Generic Application Data Extraction Process

Find out how to run your application on your desktop.

By Mudassar Shahzad Choudhry (mudassarhanif@hotmail.com)

Application Databases in Production environments are hitting 100’s of GBs these days. Despite the fact that storage has become cheap, still these huge databases can only be maintained on huge storages. In such circumstances it becomes very difficult for anyone to have a local application environment on his/her desktop. Having application data on your own local desktop has become a necessity and helps address the following problems:

- Testing – Having a local desktop environment is very helpful for testing your application at your own convenience.
- Development – Development often requires application data accessibility at your own expediency.
- Training – It is often not practical to have training environments on databases sized in 100’s of GB. Mostly it is required to maintain and have a subset of the data.

If you are a DBA or a Developer who has been asked to extract a subset of your application data for testing, development or training environments, then this article is definitely going to solve your problem. It shall provide you with a ready made utility that you can use to achieve this purpose. The start_export_tables.ksh shell script shall address your concerns in this very regard and shall provide you with a flexible and powerful way to extract your application data based on specific requirements for any target environment. The script is fully dynamic and requires a one-time configuration setup for your specific application and then you are ready to extract the application specific data. Now, you don’t need to spend time extracting specific application data every time it is required. Your one time setup is the only time consuming activity. This is the first beta release of the script.

Background / Overview

We have a custom made telecom application solution named TABS that used to vary from site to site. Having different solutions made it very difficult to extract specific and different data for each installation. To address this very problem, I developed the start_export_tables.ksh script to extract specific data from every site. The start_export_tables.ksh script assumes certain preliminary setup, which shall be discussed in the section here-under.

Step 1: Preliminary Setup

Before using the start_export_tables.ksh, please make sure to download the zip file that is listed under the 'Prerequisite' heading in the note (392695.1) itself. Put the extracted files into a UNIX directory. The package contains the following files:

- start_export_tables.ksh – Main Executable Shell Script
- vartemplate.cfg – Variables Definition Template file, can also be used as a sample variable definition file
- stor_par.cfg – Storage Parameters Modification file used for modifying the export script that is generated. (Courtesy: Oracle MetaLink)
- exp_params.cfg – Export Parameters file
- muls.cfg – File used for defining multiple variable values for exporting data based on more than one criterion. The variables should be present in the same sequence as the variable template file.
- bigtables.cfg – File used to define the big tables. File also can be used to define whether the bigtables should be exported or not and the export condition for the big tables. The export condition uses the same variables as defined in the templates file.

You need to have a DBA or SYSDBA user for the database to be able to use the scripts.

Step 2: Script Usage

Let us begin by using the help of the script:

Running the following command will show you the help of the script on the screen:

```
home@muds[/mudather/t3extract]# start_export_tables.ksh –h
```

Business Case 1:

Referring to our Telecom Application example, the requirement is to export the data for a specific subscriber from different application tables. The first step is to create a file having the list of variables to be used for the purpose of the export. Below is an example of the variables files. The first field is the variable name, the field separator is the ~ character and the third field is the variable description. The second field is not currently used. So, basically, for exporting the data for my application, I need to have the following variables which have a meaning in my application.

```
home@muds[/mudather/t3extract]# more vartemplate.cfg

V_CONTRNO~S~Contract Number
V_SUBNO~S~Subscriber Number
V_USER~S~Logname/username for selecting data by using this field
V_DATE1~S~Date used for exporting data from some tables, it should be present in
the NLS_DATE_FORMAT for the session
V_ITEM~S~Item code to be used for exporting data
```

After the variables are defined in the file, the next step is to create a file with the list of tables to be exported and their export condition. For example, I have a table named tabs.ADM_DEPOSIT_HIST and I need to export the data from this table for a specific subscriber. So I will prepare a file with this table name and the export condition using the variables that were declared in the previous file. Below is a subset example of this file. Field one is the table name to be exported which MUST be prefixed by the table owner. Field two is a flag indicating whether the table is to be exported or not. Y indicates that the table should be exported and N would indicate not to export the table. Field 3 which starts at position 300 is the export condition for the table. As noticeable the export condition uses the variables declared in the variable declaration file. For e.g. for the first table the data should be exported for a specific subscriber. It is important to note the use of the escape character "\" as this is required three times due to the shell expansion.

```
home@muds[/mudather/t3extract]# more bigables.cfg
tabs.ADM_DEPOSIT_HIST~Y~



                                    \\\"where subno=\\\'$V_SUBNO\\\'\\\"

tabs.ADM_LOGQ04011~Y~



                                    \\\"where subno=\\\'$V_SUBNO\\\'\\\"

tabs.BILL_CONTROL~Y~



                                    \\\"where contrno=\\\'$V_CONTRNO\\\'\\\"
```

