PURESTORAGE

Using FlashArrays for Oracle Backup and Recovery

AR-160601-v01

July 2016

Document History

Version	Changes
July 2016	Draft d01. Starting from Daniel Higgins original paper
July 2016	Draft d02. Incorporated revised appendix screenshots by Somu Rajarathinam
July 2016	Draft d04. Incorporated revised Appendix A screenshots by Somu Rajarathinam
July 2016	Draft d05. Incorporated further corrections from Somu & Daniel.
July 2016	v01. Incorporated legal corrections, changed markings, and published

Contributors: Somu Rajarathinam

Pure Storage, Inc.
CTO Office
650 Castro Street, Suite 300
Mountain View, CA 94041

http://www.purestorage.com

Contents

Illustrations

Abstract

Protecting an organization's data is easily the most important responsibilty of any IT department. In most organizations, database administrators (DBAs) are responsible for ensuring that data is well protected and readily available.

A variety of backup and recovery tools and strategies for using them to protect databases against hardware failure, natural disaster, human error, and so forth, are available to DBAs. A common property of all methods however, is that the larger a database is, the more difficult and expensive it is to protect it adequately.

This report demonstrates how features of Pure Storage™ FlashArrays can significantly reduce the time and cost of protecting Oracle databases compared to other methods. Although the report focuses on Oracle, many of the same techniques can easily be adapted for use with other database technologies.

Audiences

The primary audiences for this report are Pure Storage System Engineers (SEs) and Technical Support Engineers (TSEs), Certified Partner system engineers, Oracle database administrators (DBAs), and other IT professionals who are concerned with the economics and operational impact of protecting data in Oracle databases. The report assumes that readers are generally familiar with Oracle database backup and recovery, and with FlashArray concepts and the CLI and GUI administrative interfaces.

References

- Recovering Through an Added Datafile: Scenario
  https://docs.oracle.com/cd/B10500_01/server.920/a96572/recoscenarios.htm#14480
- RMAN Incrementally Updating Backups
  http://docs.oracle.com/cd/E11882_01/backup.112/e10642/rcmbckba.htm#BRADV8186
- Using Block Change Tracking to Improve Incremental Backup Performance
  http://docs.oracle.com/cd/E11882_01/backup.112/e10642/rcmbckba.htm#BRADV8125

Oracle® Backup Strategies

Using a production host to back up a live database is costly, both in terms of the storage capacity it requires and of the negative effect it can have on source database performance. Moreover, as a database grows, backing it up consumes more storage, takes longer, and has a larger effect on source database performance. The "right" database backup strategy is one that balances cost, data protection, and accessibility to meet the organization's requirements.

Storage array capabilities, especially data reduction and snapshots, can significantly reduce both the storage requirements and performance impact of live database backup. This report describes the most efficient Oracle RMAN backup strategy to use with Pure Storage™ FlashArrays, and shows how FlashRecover snapshots can minimize storage requirements and all but eliminate the performance impact of backing up Oracle databases.

Conventional and FlashArray-Based Oracle Backup

The conventional mechanism for backing up live Oracle databases is the *Recovery Manager* (RMAN) utility that has been part of the database management system since version 8. RMAN can back up an entire database or selected tablespaces while the database is "live." Backups may be full or incremental. Table 1 contains representative resource consumption for RMAN full and incremental backup of a hypothetical 1 terabyte database, and compares it with comsumption for two techniques available for databases that use Pure Storage FlashArrays for storage.

Backup Type	Storage	Database Size	Database Space	Backup Space	Total Space	Backup Duration	CPU Consumption
RMAN Full Compressed	Conventional Disk Array	1000GB	1000GB	500GB	1500GB	8hr	high
RMAN Incrementally Updating	Conventional Disk Array	1000GB	1000GB	2000GB	3000GB	30 min	low
RMAN Incrementally Updating	FlashArray	1000GB	333GB	250GB	588GB	5min	minimal
FlashRecover Snapshot	FlashArray	1000GB	333GB	150GB	488GB	0 min	none

Table 1: Comparison of Oracle Backup Strategies

Oracle Backup Methods with and without FlashArray Storage

Backup and recovery is easily a DBA's single most important responsibility. Loss of key data can affect an organization's profitability, and in extreme cases, even its ability to operate. But maintaining reliable, readily available backups of large databases is expensive. As a consequence, DBAs have historically been forced to choose backup strategies that balance storage cost and performance impact against recovery points, restore times, and consistency of the recovered data.

RMAN is widely used by Oracle DBAs, partly because it automates what had previously been multiple separate tasks, but perhaps more importantly because it maintains an *RMAN Catalog* (RCAT) of the times, types, and locations of database backups. RCAT metadata simplifies recovery by eliminating the need for DBAs to determine the correct combination of datafiles, control files and online and archived redo logs required in a particular recovery scenario. A DBA can instruct RMAN to recover a database or a subset of it to a specific point in time. The utility determines which files are required, restores them, and rolls the redo logs forward to recover the database to the specified state.

Other backup methods utilize 3rd party storage array features, notably snapshots and split mirrors. While these can reduce backup times and offload backup IO from the source database, they are generally complex to implement, costly in terms of additional storage capacity, and require coordination between DBAs, storage administrators, and server administrators.

FlashArrays reduce backup storage requirements, and minimize or eliminate the performance impact on source databases, but without the complexity of other 3rd party solutions. As a foundation for understanding how FlashArrays reduce backup cost and complexity the two sections that follow describe the most common Oracle backup strategies in use today.

The scenarios described in these two sections assume that:

- General best practices are being followed
- The database utilizes a Fast Recovery Area (FRA)
- Automatic Storage Management (ASM) manages +DATA and +FRA Disk Groups
  (The same principles apply when datafiles and the FRA are stored in separate file systems).

RMAN Strategy 1: Full Compressed Backup

Description

RMAN full compressed backup, illustrated in Figures 1 and 2, is commonly used to back up databases of one terabyte or less. It keeps backup storage requirements to a minimum by reading data from the **+DATA** area and compressing it before writing it to the **+FRA** area. RMAN compression typically reduces alphanumeric data to a fifth of its original size. For example, an RMAN full compressed backup of a 1 terabyte alphanumeric database would typically occupy about 200 gigabytes of physical storage.

```
RMAN> BACKUP AS COMPRESSED BACKUPSET DATABASE
      PLUS ARCHIVELOG;
```

Figure 1: Example Command for Executing RMAN Full Compressed Backup

RMAN full compressed backup requires FRA space for a minimum of two backups because the newest backup must be complete before the previous one can be removed. Thus, for the 1 terabyte alphanumeric database, 400 gigabytes of FRA space should be reserved.

In addition, the FRA must have sufficient space for archived redo logs, the size of which depends on the database's rate of change and the log retention policy. Adding a 100 gigabyte reserve for archived logs to allow for a 10% rate of change in the source database would mean 500 gigabytes of FRA space for the 1 terabyte database—a 0.5 to 1 **+FRA** to **+DATA** ratio--to retain two cycles of backup.

