CGI Helps Major North American Oil & Gas Company Realize $500,000 (CDN) in Cost Savings by Moving from Third-Party Backup Tool to RMAN

By: Maria Anderson, Senior Consultant, CGI
September 2006

This paper takes you through a migration from a third-party database backup tool to Oracle9i Recovery Manager (RMAN) at a major North American oil & gas company. Founded in 1976, CGI is the eighth largest independent information technology and business process services firms in the world. CGI and its affiliated companies employ approximately 25,000 professionals. CGI provides end-to-end IT and business process services to clients worldwide from offices in Canada, the US, Europe, Asia Pacific as well as from centers of excellence in Canada, the US, Europe and India.

This paper will focus on the following objectives:

- To share practical information from an actual RMAN implementation
- To better understand the importance of basic project management concepts

The project lifecycle, including the initial due diligence and evaluation of RMAN, as well as the actual project planning and delivery steps will be discussed.

Maria Anderson was the technical project lead for the backup and recovery migration to RMAN described in this paper. Maria is a Senior Consultant with CGI, where she is part of the Database Services team with responsibility for the management and administration of client databases. Maria has been a DBA for the past 10 years. She is an Oracle Certified Professional, holds a CCP designation from the Institute for the Certification of Computing Professionals, and has her I.S.P. designation from the Canadian Information Processing Society.

What is RMAN and How Does It Work?

RMAN is a free, Oracle supplied tool with both a command-line and GUI interface via Oracle's Enterprise Manager. It manages database backups and recoveries for you as opposed to traditional, user-managed backups using scripts. RMAN is tightly integrated with the Oracle kernel so this makes it a very powerful backup and recovery utility. RMAN reads through the blocks in the database as it performs backups. This provides potential for the backups to be smaller in size as RMAN only backs up blocks that have ever been written to; the added benefit is that there is no need for rigorous testing across platforms and versions. Oracle has already done this for you! At the same time, RMAN checks for corruption while it is reading through the database blocks. This is completely unique to RMAN as neither in-house backup scripts nor third-party products provide this value-added function.

The main components of RMAN include the target database to be backed up and the RMAN client. The main function of the client is to interpret backup and recovery commands, as well as record backup and restore actions in the target database's control file. Channels are opened (usually multiple channels if Enterprise Edition is being used) and the database blocks are read through buffers to check for corruption. If there is no corruption detected, the blocks in the buffer are written to the backup set which could be on either disk or tape.

An optional component of the RMAN architecture is the catalog. This is a schema in a separate database (on a separate server from the target database(s)) that keeps historical backup and recovery information for all target databases backed up by RMAN. While using a catalog is considered an RMAN best practice, it is not required to perform most basic backup and recovery functions. This, however, comes with some caveats: one is that the controlfile of the target database is available or can be easily recovered, and, second, the controlfile contains backup information that goes far enough back in time to make recovery possible (since control file records can be recycled based on the initialization parameter control_file_record_ keep_time).

Comparison of RMAN to Third-Party Software

There are alternatives to using Oracle's RMAN for database backup and recovery functions. One alternative is to write scripts to perform physical database backups. Writing scripts requires a considerable amount of time and effort, not to mention the time required for testing. The last thing you want is to discover that your backup scripts have neglected to perform a crucial step that has rendered a database recovery impossible. Scripts also need to be updated as new database releases become available. If UNIX shell scripts are being used to perform database backups, there will often be subtle differences between major OS releases that may require additional fine-tuning.

Another alternative to using RMAN is to purchase a third-party product. There are several products readily available on the market. One advantage to using a third-party product is that the vendor maintains the product, not you. So, unlike scripts, the backup software will work across a number of operating systems and database releases. Some products also work across different database products i.e., Oracle, Sybase, DB2, etc. There are a number of disadvantages to using vendor-purchased database backup products. One disadvantage is that there is some setup required before you can start to backup databases. Usually you must install the product, sometimes on multiple servers, and then configure it so that it will perform database backups. Another disadvantage is that you now have more software that you must backup and manage. Recovery of the backup software must be built into your business continuity and disaster recovery planning; operating system backups must include this software, as well as any specific configuration files. One final disadvantage to purchasing third-party database backup software is, of course, the cost. These products are often very expensive to purchase and maintain. Beyond the purchase price, there may be an additional annual cost to maintain or license the product. Finally, licensing of the product may be based on a number of factors (server size, total backup size, etc.) that could make the cost of the product prohibitive.

