How to automatically test and validate your database backup and recovery strategy

Ruben Gaspar Aparicio
CERN, IT department, Geneva 23, CH-1211 Geneva
ruben.gaspar.aparicio@cern.ch

Abstract. The major challenge we solve with this software project is the automated validation of backups sent to tape for Oracle databases. While Oracle Recovery Manager (RMAN) provides tools like ‘restore validate’, the real and only certain proof is a restore. This initial aim evolved to provide a recovery platform capable to cover more complex user cases, such as validations of backup strategy of Very Large DataBases (VLDB), and schema recoveries to cure logical errors or to provide the kind of database snapshots by means of exports.

1. Introduction

It is very important to be able to recover an Oracle database to a working state after an incident. Though Oracle server provides some tools, the ultimate reliable proof is to perform the recovery on a different database server. At CERN, we have developed a recovery platform to satisfy the needs for backup validation as well as for helping the database administrators perform other actions.

The platform is designed to perform restores of different Oracle database server versions, which are so far 9i, 10g and 11g, on both Solaris and Linux. While 32 bits recoveries have been only tested on Solaris, the platform independent code (Perl and bash scripting) should allow extrapolating it to Linux 32 bits. The platform has usually been run against databases running Red Hat Enterprise Linux (RHEL) 64 bits.

Though the code has been developed to work with IBM Tivoli libraries (TSM) [7], RMAN [2] provides an abstraction to the tape storage used underneath. A different channel configuration may be required in case different tape solutions are used. This would require minimal changes in the code.

The platform would effectively help Database Administrators (DBA’s) to perform the following tasks:
- Testing validity of information sent to tape;
- Validation of a backup strategy;
- Partial restore: just a single schema from a database;
- Partial restore of Very Large DB, notably those having a lot of space used by read-only tablespaces;
- Keeping a copy of last successful restore scripts;
- After a successful recovery an export of the database using either “exp” or “expdp” utilities [3] can be taken. The export can be configured to just extract certain schemas, per user, or extract the whole database, a full export. Later, it can be stored on tape and/or offsite file server;
- Total isolation from your production system;
- Speed-up logical error correction on VLDB;
- Depending on configuration it can be used as backup to disk;

Published under licence by IOP Publishing Ltd
- Generating a complete set of recovery scripts considering actual status of the database.

The recovery platform code is provided under Apache License version 2.0, copyright 2010 CERN, and can be downloaded from the SourceForge web site [1].

2. Description of the platform

The main software components can be seen in the following components diagram - Figure 1 together with its physical location on the file system, Figure 2:

![Components diagram](image1)

![Directory arborescence](image2)

The start-up of the application is a bash script wrapper (recovery_wrapper.sh) around the main Perl script: recexe.pl. The environment desired for the recovery is set in this wrapper (i.e. 9i,10g or 11g database). The wrapper also logs results into an external web application. At the same time it allows to query the recovery in order to check how much work has been completed (\textit{-p} option) or remains to be done (\textit{-r} option), or it simply shows what it would do (\textit{-t} option).

The wrapper depends on the CERN configurations and it is provided as an example of implementation.

The directory arborescence shows the locations of all software components on the recovery server. \textsc{TOP\_DIR} is nowadays fixed to `/ORA/dbs01/syscontrol/projects/recovery`, though it can be changed if necessary, which will require modifying parts of the code in order to locate needed Perl modules. Therefore, it is recommended to keep it in a first approach. All components shown in figure 1 are located under the `bin` directory. The Perl modules implement all core functions. This approach makes the whole platform flexible and easy to adapt to new environments, changing in commands due to new Oracle servers releases, etc. Thus, four Perl modules were developed:

- \texttt{Recovery.pm}: core functionality of the platform: creating files, checking OMF specs, time formatting, etc.
- \texttt{Logger.pm}: it logs any action performed by the platform.
- \texttt{DBPerl.pm}: it is being used since partial recoveries have been introduced. Basic functionality is to connect and run queries using DBD::Oracle package.
• General.pm: global variables used by the platform. All variables contained in a configuration recovery file are exported to all Perl modules. To note that $dirtobackup is a variable that indicates where scripts used by a successful recovery will be placed for future use.