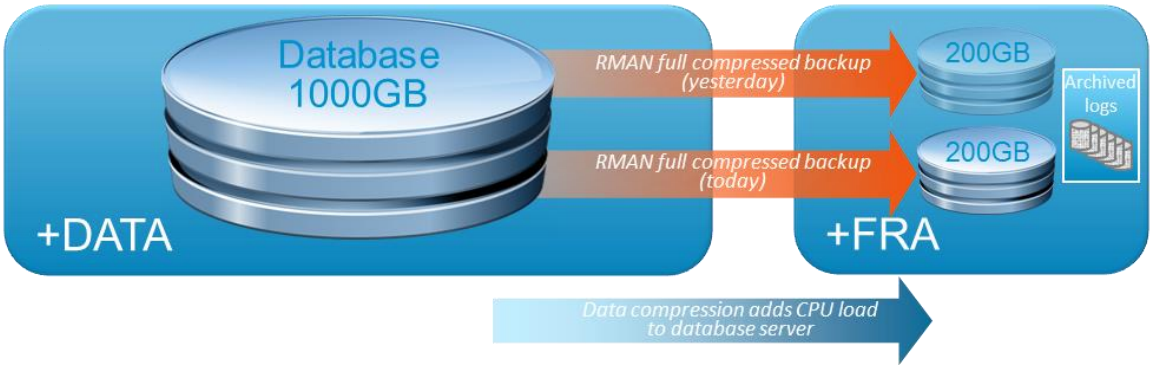

Figure 2: RMAN Full Compressed Backup

RMAN full compressed backup works well for smaller databases, but does not scale well with database growth. Its CPU-intensive data compression reduces the compute power available to source database transactions and applications. Moreover, compression increases backup elapsed time by as much as a factor of four over non-compressed full backup.

A non-compressed backup of a 1 terabyte alphanumeric database typically runs in about 2 hours. With compression, it could take up to eight hours, during which time source database performance would diminish due to the CPU load of compression. Parallel backup streams can reduce elapsed time, but concurrent CPU-intensive RMAN jobs only exacerbate the impact on source database performance.

For multi-terabyte databases, RMAN compressed full backup is simply unsustainable—backup times of 20+ hours have been observed with larger databases. In such situations, daily backup is simply not feasible, because the incremental CPU load detracts from source database and application performance essentially around the clock.

The scaling limitation of RMAN full compressed backup motivates consideration of more scalable backup strategies.

RMAN Strategy 2: Incrementally Updating Backup

Description

Oracle version 10g introduced a *incrementally updating backup* feature to speed up daily backup and consume less CPU in the process. Figure 3 is a sample script for daily incrementally updating backup:

```
RMAN> RUN
{
  RECOVER COPY OF DATABASE WITH TAG 'my_database'
  UNTIL TIME 'SYSDATE -7';
  BACKUP INCREMENTAL LEVEL 1
  FOR RECOVER OF COPY WITH TAG 'my_database'
  DATABASE;
}
```

Figure 3: Example Script for executing RMAN Incrementally Updating Backup

- The **RECOVER** command applies the incremental backup from seven days ago to the full datafile copy. The command does nothing until a 7-day old full datafile copy, created by successive iterations of the **BACKUP** command, exists.
- The **BACKUP** command creates an incremental backup each time the script runs. If no full datafile copy exists, the command creates one instead.

Figure 4 is a visual depiction of incrementally updating backup.

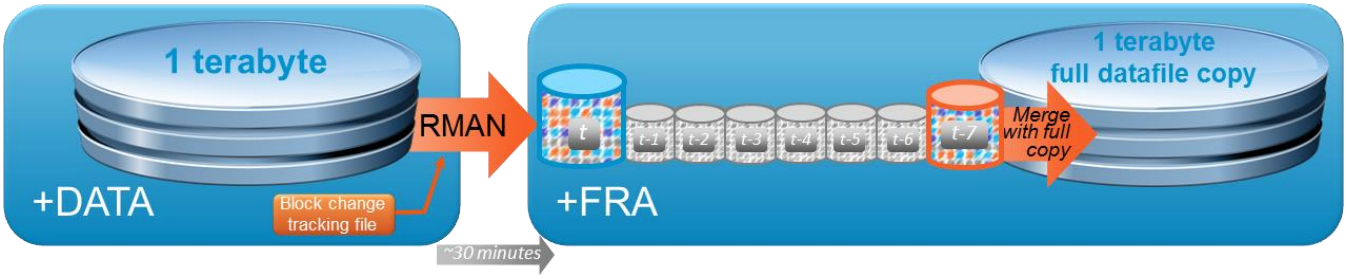

Figure 4: RMAN Incrementally Updating Backup

Incremental backups contain only data blocks that have been modified since the previous backup. Oracle tracks changed blocks in a *block change tracking* (BCT) file, so incremental backup is efficient because there is no full database scan to determine which blocks to include. Incrementally updating backup is the default strategy for backups scheduled by the Oracle Enterprise Manager (OEM).

In the example of Figure 4, daily incremental backups for 7 days are stored in the **+FRA.** At the end of each backup job,

RMAN merges the incremental backup from 7 days ago into the full datafile copy. Thus, at all times, the **+FRA** contains a full copy of database datafiles as of 7 days ago. The source database can be recovered to any point in time within the last 7 days by:

- Restoring the datafiles from the 7-day old full copy
- Applying the incremental backups up to the day of the restore point
- Rolling forward through the redo logs.

Incrementally updating backups cannot be compressed because that would prevent incremental backups from being merged into the full datafile copy. Because they typically copy only a small subset of a database's blocks, and because there is no compression, incrementally updating backup durations are typically an order of magnitude less than for compressed full backup. Shorter backup duration and elimination of compression minimize performance impact.

But while incrementally updating backup is more efficient than compressed full backup in terms of elapsed time and CPU load, there is a cost in terms of the amount of **+FRA** storage capacity required.

For example, a 1 terabyte database would require 1 terabyte of **+FRA** storage for the datafile copy, plus space for 7 days of incremental backups (and archived redo logs). A conservative **+FRA** allocation for an alphanumeric database would be 2 terabytes, giving a **+FRA** : **+DATA** ratio of 2 to 1 (compared to 0.5 to 1 for full compressed backup). So while incrementally updating backups run quickly and don't degrade database performance, they require much more storage than full compressed ones—as much as 4 times for a typical alphanumeric database. Even allocating the **+FRA** disk group on low-cost disk-based "tier 2" storage, this can become expensive for large databases.

FlashArray storage dramatically reduces the storage cost of RMAN incrementally updating backup, to the point where the physical storage required for the FRA is less than that for full compressed backup. Using flash to store backup copies is counter to the conventional database wisdom that backups should be kept on the least expensive storage available. FlashArrays completely reverse the equation, however, and have other benefits as well, as the next section demonstrates.

