CONF-9409368--1

Utilizing ORACLE tools within Unix

Ray Ferguson
Science Applications International Corporation
wp5.2
abstract # 136

Introduction

Large databases, by their very nature, often serve as repositories of data which may be needed by other systems. The transmission of this data to other systems has in the past involved several layers of human intervention. The Integrated Cargo Data Base (ICDB) developed by Martin Marietta Energy Systems for the Military Traffic Management Command as part of the Worldwide Port System provides data integration and worldwide tracking of cargo that passes through common-user ocean cargo ports. One of the key functions of ICDB is data distribution of a variety of data files to a number of other systems. Development of automated data distribution procedures had to deal with the following constraints:

1. variable generation time for data files,
2. use of only current data for data files,
3. use of a minimum number of select statements,
4. creation of unique data files for multiple recipients,
5. automatic transmission of data files to recipients, and
6. avoidance of extensive and long-term data storage.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Variable Generation Time

Procedures for the extraction, generation, and transmission of data files to recipients had to be designed to be executed at variable intervals. The normal interval for execution is daily, but a means was necessary to allow execution at other periodic intervals. The Unix CRON utility is used to execute Unix script files at specified times. Changes to the CRON entry can be accomplished by use of an ORACLE SQL*Forms screen. This screen reads and displays records from a table, crontbl, with fields corresponding to the CRON entry. Users may change the frequency of execution of data distribution procedures by changing screen values. Committing these changes updates the crontbl table and executes the 'host' command. The host command executes a Unix script file which selects the updated records from the crontbl table into a flat file. The Unix crontab command is then executed using the flat file as new values for the CRON entry. Use of a SQL*Forms screen to alter values in the CRON entry allows the user to change the frequency of execution of data distribution procedures and to execute data distribution procedures at period of low system use.

Use Of Only Current Data

The ICDB journal table is updated each hour with transactions which have occurred against tables in the ICDB. Data to be included in data files for data distribution comes from the journal table as well as other ICDB tables. As journal records are received by ICDB, a record is inserted into a pointer table, pointing to a unique record in the journal table. Both the journal record and the pointer record contain a date field indicating the date the journal record was received by ICDB. This date is an ORACLE sysdate value used in determination of length of time data is to be kept in the journal and pointer tables. Since transaction data referenced by a journal record may be included in a variety of data files for a variety of recipients, other flag fields are included in the pointer table to indicate whether the transaction data has been included in particular data files. The flag fields are necessary

since some systems may wish to receive data at intervals other than daily. The flag fields identify journal records which have been included in particular data files and prevent these data items from being duplicated during future generation of data files for the same receiving system.

Minimum Number of Select Statements

Each instance of execution of data distribution procedures is via a Unix script file whose execution is controlled by the Unix CRON utility. Data files to be generated contain data from several ICDB tables. Since there are several types of data files to be created with each type containing its own unique data set, a generic table, datatbl, is used to contain all data elements needed for all the files to be generated. One function of the Unix script file is the execution of an ORACLE stored procedure to select data into the datatbl. Using one stored procedure to select all the data needed for all data files reduces the number of select statements with joins against multiple tables to a minimum. Syntax within the Unix file needed to execute the stored procedure for data selection is as follows:

```
sqlplus -s username/password << end > /dev/null
truncate table datatbl
/
exec select_procedure ($parameter1 $parameter2 ...)
/
end
```

This code will initiate a SQL*Plus session for the user and execute SQL*Plus statements until the word 'end' is encountered. In this example, the datatbl is truncated, disposing of any records used in generation of previous data files, and new data is selected by execution of the stored procedure. The stored procedure uses the pointer tables described previously to determine which journal records may be used in current data files. Using these pointer records, data is selected from ICDB tables into

the datatbl table. Execution of the stored procedure from within the Unix script file allows population of the datatbl table with a wide range of values by using ORACLE'S PL/SQL procedural language for data selection and manipulation.

Unique Data Files for Multiple Recipients

Each recipient receives a unique data file consisting of one or more sets of data selected on the basis of predefined selection criteria. Table recip_address contains fields for the recipient name, the type of data file to be received, and the electronic address to which the data file is to be transmitted. The following syntax within the Unix script file will select records from the recip_address table into a previously defined flat file, $recip_address, where the type of data file to be received is 'DAILY'.

```
1       while read line
2       do
3         count=1
4     for word in $line
5         do
6         case $count in
7         1)
8         recipient=$word;
9         count=2;
10        ;;
11        2)
12      file_type=$word;
13        count=3;
14        ;;
15        3)
16        address=$word;
17        count=4;
18        ;;
19        esac
20        done
21      $datafile=$datafile.$recipient
22      gen_data_file $recipient $datafile
23    rcp $datafile $address > /dev/null 2>&1
24      find /home/storage -name "*.$recipient" -ctime +7 exec       rm          {} /;
25    done
```

Lines 1-20 select a row from the $recip_address file. Line 21 assigns the recipient's name as the extension to a previously defined temporary file, $datafile, created at the beginning of the script file. The Unix script file, gen_data_file, described in the following code section, executes code to select the appropriate data for the particular recipient. Line 23 uses the Unix 'rcp' utility to electronically transmit the data file to the recipient's address as determined by the $address variable. Line 24, the Unix find command, is discussed in the Data Storage section of this paper. Syntax for the gen_data_files script file is shown below.

```
if [ $1 = "JCCO" ]; then
  select_data_type1.rpt $2
  select_data_type5.rpt $2
elsif [ $1 = "ILC" ]; then
  select_data_type2.rpt $2
  select_data_type3.rpt $2
.
.
.
fi
```

The gen_data_files script file executes a series of if/elsif statements, and based on the recipient value passed as parameter one, executes a unique Unix script file to select data for the recipient. Syntax for a sample .rpt file is shown below.

```
sqlplus -s username/password << end >> $1
set ... ( SQL*Plus set commands to set SQL*Plus environment)
select ... (Select appropriate data)
/
update ... (Update appropriate pointer table flags)
/
end
```

Each .rpt file will execute a select statement or statements to select the appropriate data from the datatbl for the recipient. This data will be appended to the data file of the recipient as specified by

parameter one. After data has been selected, the update command updates flags in the pointer tables for those records which have been used in the type of data file generated, preventing inclusion of this data in future data files of the same type for the same recipient.

Data Storage

Numerous data files are generated each time the data distribution procedures are executed. Over a period of time, data storage will become a problem without a means of limiting the period of time data files are kept. The Unix find command shown in previous code will find and remove those data files whose generation date is seven or more days prior to the current date.

Summary

This paper has presented some examples of how ORACLE tools may be used in conjunction with Unix script files to automate data distribution procedures. The use of the Unix CRON utility allows the execution of Unix script files at user-selected intervals. The ability to execute SQL*Plus commands and stored procedures utilizing PL/SQL procedural language from within a Unix script file provides the ability to automatically update tables, execute complex select statements and perform data manipulations necessary for automation of the data distribution process. Use of the Unix 'rcp' utility provides for the automatic transmission of data to other systems and the Unix 'find' and 'rm' commands prevent excessive data storage. The ability to utilize ORACLE tools within a Unix script file provides the user with many avenues toward process automation.