Now that we have identified the tables to be exported with their export conditions, the next step is to specify export parameters in an export configuration file. As noticeable in the file, the export parameters are Oracle Standard export parameters. It is worth noting that the Oracle Parameter direct must be set to "N" for the usage of where condition during export.

```
home@muds[/mudather/t3extract]# more exp_params.cfg
```

```
direct=n
buffer=10485760
compress=n
```

We are now all set to run the program to export data for specific subscribers from specific tables. Let us now proceed with running the program. As noticeable below, the script prompts for the username and password to connect to the database to generate the export scripts. This user must be a DBA or a SYSDBA. The usage for both is identified. In our case, we are using the "/" string to connect to the database. The next question is the Oracle database owners for all the schemas from which the tables are to be exported and listed in the bigtables.cfg file. This parameter has another significance which will be discussed later in the document. In our case the Oracle database owners are "tabs,inven" (which is TABS and INVEN). The third question is about the output directory for the export scripts and export data. This can be left null and the script will generate a directory in the current directory with a name starting from "export_" followed by the data and time of the script run.

```
home@muds[/mudather/t3extract]# start_export_tables.ksh
Start Time: 20060306145447
Please enter the Username and password to use for running export command, should be
DBA, for SYSDBA usage, use
                              \\\'sys/password as sysdba \\\'
                              or
                              \\\'/ as sysdba \\\'
/
Please enter the owners of database objects separated by commas. Multiple values
MUST only be separated by commas
tabs,inven
Please enter the output directory for the program
```

After supplying the above values, the script will display a menu similar to the following. Based on our original requirement, we shall select Option 3 from the menu and press Return to continue.

```
                          DEP MENU
          1 - Full Database Structure Export Only
          2 - All Tables Export with Data Except Tables in Bigtables configuration file
          3 - Export of Tables in Bigtables configuration file only
          4 - Option 1, 2 and 3 together
          5 - Export of Tables in Bigtables configuration file for multiple subscribers
          6 - Option 1, 2 and 5 together
          99 - EXIT
```

After selecting the option, the script will prompt us for the values of the variables declared in the variable declaration file. As noticeable below the digit "3" represents the option selected. The red colored text is the output of the prompts and represents the data that I need to be exported.

```
3
Only one value per variable allowed, unless otherwise specified
Please enter the value for variable : V_CONTRNO-S
Field description: Contract Number
070222
Please enter the value for variable : V_SUBNO-S
Field description: Subscriber Number
9880742
Please enter the value for variable : V_USER-S
Field description: Logname/username for selecting data by using this field
MUDASSAR
Please enter the value for variable : V_DATE1-S
Field description: Date used for exporting data from some tables, it should be present in
the NLS_DATE_FORMAT for the session
31-JUL-05
Please enter the value for variable : V_ITEM-S
Field description: Item code to be used for exporting data
SC28
```

After supplying the above parameter values and pressing return to continue, the script would start the export generation and export process. As noticeable in the output below the three tables were exported and the export file and the log file created are identified.

```
Processing Table: tabs.ADM_DEPOSIT_HIST with Query field
Processing Table: tabs.ADM_LOGQ04011 with Query field
Processing Table: tabs.BILL_CONTROL with Query field
Big Tables Export File with data created:
./exports_20060306145447/genbigtablesexport_20060306145447_1.ksh
Big Tables Export File called in nohup:
./exports_20060306145447/genbigtablesexport_20060306145447_1.ksh
Please check the log file for big tables export in:
genbigtablesexport_20060306145447_1.log
```

Below is a snapshot output of the Big tables export file that was created. Multiple lines are wrapped due to line size in the output below, but it is visible from the below output that the three tables export script was created using the export parameters that we specified and the export conditions with substituted variables that we specified.

```
exp userid=/ tables=tabs.ADM_DEPOSIT_HIST
file=tabs.ADM_DEPOSIT_HIST_20060306145447_18_6.dmp
log=tabs.ADM_DEPOSIT_HIST_20060306145447_18_6.log direct=n
buffer=10485760 compress=n query= \"where subno=\9880742\'\"

exp userid=/ tables=tabs.ADM_LOGQ04011
file=tabs.ADM_LOGQ04011_20060306145447_19_6.dmp
log=tabs.ADM_LOGQ04011_20060306145447_19_6.log direct=n buffer=10485760
compress=n query= \"where subno=\9880742\'\"

exp userid=/ tables=tabs.BILL_CONTROL
file=tabs.BILL_CONTROL_20060306145447_20_6.dmp
log=tabs.BILL_CONTROL_20060306145447_20_6.log direct=n buffer=10485760
compress=n query= \"where contrno=\'070222\'\"
```