FlashArray Strategy 1: Incrementally Updating Backup

From an operational standpoint, this is precisely the strategy described in the preceding section. The difference is that FlashArray volumes are used for both datafile and FRA storage. (ASM rebalancing can migrate a database's **+DATA** and **+FRA** Disk Groups to FlashArray volumes non-disruptively.)

With FlashArray storage, backup times become considerably shorter compared to those on disk-based storage due to the arrays' typical sub-millisecond I/O latency. An incrementally updating backup that takes 30 minutes with disk-based storage might complete in as little 5 minutes when written to a FlashArray.

Second, the FlashReduce feature includes three *data reduction* techniques that minimize physical storage requirements as well as contributing to enhanced performance:

Pattern Removal
Arrays replace sectors that contain a single byte pattern repeated 512 times with metadata that describes them.

Data compression
Arrays use a 2-stage scheme consisting of a low-overhead LZO as data enters the array, and in the background, a more exhaustive LZO+Huffman for data that remains unmodified for an extended period. This has the advantages of (a) minimizing host I/O latency, (b) minimizing the physical "footprint" of long-lived data, and (c) expending minimal processing power on short-lived temporary data.

Global deduplication
Arrays search for duplication in incoming data sector by sector, regardless of volume location and alignment in LBA space. As

with compression, they use a low-overhead process on incoming data to minimize host I/O latency, and a more exhaustive background one for long-lived data to minimize physical storage utilization.

By minimizing the footprint of stored data, FlashReduce further improves performance by minimizing the amount of "back end" I/O an array performs. Simply put: reading and writing less data takes less time, no matter what the technology.

Figure 5 illustrates incrementally updating backup storage consumption with FlashArray volumes used for the **+DATA** and **+FRA** ASM Disk Groups (or file systems). The upper (blue) row represents the host and DBA view; the lower (orange) one, physical storage consumption on the FlashArray.

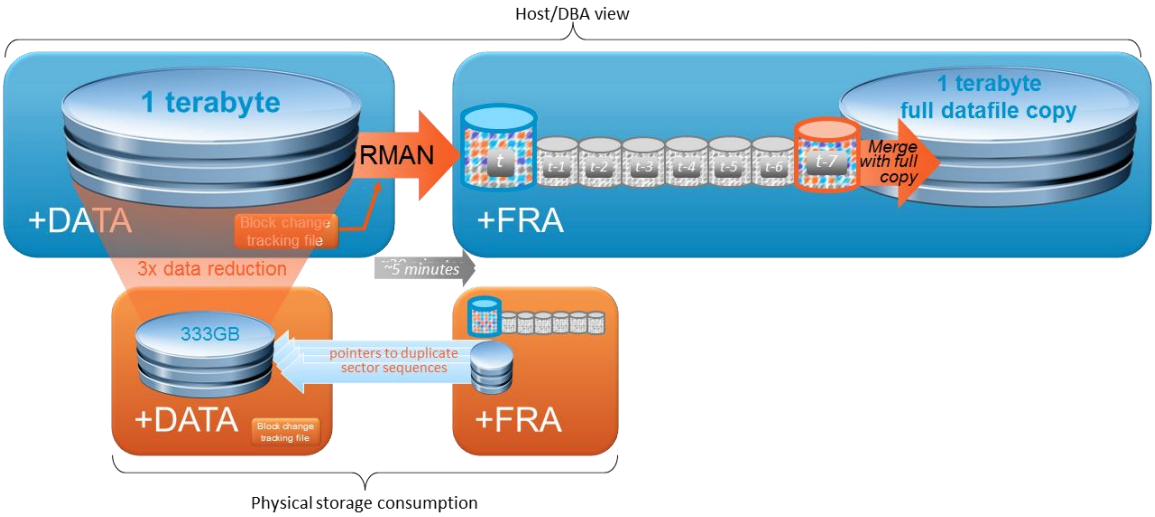

Figure 5: RMAN Incrementally Updating Backup with FlashArray Storage

First, pattern removal, compression, and deduplication reduce the physical storage occupied by the source database itself. Alphanumeric databases stored on FlashArrays typically reduce by factors of 2 to 4; the example shows a factor of 3, making the storage occupied by the **+DATA** Disk Group's volume 333 gigabytes. FlashArray volumes are dynamically thin-provisioned; the storage they occupy increases and decreases as data is added to and removed from them.

Next, the datafile copy in the **+FRA** Disk Group volume is initially an exact copy of the source database, so FlashReduce deduplicates the copied blocks, even though from the host/DBA point of view the two are on different volumes. The "full datafile copy" visible to hosts and DBAs actually occupies very little physical space in the **+FRA** volume. As the source database is updated, the space occupied by the copy increases, but it contracts again as merges of incremental backup data bring the contents of source and copy into closer alignment.

To DBAs, the datafile copy appears to occupy a full terabyte. It can be used to restore source database datafiles. Appendix A illustrates the DBA and FlashArray administrator views of storage consumption during incrementally updating backup.

For a 1 terabyte alphanumeric database incrementally backed up daily, 100 gigabytes is a conservative estimate of the physical storage required by the full datafile copy in the **+FRA** Disk Group volume. A week of incremental backups and archived redo logs (which FlashArrays also compress and deduplicate) might occupy as much as 150 additional physical gigabytes, for a total of 250 gigabytes. Thus, the **+FRA** to **+DATA** physical storage consumption ratio is 0.25 to 1—half that of a typical RMAN compressed backup. Moreover, this ratio is achieved along with the shorter backup times and lower CPU loading of the incrementally updating backup technique.

Most DBAs are familiar and comfortable with RMAN incrementally updating backup. Moving a database's ASM Disk Groups to FlashArray volumes is non-disruptive, requires no procedural changes, and delivers significant improvements in backup

time and storage consumption. By using FlashArrays' FlashRecover snapshot feature, however, database backup can be improved still further, as the following section demonstrates.

FlashArray Strategy 2: FlashRecover Snapshot-Based Backup[1]

FlashRecover snapshots capture instantaneous images of administrator-defined *protection groups* of FlashArray volumes, either on demand, or automatically on a periodic schedule. Snapshots are created entirely by the array; they consume no CPU power on the database host. Initially, they occupy no data space on the array. As hosts update blocks in the database, snapshots consume space to record the pre-change contents of the blocks.)

```
pureuser@pure> purevol snap --suffix ORADB1
                 asm_data asm_control_redo asm_fra
```

Figure 6: Example Command for Creating a FlashRecover Snapshot of Oracle Volumes

FlashRecover snapshots are *crash consistent*. They are often used as precautionary backups prior to certain Oracle operations—software upgrades, database structural changes, and major batch jobs. By themselves, however, they do not constitute a complete backup and recovery strategy, because each snapshot only captures volume contents at a single point in time. They do not include a journaling capability that would allow a snapshot to be rolled forward to a specific recovery point, as is done with RMAN recovery from database redo logs. FlashRecover snapshots are therefore complementary to RMAN backup strategies rather than replacing them entirely.