Evaluation and Investigation

The primary driver behind evaluating Oracle9i RMAN for this oil & gas company was, quite simply, cost. The current third-party database backup software was very expensive to license and maintain, while RMAN had undergone quite a number of improvements as of Oracle9i Release 2. These two factors combined were enough to prompt an evaluation of RMAN as a replacement database backup and recovery tool for what was currently in place. The DBA team at this client is composed of thirteen DBAs: four are employees of the company while the remainder are CGI DBAs onsite at the client. Out of all the members of the DBA team, four were interested in evaluating RMAN.

The Environment

The database environment at this oil & gas company is comprised of mainly Oracle databases (approximately 200 non-production and production) from version 8.1.7.4 to 9.2.0.4, with a few historical 7.3.4 databases. There are also approximately 65 Microsoft SQL Server databases. Additionally, there are a few Sybase databases that contain historical data or are being phased out. Another reason for evaluating RMAN was that we no longer required a multi-platform database backup utility so there was relatively low risk in migrating the remaining Sybase databases to use in-house scripts.

Evaluation strategy

The first step in our evaluation strategy was to read about Oracle 9.2 RMAN to determine what features it provided. This company had a large, but equal, number of Oracle 8.1.7.4 and 9.2.0.4 databases. The strategic plan at that time was to upgrade or migrate many of the Oracle8i databases to 9.2. After determining that RMAN had all the features we considered important, the next step was to plan a more structured, or formal, evaluation strategy.

The next step in our strategy was to create a 9.2.0.4 RMAN catalog, as well as two databases (one Oracle8i and one Oracle9i) and register them in the catalog. These databases were of a representative size and, since they were not actively used, could be backed up and recovered as often as was necessary. These databases were first backed up and recovered with the current third-party product. The same databases were then backed up and recovered with RMAN. Backup and

recovery times were documented, as well as backup sizes. The one obstacle we knew about even before we began testing was that the third-party product compressed the backup sets whereas Oracle9i RMAN did not. This meant that we would require more disk added to our backup volumes. Overall, backup and recovery times between the two products were very similar.

After testing basic database backups and recoveries interactively, we were ready to progress to the next phase of the evaluation.

Information-sharing session

After some initial RMAN testing and evaluation, the results seemed promising but the DBA Team Lead wished to speak with DBAs at other oil and gas companies that were using RMAN to confirm our findings. A RMAN information-sharing session was arranged between DBAs at four moderate- to large-sized oil and gas companies. The primary objective of this session was to determine how the DBAs had set up RMAN at each of their respective organizations. A secondary objective was that each DBA might also learn something new about RMAN from discussing their respective architectures and asking questions or discussing issues.

The three other companies asked to participate in this information-sharing session had all been using RMAN for at least 2 to 3 years. Some had been using RMAN since it was first introduced with Oracle 8.0 so all DBAs at this session were senior and had an incredible amount of experience with RMAN. There was a willingness from all companies to share RMAN backup scripts and architecture notes, which was most helpful.

Two of these three companies were backing up their databases with RMAN directly to tape, while the third was backing up to disk and then to tape. We had decided to continue to backup our databases to disk as we had been doing with the third-party software. We had a large disk area comprised of four NetApp filers with approximately 20 TB of disk visible from all production servers. This would be enough to keep four backups online as well as required archived redo logs for production databases. This was because three full backups were always required to be kept per our service level agreement (RMAN retention policy of redundancy=3) and the oldest backup becomes obsolete when the addition of the current backup exceeds the redundancy policy. The oldest backup's disk space can be simply reclaimed by using RMAN's "delete obsolete" command.

Encouraged by the reliability and robustness of RMAN, we decided to perform additional testing. Since the four DBAs involved in the evaluation process would be responsible for recommending a solution to the entire team and the DBA Team Lead, we wanted to be very certain of our findings. The next step was to inform the DBA team of our decision.