After the scripts are generated, they are also called in nohup and we can find the output dump files created. The big tables dump files are also compressed/zip at the end incase you want to copy the compressed/zipped version to another machine. It is worth noting that the script also creates import scripts for you for Windows/UNIX for the purpose of importing the dump files on the target machine. Below is the snapshot of the file created for UNIX: As noticeable the script is first setting the NLS_LANG based on the Source database which is followed by import statements for the individual dumps that were created.

```
home@muds[/mudather/t3extract]# more imp_genbigtablesexport_20060306145447_6.ksh

export NLS_LANG=AMERICAN_AMERICA.AR8ISO8859P6

imp file=tabs.ADM_DEPOSIT_HIST_20060306145447_18_6.dmp fromuser=tabs touser=tabs ignore=y log=imp_tabs.ADM_DEPOSIT_HIST_20060306145447_18_6.log

imp file=tabs.ADM_LOGQ04011_20060306145447_19_6.dmp fromuser=tabs touser=tabs ignore=y log=imp_tabs.ADM_LOGQ04011_20060306145447_19_6.log

imp file=tabs.BILL_CONTROL_20060306145447_20_6.dmp fromuser=tabs touser=tabs ignore=y log=imp_tabs.BILL_CONTROL_20060306145447_20_6.log
```

Business Case 2:

The previous example demonstrated the usage of exporting specific tables based on the where clauses. Let us build on that example and see the full picture and usage of the script. Referring again to our telecom application, the requirement now is as follows. The Production database is sized at 100GB and I want to have the subset of the entire application database extracted so that I can run the application on my local system without the need to have the full data. For the purpose of the data extraction process, I have categorized application tables into two categories:

1. Small Tables (or Setup Tables)

2. Big Tables (including Transaction, Historical, Log Tables etc.)

This division is a soft division, meaning that it is up to the user of this script to identify small tables and big tables. The way the script would work is that it would export the small tables directly without any export condition and the big tables would be exported using the export condition (similar to our previous example). The variable templates, export parameters and the bigtables configuration file can be defined similar to our first example. However, in this case the bigtables configuration file would have all the list of bigtables in the application schemas. The bigtables configuration file can also be used to identify the tables that should not be exported by using the "N" flag. If you recall from our previous example, the script prompted us for the application schema owners. The script will use these schemas and take all tables in the identified schemas EXCEPT the tables in the big tables configuration file and export them without any condition. These tables are considered to be the small tables. We would select Option 4 from the menu and after supplying the variable values, we would get a similar output. The file prefixed with "gendbexplist" will be the export script for exporting small (setup) tables. The file prefixed with "genfullstructureexport" will be the export script to export the full database structure. All the files created are also called in nohup as evident from the output below.

```
All Table Export File with data created:
./exports_20060306145447/gendbexplist_20060306145447.ksh
Full Table Structure export file created:
./exports_20060306145447/genfullstructureexport_20060306145447.ksh
Processing Table: tabs.ADM_DEPOSIT_HIST with Query Field
Processing Table: tabs.ADM_LOGQ04011 with Query Field
Processing Table: tabs.BILL_CONTROL with Query Field
Big Tables Export File with data created:
./exports_20060306145447/genbigtablesexport_20060306145447_7.ksh
Big Tables export file called in nohup:
./exports_20060306145447/genbigtablesexport_20060306145447_7.ksh
```

```
Please check the log file for big tables export in: genbigtablesexport_20060306145447_7.log
Full Table Structure export file called in nohup: ./exports_20060306145447/genfullstructure_20060306145447.ksh
Please check the log file for full table structure export in : genfullstructureexport_20060306145447.log
All Table Export File with data called in nohup: ./exports_20060306145447/gendbexplist_20060306145447.ksh
Please check the log file for full table export in: gendbexplist_20060306145447.log
```