As the preceding section demonstrates, RMAN incrementally updating backups to FlashArray storage are extremely efficient. Pure Storage generally recommends them, in large part because Oracle DBAs are familiar with the techniques. But for sub-second backup duration with no CPU impact, a combination of careful database layout, some simple scripting, and well documented recovery procedures make it possible for FlashRecover snapshots to replace RMAN backup in a comprehensive database backup and recovery strategy.

FlashRecover snapshot backup has limitations. Before adopting a hybrid FlashRecover-RMAN backup strategy, DBAs and storage administrators should consider the following:

Database consolidation

Snapshot-based backups are *volume* backups; restoring them overwrites entire volumes. The strategy is therefore not suitable for consolidated (e.g., Schema as a Service) databases that serve multiple applications from a single volume. Such databases must be able to recover a single application schema, which is possible with RMAN *tablespace point in time recovery* (TSPITR) but not with FlashRecover snapshots. For these applications, Pure Storage recommends RMAN incrementally updating backup to FlashArray storage as described in Appendix A.

Advanced RMAN features

Some advanced RMAN features, such as block level recovery and block consistency checking, are not available with FlashRecover snapshot backups. However, FlashArrays perform robust error correction and end-to-end data integrity checks on the data they store, which limits the need for these features at the database level.

For single-application databases, however, a well-scripted backup and recovery strategy based on FlashRecover snapshots offers near-instantaneous backup with minimal physical storage consumption.

Storage Layout for FlashRecover Snapshot-Based Database Backup

Many Oracle installations share a single large **+DATA** and **+FRA** Disk Group among multiple databases. With disk-based storage this strategy is useful, because (a) it optimizes I/O performance by distributing load across many spindles, and (b) it minimizes storage consumption by using a single shared reserve rather a reserve per database. Neither of these considerations is relevant with FlashArray storage, however.

[1] Application Brief AB-160501, *Recovering Oracle Databases from FlashRecover Snapshots*, contains additional discussion of database recovery technques.

FlashArrays manage all of their storage as a single pool from which they allocate space for writing incoming data as it arrives, regardless of LUN. They constantly balance back-end I/O across all of their flash devices. I/O performance and storage consumption are therefore both independent of the virtual volume (LUN) structure exported to hosts. This allows storage administrators and DBAs to design database storage structures based on operational considerations rather than storage array constraints. Each database that uses the FlashRecover snapshot backup strategy should have its storage allocated on three *protection groups* (administrator-defined groups of FlashArray volumes):

Datafiles
All datafiles should be located on volumes in one protection group. This section uses a single **+DATA** volume, but the protection group can contain more volumes, for example to overcome Oracle maximum device size limitations.

Control files and online redo logs
These should be located on volumes (usually one) in a separate protection group.

Fast Recovery Area
Archived redo logs and database backups should be located on volumes in this protection group. One FlashArray volume is ideal for this (maximum virtual volume size is 4 petabytes), but Oracle maximum device size limitations may require additional ones.

Figure 7 illustrates this database storage layout, with the blue rectangles representing FlashArray protection groups mapped to ASM Disk Groups (or host file systems). Each protection group may contain one or more volumes, based on operational requirements.

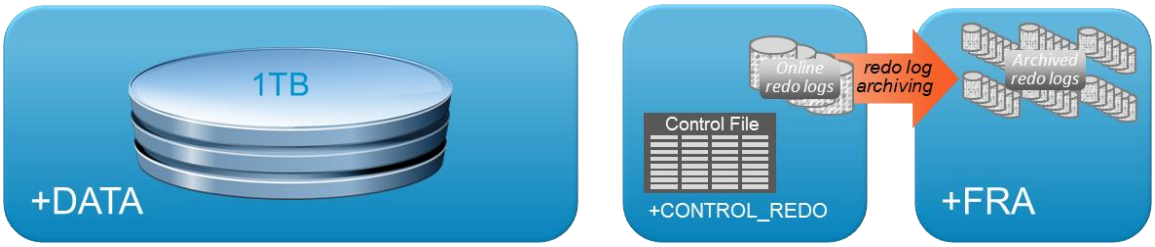

Figure 7: Database Layout for FlashRecover Snapshot-Based Backup

Separating database components like this is key to the FlashRecover snapshot backup strategy. Recovering a database to a specific point in time (including to the most recent SCN) starts with restoring a crash-consistent snapshot of its datafiles, followed by rolling forward using up-to-date control files, online and archived redo logs. Volumes restored from a datafile protection group snapshot provide a crash-consistent starting point for RMAN recovery, which rolls the restored datafiles forward to the desired recovery point using up-to-date online redo logs and archived logs.

Using FlashRecover Snapshots for Database Backup

Figure 8 illustrates a FlashRecover snapshot-based backup scenario. Again, the blue row represents the host and DBA view of storage, in which everything appears at its nominal size. The orange row represents FlashArray physical storage consumption. Daily FlashRecover snapshots are retained for a week. Archived redo logs are retained for a week as well.

In this scenario, FlashReduce compression and deduplication store the 1 terabyte alphanumeric database in 333 gigabytes of physical storage in the **+DATA** protection group. Assuming that 5% of database blocks (50 gigabytes) change daily:

- Snapshots consume a maximum of 7 days x 50 gigabytes/day =350 gigabytes in the **+DATA** ASM Disk Group
- Archived redo logs consume about 7 days x 50 gigabytes/day = 350 gigabytes in the **+FRA** ASM Disk Group
- Online redo logs and control files consume about 60 gigabytes in the **+CONTOL_REDO** ASM Disk Group

Almost all of these are database blocks, and so can reasonably be expected to reduce similarly to datafiles, thus steady-state physical storage consumption would be about (350 + 350 + 60) ÷ 3 ≈ 250 gigabytes.