The big decision

The final step in the evaluation phase was to present our findings to the entire DBA team and the Team Lead, as well as our recommendation. This took the form of a presentation plus a few recovery demonstrations. After completing our RMAN evaluation, our recommendation was to replace the third-party software with RMAN. RMAN has proven itself as a reliable and robust database backup and recovery tool. The DBA team unanimously agreed with the recommendation presented. The next step was to formally develop a project and officially start the RMAN implementation!

Project Management 101

This project was officially started with consensus reached by the entire DBA to replace the third-party software with RMAN. For most of the first phase of the RMAN implementation project, we did not have an "official" project manager but we were very committed to the project and became a self-directed project team. In early February 2005, a project manager was assigned to the project team.

Many DBAs start and finish projects just like this without a project manager or team lead to direct the project. In cases such as these, technical documentation is not the only type of documentation that is essential. There must be documentation on the progress of the project (e.g., meeting minutes), decisions made and action items documented and assigned. Without this, at best the project may succeed but there will be no learnings that could be applied to future projects. A worst-case scenario is that the project fails or is never completed because action items are not addressed and

momentum is lost. This, of course, also speaks to ownership of the project. Without a project manager, the project team must take ownership of the project and ensure that progress is made and documented.

The project team

The project team was comprised of the same four DBAs who took part in the evaluation process plus one more DBA, bringing the total to five. Now that the project team was shifting from evaluation mode to more of a "proof of concept" mode, the team was expanded by one DBA. The five DBAs were all senior DBAs who had worked with many different database platforms, operating system and backup/recovery tools. This culmination of experience was very helpful throughout the project. As well, because we worked as a self-directed project team, there was ownership in ensuring that the project was successful.

The project plan – a phased approach

The timeline for completion of this project was ten months. This gave the project team ample time to plan the implementation and ensure that there were no unscheduled outages as a result of the migration to RMAN from the current product. Our primary objective in this project was to migrate most, if not all, our Oracle production databases to RMAN with no unscheduled outages.

The project plan consisted of three phases as follows …

PHASE I

Investigation and Preparation

- Search Metalink for any potential RMAN bugs or patches required
- Investigate how to implement the use of password files for Oracle8i databases (required by RMAN)
- Determine the basic architecture of the RMAN environment
- Discuss the issue of compression and RMAN backups

RMAN Catalog

- Check with other oil and gas companies to determine if they have discovered any significant RMAN bugs
- Catalog maintenance
- Backup/recovery strategy of the catalog
- Reporting or retrieving data from the catalog
- Determine how often to synchronize the catalog
- Investigate the use of multiple catalogs for redundancy and failover

Scripting

- Review RMAN scripts used at other oil and gas companies
- Modify existing backup scripts to use RMAN rather than existing third-party product
- Test backup scripts
- Create script to monitor archive log destination
- Modify purge script (to purge archived redo logs) so that it recognizes RMAN backups
- Ensure that RMAN backup log is written or copied to the backup directory

Test Backup and Recovery Scenarios

- Online backups
- Complete and point-in-time recoveries
- How to determine when an RMAN backup is running and how to kill a backup job

Test RMAN Duplicate (Cloning) Feature

- Test duplicating an instance to the same server
- Test duplication to another server

PHASE II

Training

- Technical sessions for the DBA team throughout the implementation phase at regular intervals

Scripting

- Create script to clean up backups older than a certain day at the operating system level, and then crosscheck the RMAN catalog so these backups are expired properly within the catalog.

Test Backup and Recovery Scenarios

- Offline backups
- Tablespace point-in-time recoveries (TSPITR)
- Test database crash during RMAN backup
- Test backups if the catalog is unavailable
- Test recoveries with OEM 9.2

Migration Planning

- Create list of production database that need to be migrated to RMAN
- Move all Sybase databases from current third-party product to use in-house script for backups.
- Migrate all Oracle 7.3 databases from current third-party product to use hot backup script.
- While implementing RMAN, perform a quick health check in each server (change DBID, check exports, etc.)
- Capacity planning
    - How large will the backups get?
    - How much free space do we need on the backup disk for backups to be successful?
    - Need to notify Storage team in advance if more disk required on backup volumes.