After the above output is displayed on the screen, the script will also display the steps to import the data on the target environment. Let us have a look at the additional scripts that were created due to the Option selected. Following is the output of the script generated and run to export small tables. This script is exporting the tables using the parameter file identified.

```
home@muds[/mudather/t3extract]# more gendbexplist_20060306145447.ksh
exp parfile=gendbexplist_parfile_20060306145447.txt
zip -9r t3_all_20060306145447.zip t3_all_20060306145447.dmp
```

Below is the output of the parameter file (shortened output). As noticeable this parameter file will export all tables in the two schemas (TABS, INVEN) identified earlier except the tables in the Big Tables Configuration file. The export parameters are also listed.

```
home@muds[/mudather/t3extract]# more gendbexplist_parfile_20060306145447.txt
tables=( TABS.ADM_ADDR_DETAIL,
TABS.ADM_AGE_DISCOUNT,
TABS.ADM_BESTA0401,
TABS.ADM_BESTA0402,
INVEN.INV_RENAME_ITEMS )
file=t3_all_20060306145447.dmp
log=t3_all_20060306145447.log
userid=/
direct=n
buffer=10485760
compress=n
```

Before we look at the next export script that is created, it is worth mentioning about the Storage Parameters configuration file namely stor_par.cfg (Source: http://metalink.oracle.com). Whenever an Oracle Export is created and it needs to be imported on any target environment, it uses the Source Database Storage Parameters such as Tablespace Name, Extent Sizes etc. This parameter file can be used to configure and identify the storage parameters that you need to get rid of. Our Storage Parameter file, listed in the appendix is used to remove the following parameters from the Import Scripts, INITIAL, NEXT and TABLESPACE. As a result of using this file, the full structure export file is created as follows. The first thing this script is doing is to export the structure of the full database. This script is then using the structure dump to generate the create table and create indexes script (for each schema) using the storage parameters configuration file. This step is looped for all the schemas identified in the beginning. Besides, the script is also manipulating the synonyms and the roles.

```
home@muds[/mudather/t3extract]#   cat   genfullstructureexport_20060306145447.ksh
#!/bin/ksh
exp file=t3_struct_20060306145447.dmp log=t3_struct_20060306145447.log userid=/
full=y rows=n direct=n buffer=10485760 compress=n

imp    userid=/    file=t3_struct_20060306145447.dmp    fromuser=tabs    touser=tabs
indexfile=t3_struct_tabs_20060306145447.sql

cat - <<EOF|ed -s t3_struct_tabs_20060306145447.sql
1,$ g/REM /s/REM //g
1,$ g/INITIAL *$/j
1,$ g/INITIAL *[0-9]* /s/INITIAL *[0-9]* /INITIAL 8K /g
1,$ g/NEXT *$/j
1,$ g/NEXT *[0-9]* /s/NEXT *[0-9]* /NEXT 8K /g
1,$ g/TABLESPACE *$/j
1,$ g/TABLESPACE \"*[A-Z0-9]*\" /s/TABLESPACE \"*[A-Z0-9]*\"//g
1,$ g/TABLESPACE  \"*[A-Z0-9]*\" /s/TABLESPACE  \"*[A-Z0-9]*\"//g
1,$ g/TABLESPACE \"*[A-Z]*\" /s/TABLESPACE \"*[A-Z]*\"//g
1,$ g/TABLESPACE \"*[A-Z_]*\" /s/TABLESPACE \"*[A-Z_]*\"//g
1,$ g/^CONNECT /d
w Imp1_tabs_20060306145447.sql
q
EOF

strings -a t3_struct_20060306145447.dmp | grep ^"CREATE PUBLIC SYNONYM" |
grep -i "FOR \"tabs\"\." >> Imp1_20060306145447.sql

imp    userid=/    file=t3_struct_20060306145447.dmp    fromuser=inven    touser=inven
indexfile=t3_struct_inven_20060306145447.sql

cat - <<EOF|ed -s t3_struct_inven_20060306145447.sql
1,$ g/REM /s/REM //g
1,$ g/INITIAL *$/j
1,$ g/INITIAL *[0-9]* /s/INITIAL *[0-9]* /INITIAL 8K /g
1,$ g/NEXT *$/j
```

```
1,$ g/NEXT *[0-9]* /s/NEXT *[0-9]* /NEXT 8K /g
1,$ g/TABLESPACE *$/j
1,$ g/TABLESPACE \"*[A-Z0-9]*\" /s/TABLESPACE \"*[A-Z0-9]*\"//g
1,$ g/TABLESPACE  \"*[A-Z0-9]*\" /s/TABLESPACE  \"*[A-Z0-9]*\"//g
1,$ g/TABLESPACE \"*[A-Z]*\" /s/TABLESPACE \"*[A-Z]*\"//g
1,$ g/TABLESPACE \"*[A-Z_]*\" /s/TABLESPACE \"*[A-Z_]*\"//g

1,$ g/^CONNECT /d
w Imp1_inven_20060306145447.sql
q
EOF

strings -a t3_struct_20060306145447.dmp | grep ^"CREATE PUBLIC SYNONYM" |
grep -i "FOR \"inven\"\." >> Imp1_20060306145447.sql

strings -a t3_struct_20060306145447.dmp | grep ^"CREATE ROLE" >>
Imp1_20060306145447.sql

mv Imp1_20060306145447.sql Imp1_20060306145447.tmp

sed s/"CREATE PUBLIC SYN"/"CREATE OR REPLACE PUBLIC SYN"/g
Imp1_20060306145447.tmp > Imp1_20060306145447.sql

mv Imp1_20060306145447.sql Imp1_20060306145447.tmp
sed  s/$/\;/g Imp1_20060306145447.tmp > Imp1_20060306145447.sql
rm -f Imp1_20060306145447.tmp
zip -9r t3_struct_20060306145447.zip t3_struct_20060306145447.dmp
```