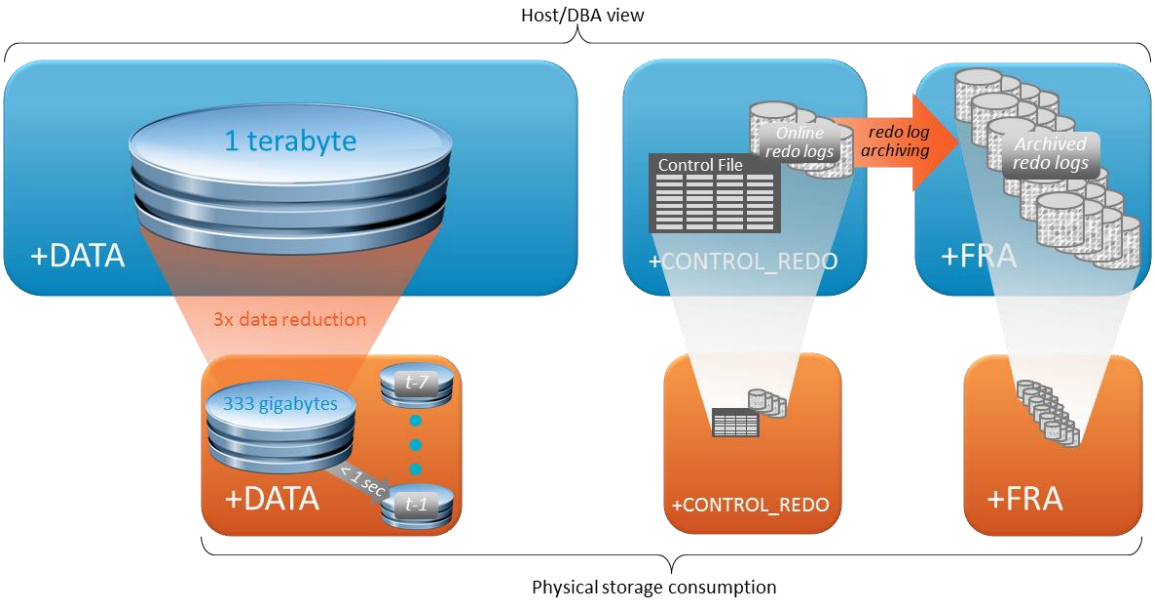

Figure 8: Storage Consumption Using FlashRecover Snapshots for Database Backup

To recover lost or corrupted datafiles to a specific point in time:

- The FlashArray administrator copies the newest snapshot preceding the desired recovery point, overwriting the (presumably corrupted) volumes. The "copy" is nearly instantaneous; internally it amounts to adjusting metadata.
- The DBA performs an RMAN point-in-time recovery, using the restored datafiles, and the up-to-date logs and control files, whose protection groups have ***not*** been copied.

Appendix B contains an example of a FlashRecover snapshot-based Oracle database backup and recovery.

Coordinating Archived Log and Snapshot Retention

The FlashRecover snapshot-based backup strategy requires coordinated archived redo log and snapshot retention. For example, if daily snapshots are retained for a week as the above example, redo logs must also be retained for a week to make database recovery to any point in time within the week possible. RMAN and FlashArray administration are not mutually aware, so DBA and storage administrator must collaborate to coordinate the two parts of the strategy.

Appendix A: Incrementally Updating Backup with FlashArrays

This appendix illustrates the DBA's and FlashArray administrator's respective views of storage consumption before and after an incrementally updating backup of a database that resides on FlashArray volumes. Figure A1 illustrates an Oracle Enterprise Manager (OEM) report of the storage consumed by a sample database. From a DBA perspective, the database's **DATA** ASM Disk Group contains 557.34 gigabytes of data, occupying 54.43% of the 1 terabyte allocated to the group.

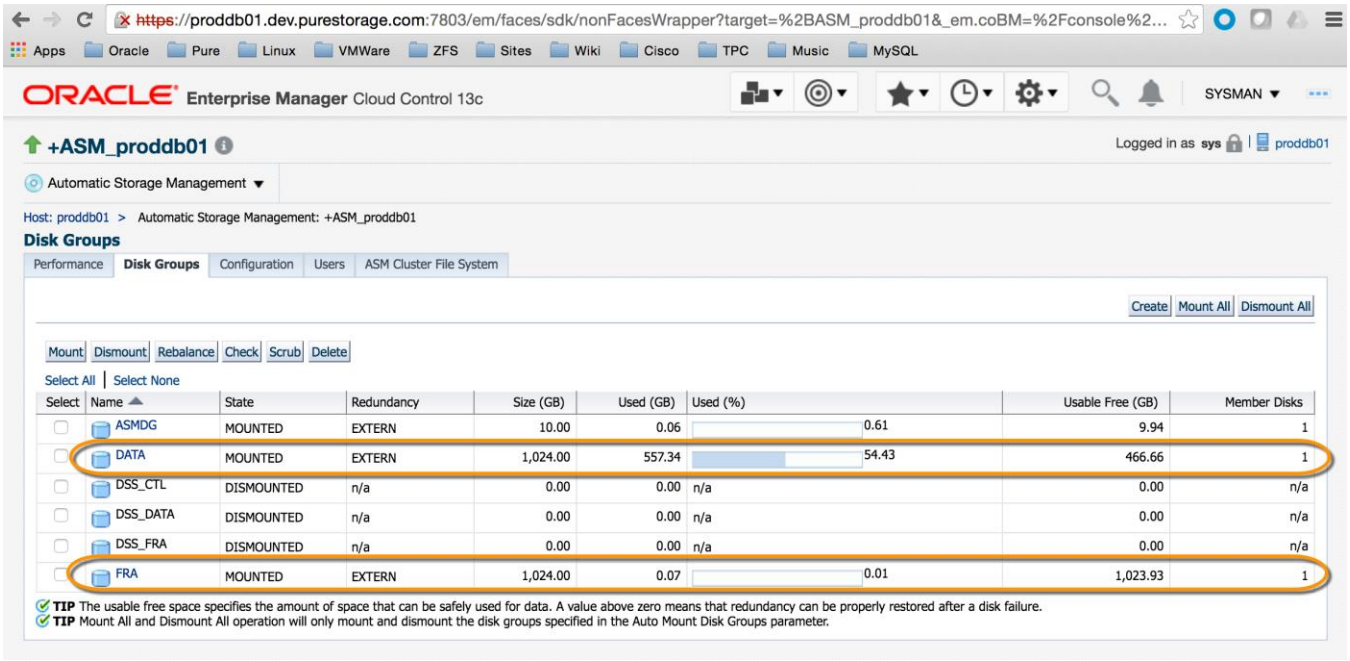

Figure A1: OEM View of Database Storage Consumption (Before Backup)

Figure A2 shows the upper pane of the FlashArray GUI **Capacity** pane, representing the storage occupied by all of the array's contents—the example database's data, log, and control files, as well as any other unrelated data on the array. In the upper right area of the bar, the array reports overall data reduction ratio of 3.4 to 1.