Documentation

- Include a RMAN section in DBA Oracle administration and operations guide
- Enhance ‘how to’ RMAN quick reference document
- Provide links to Oracle RMAN documentation on DBA website

Pre-Production Checklist

- Implement password files for all Oracle8i databases as RMAN requires connect as sysdba. Also change init.ora to use REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE.

- Change database creation procedure to create the RMAN backup user.
- Change database creation scripts for Oracle8i and Oracle9i to use a password file for remotely connecting as the backup user.
- Change any DBIDs that are duplicated as databases are migrated to RMAN.
- Determine how we want to validate our RMAN backups ('validate backupset' or 'check logical' or both).
- Determine whether we want RMAN to continue a backup if some files cannot be backed up.
- When new tablespaces or datafiles are added to production databases, either a tablespace backup or full catalog resynchronization should be performed.
- Prepare RMAN configuration parameters for Oracle9i databases.
- Ensure that the permissions on the Oracle intelligent agent are set correctly otherwise Oracle9i EM cannot be used for recoveries.

PHASE III

Training

- Half-day boot camp session for DBA team as well as on-call DBAs

Implement RMAN Backups

- Devise a phased approach for migrating production backups to RMAN to minimize user impact.
- Optimize RMAN backups

Post-implementation Tasks

- Clean up any orphaned database backups on backup volumes.

OEM 10g and RMAN (optional)

- Investigate using OEM 10g for RMAN database recoveries; how does it compare to OEM 9.2?

The project was officially kicked off in early November 2004 and was completed by late July 2005. The project was also completed within budgetary constraints.

Project documentation

Project documentation was maintained throughout and proved to be very helpful with respect to keeping the project on track. At the start of the implementation, the project team met to devise a very high level plan. Out of this high level plan, a more detailed plan was distilled and, finally, a proper project plan.

Weekly meetings were held almost without exception to ensure that action items were addressed and the project was kept on track. Meeting minutes were taken and distributed either that same day or soon after. At each subsequent meeting, action items were carried forward that were not completed or addressed.

A detailed technical implementation plan was also devised much later in the project to assist the DBAs with migrating the databases to use RMAN. The document contained a step-by-step approach to the implementation, as well as specific commands to ensure that the entire team was following the same procedure.

Prior to beginning the production rollout of RMAN, the Oracle administration and operations guide (which is referenced by the entire DBA team) was updated with useful technical information. As well, a quick reference was developed for those situations where the procedure was well known but the syntax was not.

Implementation

With the RMAN project officially underway, the following deliverables were confirmed:

- To replace the current third-party product with RMAN for performing database backups and recoveries with no unscheduled outages to any production systems by August 2005
- To provide RMAN training to the DBA team and the on-call DBAs
- To update and/or create relevant technical documentation

Testing strategy – backup and recovery scenarios

Backup and recovery testing continued throughout the project. While this may seem contradictory – one would think by the time the implementation is being planned, we should be certain about the product – it helped to reinforce and validate our early findings. At any point in the project prior to actual implementation, we could have backed out and kept the third-party product if we felt uncomfortable with RMAN. In later phases of the project, the types of recoveries we tested were more complicated. During the early phases of testing, the primary concern was testing the basic types of recoveries (complete and incomplete). In later phases, the testing evolved to include tablespace point-in-time recovery, recoveries to other nodes, and database cloning (duplicate). Each time testing was performed, the procedure and results were documented so that comparisons could be made across the project. Because results were consistent and reliable, we were comfortable knowing that the testing was thorough.

A final phase of testing was to implement RMAN on a busy development server. This was as close as we could get to production without actually deploying RMAN to production. This development server has 10 to 15 very busy development databases so it gave the project team a life-like working environment to test scripts and procedures in. The databases were all put into archivelog mode so that online backups could be run. The necessary RMAN backup scripts were scheduled and allowed to run. On several occasions, the archive destination on the server filled which, while this was not an emergency situation, did not please the developers and so had to be rectified within a reasonable amount of time. This allowed the project team to learn how to rectify the problem using RMAN in a timely fashion. It also gave us an opportunity to fine-tune the RMAN scripts after observing how they behaved in a busy environment.