Under the “etc” directory we find the definition files for a recovery, one per database. It is an important file where all parameters of a recovery should be given. A very short summary of the representative group of parameters by function is as follows:

• Storage used for the restore database is either Automatic Storage Management (ASM) [6] or file system. Following parameters defined each type of storage. $asminstance: oracle sid for ASM, $asmoraclehome, $asmdiskgroup to be used in the restored database. For a file system, $filelocation is used. It indicates where the restored database will be placed, this will be usually either a local file system or Network Files System (NFS) storage.

• Type of recovery we want to do: $partial: it restores only read-write part of the database, $readonlytbl: percentage of read-only tablespaces to restore, $tblpitr: in case of a schema recovery, $timeinterval: till when to recover the database.

• Export: $exportype: per user or full, $sexp: use traditional export (exp) or export datapump (expdp), $shelpcmdcreateidir: create the directory object on the database for expdp utility, $sparfile: parameters files for the export utilities.

• Export protection parameters: $dsmcarchive: archive the export file and log on tape, $scprepository: secure copy export file and log to an offsite backup file server.

Detailed information about all configuration parameters can be found under the SourceForge project [1].

The rest of directories are self explanatory: “export” contains export related file mainly parameters files; “logs” means all logs will be placed here; “pfile” contains init files for the restored database. Modifying parameters files or init files will not require any change in the recovery code, as those are just input variables for the platform; you can also create brand new files and point to them using the suitable configuration variable under the “etc” recovery definition file that applies to your restore.

The workflow of a recovery and posterior export can be seen in Figure 3. After a successful restore the DBA can configure the platform to perform an export. The conditions to perform a consistent export are ideal, as the recovered database has no user session connected to it, no action is performed on it and thus the export is guaranteed to complete without any Oracle error linked with consistent reading and to be consistent across tables. The tool can deal with full or per user exports using either “exp” or “expdp” utilities, depending on the database version. If the export is successful, the DBA can configure to store a copy on tape and to make an extra copy on disk. In our case we use an offsite server to keep the latest two or three weekly exports at hand, while copies on tape can be kept for longer duration.
The “-t” parameter can be used to run the platform showing just what it would do. In this case a username and password ($user$ and $password$ configuration parameters in the recovery definition file) would be required in order to connect to the real database and extract all information. Running the ‘-t’ option is optional however. Otherwise without the “-t” parameters a real restore/recovery will be launched where all information is retrieved from the restored controlfile.

The platform works in isolation from the production environment. This is achieved by a trimmed tnsnames.ora (just one entry of the production database to be restored, to be used by the aforementioned “-t” parameter ), with no RMAN catalogue connection, and as a result, it just uses the controlfile backup information. No jobs are allowed to be run, which is accomplished by a clean-up of dbms_scheduler jobs as soon as the database is “open resetlogs” on 10g and 11g databases and having the “job_processes” init parameter are set to zero on all Oracle server releases.

Sequence diagram of recovery and an export are shown under the documentation in the SourceForge project [1].

3. Use cases
There is not much to say about a normal recovery, this functionality is the base feature of the tool. After starting the database using a previously recovered or brand new init or spfile, the process continues with restoring a suitable copy of the controlfile that should contain backup information needed to proceed with the restore and further recovery till the desired point in time. The process usually ends in opening the database in a new “incarnation”. Herewith, we present some use cases that can exploit advance features of the recovery platform.

3.1 Using recovery as a backup to disk.
As prerequisite, the storage used for the restore/recovery operation must be reachable by the production database.

We consider the case where due to some manipulation a production database is at risk. While we are dealing with the problem, we may want to start a restore/recovery operation till a point in time just before the issue started; we configure it using the $timeinterval variable. In this situation we will run either the recovery_wrapper.sh or directly recexe.pl script with the “-t” option. This will generate the three basic scripts of a recovery: contrl_restore.rcv that restores the control file at the right time; db_restore.rcv that restores and recovers the database till $timeinterval; and db_start.sql that opens the database. But this time we run the first two manually. After this is achieved, we only need to register the new database image files at the right point in time in the production database. This is achieved by means of the RMAN command “catalogue datafilecopy” [2]; effectiveness of such an operation can also be tested via a RMAN command “restore database preview”. The production database must be at least mounted for these operations to succeed.