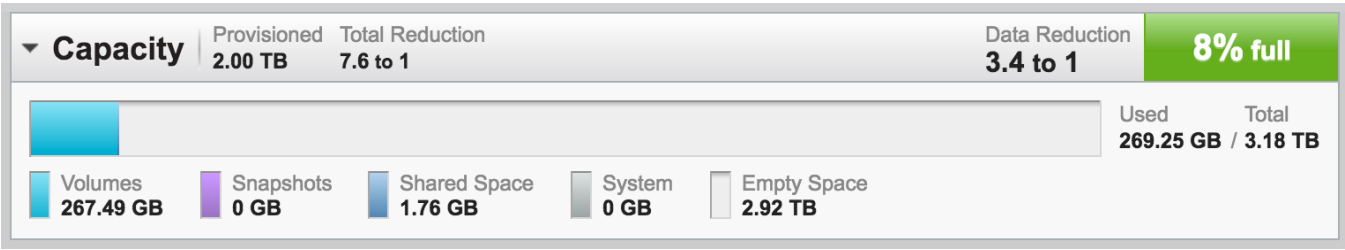

Figure A2: Array-Wide Physical Storage Consumed by Reduced Data (Before Backup)

*The upper left of Figure A2 shows **Total Reduction** of 7.6 to 1. **Total Reduction** is the ratio of provisioned storage (the sum of the sizes of all volumes as seen by hosts) to the physical storage occupied by data in the array after reduction. **Data Reduction**, the number of unique sector addresses to which hosts have written data divided by the amount of physical storage the data occupies, is a more accurate indicator of how much additional data of a similar character an array might be capable of storing.*

Figure A3 shows the upper pane of the GUI drill down view of the database's datafile volume, **db_ora_data**. The 557.34 virtual gigabytes in use in the volume occupy 267.49 gigabytes of physical storage (reported as **Volumes** in the display).

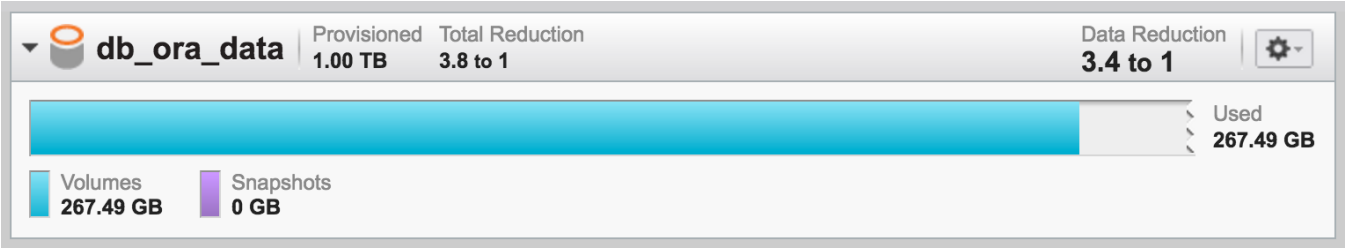

Figure A3: FlashArray Administrator View of Data Volume Storage Consumption (Before Backup)

*The array used for this example hosts multiple volumes. Because FlashArray deduplication is array-wide, sectors that hosts write to the **db_ora_data** volume may be duplicates of sectors on unrelated volumes. Arrays report space occupied by data shared by two or more volumes (**Shared Space** in Figure A2), but there is no computationally feasible way to apportion it to individual volumes. This partly accounts for the difference between the reported **Data Reduction** of 3.4 to 1 and the result of dividing the sum of the 557.34 gigabytes of database data files reported in Figure A1 by the 267.49 gigabytes of physical storage reported as **Volumes** in Figure A3 (about 2.08).*

Figure A4 shows the upper pane of the GUI drill down view of the database's **db_ora_fra** volume. Prior to backup, this volume only contains archived redo logs (about 70 megabytes of online and archived logs reduced to 30.39 megabytes of physical storage. Again, the reported reduction of 4.5 is influenced by unrelated data in the array).

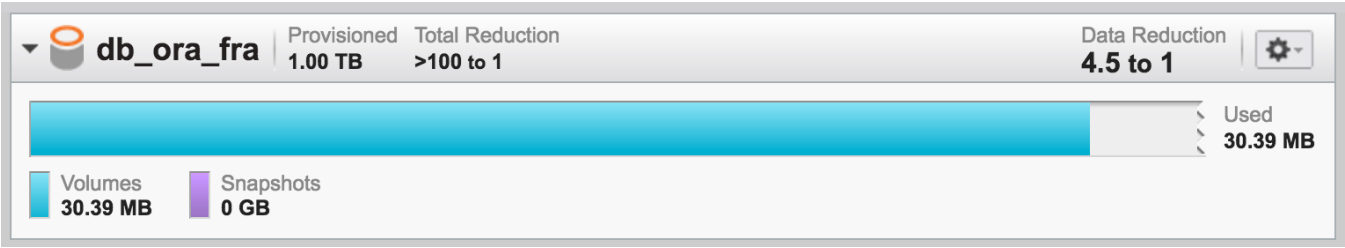

Figure A4: FlashArray Administrator View of Database Flash Recovery Area (Before Backup)

Figure A5 shows the elapsed time for an RMAN incrementally updating backup of the example database's datafiles, as might be performed by a script similar to that shown in Figure 3 of the main body of the report. From a DBA standpoint, the backup copies 555.18 gigabytes in 24 minutes and 17 seconds, for a reported transfer rate of 390.19 megabytes/second.