Migration planning

Considerable time and effort by all project team members went into planning the migration, or rollout, of RMAN. Our primary objective was to make the transition transparent to the user community. We had to be able to transition from the third-party product to RMAN without unscheduled outages, and we also must be able to recover with either product depending on when a recovery was required.

A spreadsheet with a list of production databases was created with the following information:

- Database version
- Location of database server (there were two data centers)
- Business group
- Priority rollout (1 through 12)
- Backup utility (RMAN for Oracle8i and Oracle9i databases; scripts for Oracle 7.3 and Sybase databases)
- Password file
- Duplicate DBID
- Applications supported
- IT Contacts
- Implementation Date
- Implementer
- Comments

The first step was to determine which production databases did not have a password file. Because RMAN requires SYSDBA privileges to stop and start the databases, the parameter REMOTE_LOGIN_PASSWORD must be set to EXCLUSIVE. Not all production databases had a password file so this required planning outages with the application support teams.

The next step was to designate which servers and databases would be migrated first. The least visible (and least accessed) databases were chosen to be migrated first which would lessen the impact if something did go wrong unexpectedly. The corporate-wide, mission-critical databases (such as those supporting document management and financial systems) were migrated last.

Once scheduled outages were approved, members from the entire DBA team (not just the project team) were recruited and assigned to create password files. The workload was distributed evenly across the entire team, which meant that the project team members did not have the responsibility for carrying out the entire migration. The same strategy was followed for implementing RMAN – members of the team were recruited to perform this work, as well. This helped the rest of the DBA team become more familiar with RMAN and the architecture that the project team decided upon.

RMAN catalog strategy

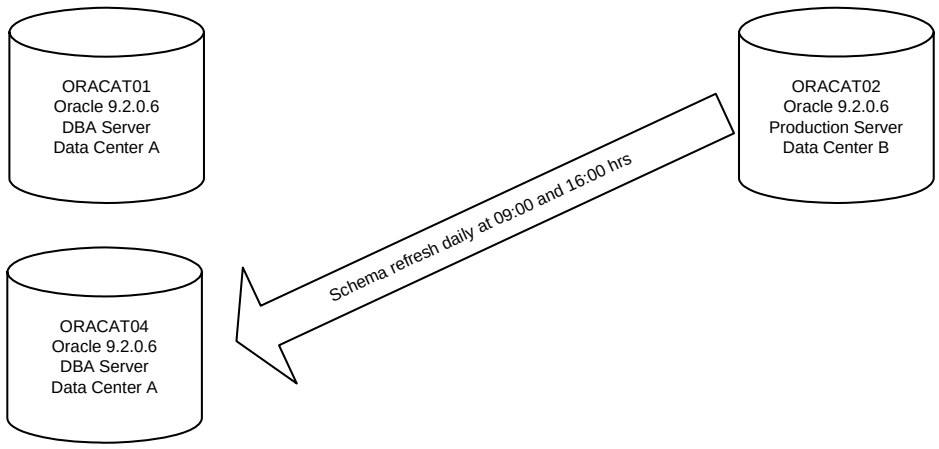
The RMAN catalog was initially created on a server devoted entirely to DBA tools and utilities. This is where the OEM repository existed, as well as other performance tools such as *Indepth for Oracle*. This first catalog (ORACAT01) was created on this server with its own ORACLE_HOME. This way, we could patch the catalog database first without affecting any other databases as no other databases were using these binaries. This catalog was eventually relegated to be a development RMAN catalog. This is where we would backup non-production databases and also test RMAN catalog upgrades. The first production catalog (ORACAT02) was created on a production server with its own ORACLE_HOME in the opposite data center from the first catalog. This is the catalog for all production databases.

The RMAN scripts (online backups, cold backups and archived redo log backups) require the catalog to be available for the backups to begin and complete successfully. Many of the production databases in this organization are co-hosting a multitude of applications that are very busy so enormous amounts of archived redo logs are produced on a daily basis. As a result of this, the RMAN catalog cannot be unavailable for more than 2 or 3 hours at a time or the archive destinations will fill. With this in mind, a catalog failover strategy was devised.