3.2 Partial recovery, validation of a backup strategy
For very big databases, where most of the database is read-only, test recoveries can be too lengthy and require allocation of a lot of resources. Partial recovery was created as a way to validate a possible backup strategy for such big databases. Using $partial and $readonlytbl variables on the recovery definition file we can indicate first to restore/recover the read-write part of the database and then either restore a percentage of read-only tablespaces, or restore an specific list of those or none at all. The exact details including commands used are provided in [1].

3.3 Partial restore: tablespace point in time recovery.
Due to logical errors it may be needed to restore a schema at a certain point in time. If the database is big, this may require some time. Under certain conditions, restoring the main pieces of a database (system, syaux and undo tablespaces) plus the self-contained schema would be a much faster operation. The $tblpitr configuration variable allows to configure this type of restore. Apart from the ones already mentioned, all tablespaces containing objects of SYS and SYSTEM schemas and the ones containing objects of the schema we want to restore need to be considered.

As explained in Oracle support note [3], if the database uses XMLDB, the tablespace concerned by this feature needs to be appended on $tblpitr.

We observed some issues with export datapump [4] that usually are bypassed using traditional export. Further details are provided in [1].

The use of this technique presents several advantages because it is usually faster to recover several tablespaces than the whole database. In our day-to-day practice we have observed a time reduction varying from 60% till 80% compared to a whole database recovery. Additionally, in terms of recovery resources, less space is required while doing a partial restore, which could be an important factor for VLDB. Furthermore, it presents serious advantages compared to the in-built tablespace point in time recovery for Oracle server version higher than 9i [5], whereby no backups are invalidated once the restore has been done, hence the controlfile of the production database is never altered.

4. Conclusion
The Recovery platform was initially developed to validate RMAN tape backups. It runs on the principle of total isolation from the production database. After meeting this initial requirement, the platform evolved to provide a tool for DBAs that can be used in multiple situations: using a recovery as a backup to disk; export a specific user at a given point in time in the past; validating backup strategies like the one for very big databases; speeding up recoveries like partial ones; etc.

In normal operations after a recovery has been achieved, an export (full or per user) is done. The platform can be configured in a way that this export can be stored on long term storage, i.e. TSM for future use. It is indeed a kind of snapshot of the database at a certain point in time. The platform also provides long term storage for successful recovery scripts, which may come handy in case of a real recovery on the production database. In addition, the platform can be easily deployed on any server provided installation requirements are fulfilled as described in [1]. The underlying storage can be a local filesystem, NAS/NFS file access or SAN/ASM block access. Multiple Oracle versions are supported as well as operating systems. We are confident that the platform can easily be extended to support other database versions, operating systems and tape storage management software.

To conclude, with more than 3500 validation recoveries in about two and half years, the recovery platform has shown its importance and success in the CERN database environment. It has helped to discover and manage difficult cases, such as schema recovery on VLDB to solve logical errors like deleting a table or tape incomplete full backups for a database. As far as we know, no other software enables a complete validation of the database backups.

References
[1] Ruben Gaspar Aparicio, 2010/06/30, Recovery platform for Oracle databases, SourceForge project: http://sourceforge.net/projects/recoveryplat/
[2] Antonio Romero, 2005, Backup and Recovery reference 10.2 B14194-3, Oracle Database.
[3] Oracle, 2008, Oracle 9i: Full Export Fails with ORA-1403 (No Data Found) on Exporting Cluster Definitions [ID 255724.1], My Oracle Support web site.
[4] Kathy Rich, 2005, Utilities 10.2 B14215-01, Oracle Database.
[5] Antonio Romero, 2008, Backup and Recovery Advanced user’s guide 10.2. B14191-3. Oracle Database.
[6] David Austin, Mark Bauer, Richard Strohm, Douglas Williams, 2010, Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide 10.2. B14197-15. Oracle Database.
[7] IBM, 2010, Administrator’s reference Tivoli Storage manager for Linux version 6.2. IBM corporation.