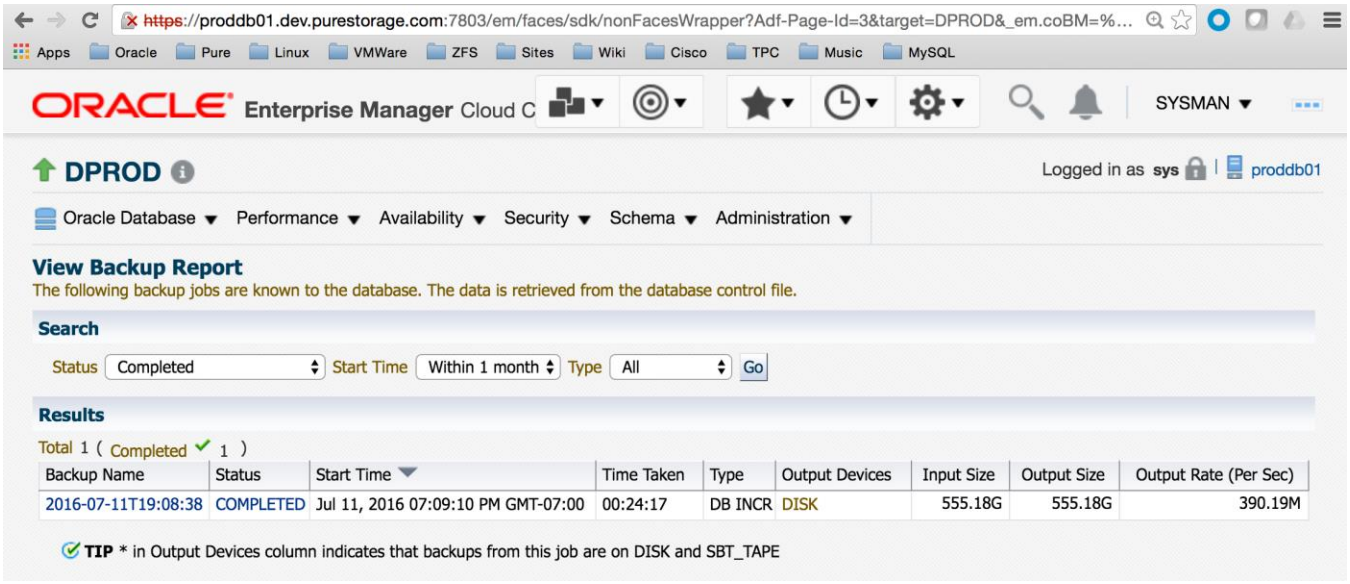

Figure A5: OEM Backup Execution Time Report for Example Database Datafiles

Figure A6 shows the storage consumption reported by OEM after an RMAN backup of the database's datafiles. Because no full backup had as yet existed, the script performed a full datafile copy. Because RMAN incrementally updating backups do not compress data, the full copy appears to occupy about the same amount of storage as the source database's datafiles; the two would differ only by updates to source database blocks that RMAN had already copied. (The difference shown in the Used column of Figure A6 is accounted for by archived redo logs stored in the FRA disk group.

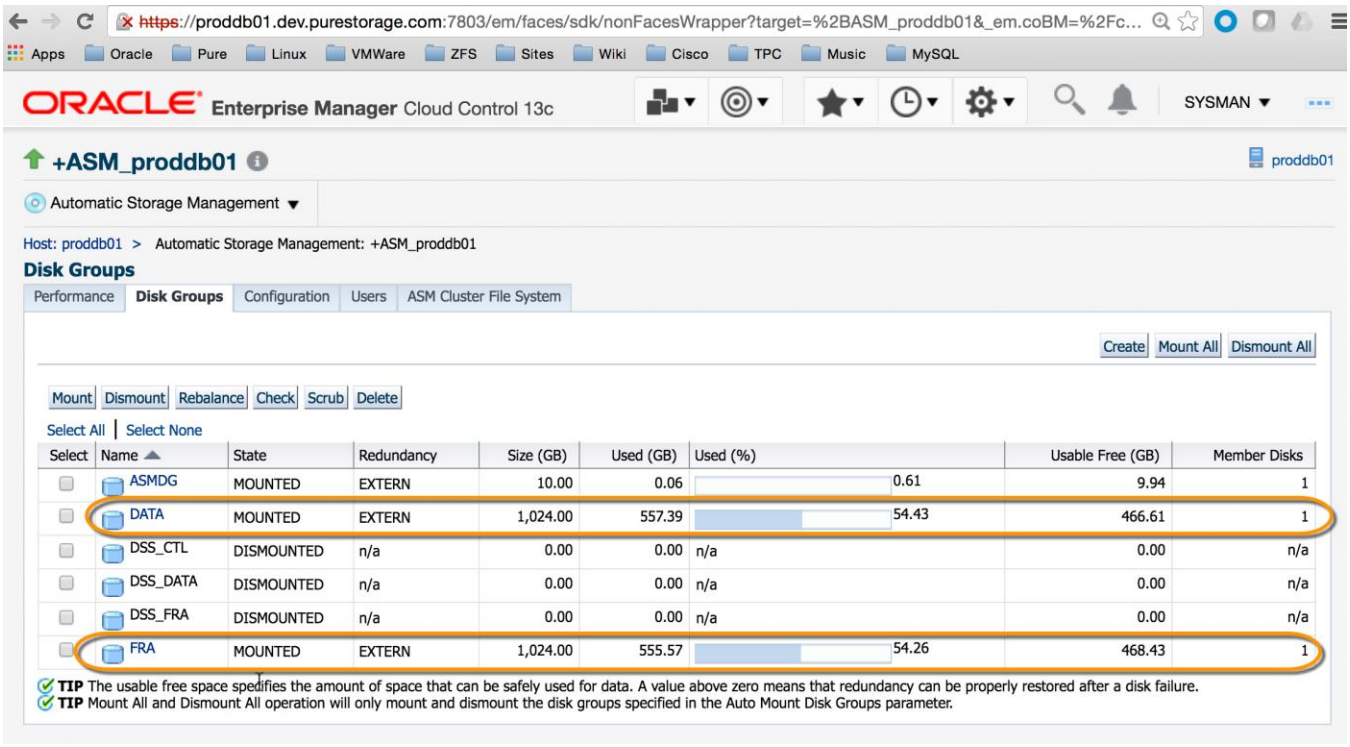

Figure A6: OEM Report of Database Storage Consumption (After Backup)

The database's FRA represented in Figure A6 contains a full uncompressed copy of the datafiles as well as archived redo logs, for a total of 555.57 gigabytes.

Figure A7 shows the physical storage consumed by data in the **db_ora_fra** volume after the RMAN backup. The array reduced the 555.19 gigabytes that RMAN copied to 27.52 gigabytes of physical storage.

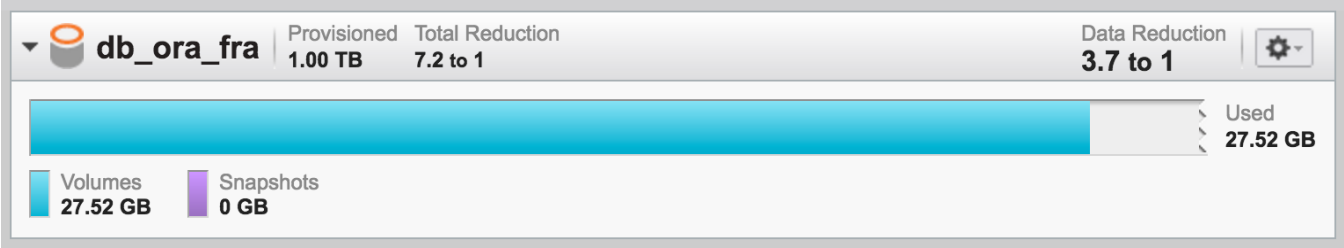

Figure A7: FlashArray Administrator View of Database Flash Recovery Area (After Backup)

Most of the blocks in the uncompressed backup copy are identical to blocks in the source database. The array deduplicates the two against each other, even though they are on separate volumes. Figure A8 shows the array-wide storage consumption after the RMAN full uncompressed datafile backup.