We could have used Data Guard for this purpose but because the catalog itself is very small in size, we decided that a twice-daily export/import from one catalog to another would meet our requirements. Another catalog was created on the DBA server (ORACAT04) and this would be the failover catalog for ORACAT02. Each day at 9 AM and 4 PM, a shell script would perform an export of the RCAT schema in ORACAT02 and, if successful, would import this into the same schema in ORACAT04. If ORACAT02 should become unavailable for any length of time, it would be a simple matter of pointing all production databases to ORACAT04. The information in the catalog should be no more than a few hours behind the controlfiles, and they would resynchronize with the next backup job. In addition, ORACAT02 and ORACAT04 resided on servers in opposite data centers so this assisted with our disaster recovery strategy.

Other than twice-daily exports, the production catalog (ORACAT02) was in archivelog mode and backed up daily with in-house hot backup scripts. We could have chosen to backup the catalog with RMAN (using the nocatalog option) but we decided that if we needed to recover the catalog quickly, the DBAs were most familiar with hot backups. After RMAN is implemented enterprise-wide, backing up the catalog with RMAN might be a possibility in the future.

A diagram of the RMAN catalog architecture follows:



RMAN catalog failover strategy

Scripting

The existing shell scripts performed a number of other functions besides backing up the databases. History data, for example, was entered into a database where the DBAs could query to determine how often certain database backups or exports failed. An HTML report was also produced with this information that would allow the DBA responsible for database monitoring to quickly see which backups had failed for a particular day. After evaluating a number of scripts from DBAs at other organizations and on the internet, we decided to keep our existing scripts and modify the actual backup commands to be RMAN commands.

When starting a project like this, one would think that scripting will be the most difficult phase of the project. In fact, it turned out to be the easiest. This is not to say that scripting was an easy task, but we had a project team member who was very good at scripting which helped make this phase of the project go smoothly. By and large, the existing backup scripts remained unchanged except for the specific backup syntax so they were already well tested. Nevertheless, as soon as the backup scripts were ready they were used through the project and recoveries were thoroughly tested after databases were backed up with the scripts.

There were basically two RMAN backup scripts: one performed either an online or offline database backup depending on the parameters passed, while the other backed up archived redo logs. We chose to perform nightly full online backups rather than introducing incrementals into the backup strategy; we wanted to keep as much the same in this migration as possible. Both scripts take parameters from a parameter file called rman_master.lst. This file has specific backup parameters for each database. The backup scripts read the parameters in this parameter file, generate the RMAN backup commands, and then execute them. The archived redo log backups were scheduled to run against all production databases every 30 minutes. This way, we could be assured that no archive destinations would fill.

One other script was written to delete any old backups from the backup volumes on disk and expire them in the catalog. This script ran during the business day against all databases. It first performed a crosscheck with the catalog, generated a list of backup sets to delete and then deleted them from the operating system. As per this client's corporate policy, three valid backups were kept on disk at any one time, although this did vary somewhat by database and according to business requirements. The Storage team took responsibility for ensuring that the backups on disk were backed up to tape.

The key to making this portion of the project a success is to have good script writers creating or modifying the backup scripts and to have thorough testers. Additionally, those project team members who are scripting should enjoy performing this function otherwise frustration will soon take over and make it very difficult to progress to the next stage of the implementation.

Disaster Recovery Considerations

Throughout this project, disaster recovery considerations were addressed. If there was ever a disaster situation, we had to be confident in the fact that we could not only recover our production databases, but that we would also be able to tell the Storage team which backup sets to recover from tape. To meet this requirement, the RMAN backup logs were copied to the backup volumes to reside with the backup sets. The log files contained the names of the RMAN backup sets so we could be sure that the log files would be backed up to tape by the Storage team.

Another consideration was ensuring that copies of the controlfiles were created for all production databases on a frequent basis. While this can be automatically set with a configuration parameter in RMAN 9.2, this was not the case with 8.1.7.4 databases. There was an existing script that ran several times each day to create controlfile copies, so we kept this script running even after RMAN was implemented. This was to ensure that we could recover databases even if we could not recover the catalog. The information we required would be in the controlfiles and the database could then be recovered with RMAN assuming that the backup sets could be restored.

A final consideration was to use a backup disk volume that resided in the opposite data center of the database itself. For example, if Production Database A resided in Data Center A, then a backup volume residing in Data Center B was chosen for the backups.