Besides, the above set of files, there are some other files also created to help in the purpose of importing the data above. Below script created to import the schemas (notice the NLS_LANG is identified). The script run sequence is mentioned in the help.

```
home@muds[/mudather/t3extract]# cat imp_gendbexplist_20060306145447.ksh
export NLS_LANG=AMERICAN_AMERICA.AR8ISO8859P6
imp    file=t3_all_20060306145447.dmp    log=imp_t3_all_20060306145447.log
fromuser=tabs touser=tabs ignore=y
imp    file=t3_all_20060306145447.dmp    log=imp_t3_all_20060306145447.log
fromuser=inven touser=inven ignore=y
```

Similarly, a full structure import script:

```
home@muds[/mudather/t3extract]#cat
imp_genfullstructureexport_20060306145447.ksh

export NLS_LANG=AMERICAN_AMERICA.AR8ISO8859P6
imp   file=t3_struct_20060306145447.dmp   fromuser=tabs   touser=tabs   ignore=y
log=imp_t3_struct_20060306145447.log
imp   file=t3_struct_20060306145447.dmp   fromuser=inven   touser=inven   ignore=y
log=imp_t3_struct_20060306145447.log
```

Following is the listing of some other scripts generated to aid in the import process. Scripts for table and index creation, disabling and enabling constraints, disabling and enabling triggers, create users, create public synonyms and create roles.

```
home@muds[/mudather/t3extract]#ls *sql

Imp0_20060306145447.sql                      Imp_discon_20060306174559_inven.sql
Imp_encon_20060306174559_inven.sql           t3_struct_inven_20060306145447.sql
Imp1_20060306145447.sql                       Imp_discon_20060306174559_tabs.sql
Imp_encon_20060306174559_tabs.sql     t3_struct_tabs_20060306145447.sql
Imp1_inven_20060306145447.sql                Imp_distrig_20060306174559_inven.sql
Imp_entrig_20060306174559_inven.sql
Imp1_tabs_20060306145447.sql                  Imp_distrig_20060306174559_tabs.sql
Imp_entrig_20060306174559_tabs.sql
```

Let us also see a snapshot of the table/index creation scripts for one schema. Notice that the tablespace name(s) and the storage parameters (initial, next) have been removed from the script so that you can import the tables in any target environment.

```
home@muds[/mudather/t3extract]# more Imp1_inven_20060306145447.sql
CREATE TABLE "INVEN"."ADM_VENDORS" ("V_CODE" VARCHAR2(10) NOT NULL ENABLE, "V_NAME" VARCHAR2(50) NOT NULL ENABLE, "AGREEMENT_TYPE" VARCHAR2(10) NOT NULL ENABLE, "V_DESC" VARCHAR2(50), "PHONE" VARCHAR2(20), "FAX" VARCHAR2(20), "CONTRACT_NO" VARCHAR2(10), "AR_NAME" VARCHAR2(50)) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 STORAGE(INITIAL 8K FREELISTS 1 FREELIST GROUPS 1) LOGGING NOCOMPRESS ;

CREATE UNIQUE INDEX "INVEN"."VENDORS_IDX_U1" ON "ADM_VENDORS" ("V_CODE" ) PCTFREE 10 INITRANS 2 MAXTRANS 255 STORAGE(INITIAL 8K FREELISTS 1 FREELIST GROUPS 1) LOGGING ;
```

