  
      rusr=`echo $line | cut -d : -f2`  
      rhost=`echo $line | cut -d : -f3`  
      rcmd=`echo $line | cut -d : -f5`  
      flag_dirf=1  
      break  
    fi  
  done <$data_dir_name/ora_xtra_map.txt  
else  
  echo $file_not_found_err 1>&2  
  echo $data_dir_name 1>&2  
  exit 1  
fi  
  
if [ $flag_dirf = 1 ]  
then  
  ssh -q $rusr@$rhost $rcmd $rdir/$proxy_file_name  
fi
```