The Big Rollout

Once all production databases had password files and we ensured that all DBIDs were unique, it was time to start migrating the production databases to RMAN from the current third-party product. This was an exciting time for the project team! The production rollout began in late May 2005 and was completed in late July 2005. As mentioned, there were no unscheduled outages as a result of this migration, and the implementation was completed on time and on budget. As with other phases of the project, the entire DBA team was recruited to assist with the production rollout.

Migration procedure

When the project team initially planned the migration to RMAN, all migrations were planned during regular maintenance windows so that if something went wrong it was after the end of the business day, at least. But the more this strategy was discussed, the more it seemed like this was maybe not a good idea. If the backups did not work for some reason and the archive destinations were to fill, it would be better that this happened during the business day when all members of the project team were available to assist the person implementing the change. After business hours, not everyone is available, so it was decided that all production servers would be migrated to RMAN during the day.

The migrations occurred on a server by server basis and were prioritized so that the most visible databases were migrated last. The first step in the migration is to ensure that the target database(s) has the BACKUP_ADMIN user created with SYSDBA privileges. This is the user that RMAN will connect with to the target database to perform backups. A check is also performed to determine whether the REMOTE_LOGIN_PASSWORDFILE parameter is set to EXCLUSIVE. If not, the migration cannot continue since a scheduled outage must be arranged at a later date to change this parameter, then stop and start the database. Assuming this parameter is set appropriately, the next step is to check whether the DBID of the database already exists in the catalog. If it does, the migration cannot proceed as the DBID of the target will have to be changed using NID (create new DBID utility) or dbms_backup_restore.zerodbid during a scheduled outage. Assuming the DBID does not exist in the RMAN catalog, the migration can proceed at this point.

The target database must now be registered against the production catalog – in this case, ORACAT02. If the target database is release 9.2, configuration parameters must be set; this does not apply to 8.1.7.4 databases. All cron jobs and processes used by the third-party software must be stopped so that they no longer run on the server. Before attempting to backup the database or archived redo logs with RMAN, a crosscheck must be run against the target database since this is a newly registered database with the catalog so RMAN will expect to find all archived redo logs since the last resetlogs was performed. The crosscheck procedure marks missing archived redo logs as unavailable.

At this point, the RMAN backup scripts are added into cron and the first archived redo log backup job is monitored. If the archived redo logs are being backed up successfully to the backup volumes, this indicates that the online backup will likely be successful as they are executing basically the same script. Since the backups usually run during the evening or early morning hours, the DBA responsible for the migration will follow-up the next day to ensure that the online backups were successful for that server.

The very last step in the migration is to delete any old backups left behind by the third-party backup software. As this takes up disk space on the backup volumes, they must be deleted after they have been copied to tape. This final step is usually not performed until three or four days after a successful migration to RMAN.

Statistics on servers and databases migrated

All UNIX servers migrated were either Solaris 8 or 9 and all Oracle binaries were 32-bit.

Number of Oracle 8.1.7.4 databases migrated:	38
Number of Oracle 9.2.0.4 databases migrated:	25
Total number of databases migrated:	63
Total number of database servers:	12

Smallest database migrated:	2.0 GB
Largest database migrated:	350.0 GB
Average database size:	31.0 GB
Median database size:	11.0 GB
Total migrated:	2.0 TB

Summary

This RMAN implementation was definitely a success – the third-party backup software was replaced by RMAN on time and on budget with no unscheduled production outages. Over a three year period, this saved the client approximately $500,000 (CDN). Also, the project team members discovered that not only was RMAN a logical replacement for the third-party product, but there were some additional impressive features, such as duplicating a database. RMAN was very easy to implement (with sufficient planning) and became very easy to use once we had some exposure to it and tested the required database backup and recovery functions.

With some basic project management skills, DBAs can ensure that projects they are leading or even just participating in will be successful. If you are the project lead, take a common sense approach to the project. It will be necessary to create a project plan, to hold meetings and take minutes, keep track of action items, and do much of the administrative paperwork but it is also important to not get lost in the process. As the project lead, you must always be one or two steps ahead of the project team to mitigate risk and keep the project on track.