Since the scripts are called in nohup, you can monitor the processes running to verify whether the extraction process is finished or not. After the export processes are finished, you can use the steps outlined in the help to import the data exported and your target environment will be ready. The steps are pasted from the help below:

For Menu Options 4 and 6, meant to create a subset of TABS database, do the following steps:

Note: The Date and Time in the filename will vary based on your program run On The Target Environment

1. Create a New Tablespace to Hold the Data
2. Run the Generated Create User Script to Create the Users.

File Name: Imp0_20060921130953.sql

3. Run the Generated Create Public Synonyms and the Create Roles Script.

File Name: Imp1_20060921130953.sql

4. Run the Generated Create table and Create Index Script.

Multiple files might be generated for different Users

File Name: Imp1_USER_20060921130953.sql

5. Run the Generated Full Database Structure Import Script.

File Name: imp_genfullstructureexport_20060921130953.bat OR imp_genfullstructureexport_20060921130953.ksh

6. Run the Generated Disable Constraints Script

Multiple files might be generated for different Users

File Name: Imp_discon_20060921130953_USER.sql

7. Run the Generated Disable Triggers Script

Multiple files might be generated for different Users

File Name: Imp_distrig_20060921130953_USER.sql

8. Run the Generated Script to Import All Small/Setup Tables

File Name: imp_gendbexplist_20060921130953.bat OR imp_gendbexplist_20060921130953.bat

9. Run the Generated Disable Constraints Script

Multiple files might be generated for different Users

File Name: Imp_discon_20060921130953_USER.sql

10. Run the Generated Disable Triggers Script

Multiple files might be generated for different Users

File Name: Imp_distrig_20060921130953_USER.sql

11. Run the Generated Script to Import Big Tables

The "n" in the filename represent a sequential number

You might have multiple files generated based on your options selected

File Name: imp_genbigtablesexport_20060921130953_n.bat OR imp_genbigtablesexport_20060921130953_n.ksh

12. Run the Generated Enable Constraints Script

Multiple files might be generated for different Users

File Name: Imp_encon_20060921130953_USER.sql

13. Run the Generated Enable Triggers Script

Multiple files might be generated for different Users

File Name: Imp_entrig_20060921130953_USER.sql

14. Run the $ORACLE_HOME/rdbms/admin/utlrp.sql script to recompile any invalid objects

```
For Windows use: %ORACLE_HOME%\rdbms\admin\utlrp.sql

Ready to Use the Target Application Environment
```

You can use the other options in the menu also to your taste to either export big tables only, database structure only or small application tables. For exporting data from big tables based on multiple parameter sets you can use the sample supplied configuration file as below. The file has parameter values separated by the ~ character and having multiple set of parameters in different lines. Once the file is prepared, you can use Option 5 or 6 to export data based on this criteria.

```
home@muds[/mudather/t3extract]# more muls.cfg
070222~9668401~MUDASSAR~31-JUL-2005~SC28
070222~9880742~MUDASSAR~31-JUL-2005~SC28
```

Comments

Following are some of the limitations of the script and some comments about it.

1. The User used for exporting MUST be DBA, there is no checking in the program.

2. The Big tables configuration file must have the table names prefixed by the owner, not doing so may result in abnormal behavior.

3. The Date/Time listed in every file created will be the date/time of the script run.

4. The source environment to run the script can be any UNIX Operating System with Korn Shell. The script however has been tested on Sun SPARC Solaris 8.

5. The target environment can also be any UNIX Operating System with Korn shell and any Windows Operating System. The script however has been tested on Microsoft Windows XP.

6. Feedback about the script is most welcome for the purpose of improving the script.

7. The script has been tested on Oracle Server Enterprise Edition 9.2.0.1, however it should run on any Oracle Version from 7.3 onwards till the latest version.

Conclusion

To conclude the article, the Generic Application Data Extraction Process is a very handy utility and shall form the foundation of future powerful utilities that shall expedite and enhance the data extraction process for the purpose of making the testing, training and development process much simpler and reusable.

About the Author

Mudassar Choudhry (mudassarhanif@hotmail.com) is a Senior Customer Service Engineer at International Turnkey Systems, Kuwait (http://www.its.ws) (The largest Systems Integrator in the Middle-East providing solutions for the Banking, Telecom and Higher Education industries).