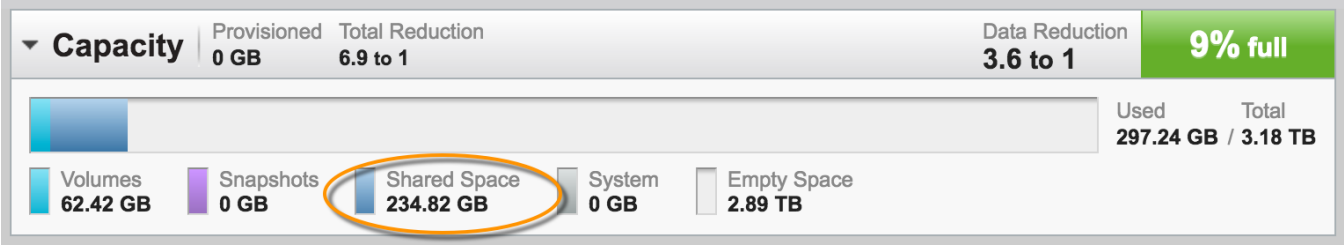

Figure A8: Array-Wide Physical Storage Consumed by Reduced Data (After Backup)

A comparison of Figure A8 and Figure A2 indicates that copying 555.19 gigabytes of datafiles has increased the array's overall physical storage consumption (labeled **USED** on the right side of the figures) by only 27.99 gigabytes (from 269.25 to 297.24 gigabytes). The array has deduplicated a significant amount of data as RMAN copied it. Two factors can account for most of the incremental physical storage consumption observed after an initial RMAN incrementally updating backup makes a full datafile copy:

Updates to the source database

As an RMAN backup job runs, applications may be updating the source database. Source database blocks that RMAN has copied before they are updated are no longer identical to the RMAN copies, so the array retains the original content and charges it to the copy as it stores the new content in the source database.

2-stage data reduction

FlashArrays reduce data in two stages. To minimize host I/O latency, they deduplicate incoming data only against the most likely duplication candidates, and use a low-overhead LZO compression algorithm. As data remains unaltered, background tasks deduplicate and compress it more exhaustively outside the host I/O path. As a consequence of two-stage data reduction, reported storage consumption is necessarily approximate—it can increase or decrease due to incoming data, host unmapping, and background deep reduction.

At the time of the screen capture in Figure A8, reduction of the data written by RMAN backup has increased the array's overall data reduction ratio from 3.4 to 1 to 3.6 to 1. Again, because deduplication is array-wide, the ratio may reflect the influence of data on unrelated volumes.

Deduplication also partially explains the high apparent I/O rate (and the resulting short backup duration) reported in Figure A5. A significant amount of the data that RMAN "writes" is not stored. The array replaces it with metadata that points to already-stored identical sectors from the source database.

RMAN backup data that the array deduplicates is shared by two volumes, **db_ora_data** (Figure A3) for source database blocks and **db_ora_fra** (Figure A7) for deduplicated blocks in the RMAN backup copy. FlashArrays report space occupied by data shared by two or more volumes as array-wide **Shared Space** (234.82 gigabytes in Figure A8); they do not attempt to apportion it among the sharing volumes. Because much of the physical storage is now shared, however, it is no longer attributed to the individual volumes, as Figure A9 illustrates—after the backup, the **db_ora_data** volume is charged with 34.90 gigabytes of physical storage; the remainder is accounted for in the **Shared Space** reported in Figure A8.

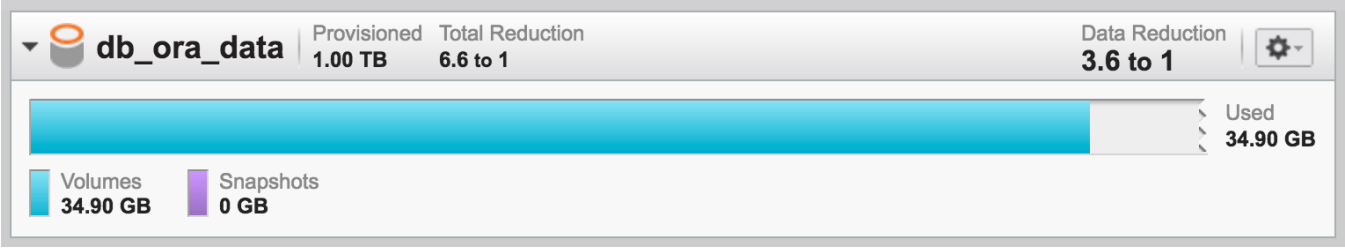

Figure A9: FlashArray Administrator View of Data Volume Storage Consumption (After Backup)

Recovery

To recover corrupted files for a database located on conventional storage, an administrator would replace them by copying backups that predate the corruption from the **FRA** to the **DATA** disk group and using RMAN to roll them forward. This can be time-consuming, principally because FRAs are usually located on low-cost (lower-performing) storage. For databases located on FlashArrays, copying is many times faster, due both to FlashArrays' high performance, and to data reduction that minimizes the number of bytes to be moved. But when FlashArray storage is used for both DATA and FRA disk groups, an even faster recovery option is available—corrupt files can be switched out and replaced with the intact copies in the FRA, which are immediately recoverable using RMAN, as the example in Dialog A1 illustrates.

```
ALTER DATABASE DATAFILE 4 OFFLINE;
SWITCH DATAFILE 4 TO COPY;
RECOVER DATAFILE 4;
ALTER DATABASE DATAFILE 4 ONLINE;
```

Dialog A1: Switching to a Copy of a Corrupted Database File

The time to switch out `DATAFILE 4` in this example is essentially the time to enter the commands. The datafile is online again as soon as the RMAN `RECOVER` operation rolls it forward to incorporate the most recent transactions.

Summary

With incrementally updating database backups to FlashArray storage, all backup and recovery constructs and operations are native Oracle. To the DBA, storage appears as conventional numbered logical units (LUNs). Reduced storage requirements and shorter backup times are entirely due to FlashArray data reduction and I/O performance. DBAs back up and recover using familiar RMAN commands, with all the flexibility they provide, but backup times are shorter and storage consumption is lower due to the FlashArray technology. Combining Oracle incrementally updating backup with FlashArray storage reduces backup times, storage requirements, and CPU loading significantly, and in addition, shorter recovery times make it possible for a DBA to adopt more aggressive database recovery time objectives (RTOs).

Appendix B: Using FlashRecover Snapshots for Oracle Backup

This appendix contains an example of a procedure for creating ASM Disk Groups and a database for use with FlashRecover snapshot-based backup, and illustrates restoring and recovering datafiles.

> ***Note**: If the Oracle host server is a VMware guest virtual machine, LUNs should be presented to it as Raw Device Mappings (RDM's).*

Creating FlashArray Volumes

The first step of the procedure is to use the FlashArray CLI or GUI to create a volume configuration similar to that illustrated in Figure 7 on page 12. (This example assumes that a single 1 terabyte volume for each Disk Group is adequate, so protection groups are not required. If size limitations or other operational considerations require multiple volumes, a protection group should be created for whichever of datafiles, control files and online redo logs, and fast recovery area require multiple volumes.) Dialog B1 shows the FlashArray CLI commands for creating the volumes.

```
pureuser@purearray> purevol create --size 1T dg_dprod_oradata
pureuser@purearray> purevol create --size 1T dg_dprod_oraredo
pureuser@purearray> purevol create --size 1T dg_dprod_orafra
```

Dialog B1: CLI Commands to Create FlashArray Volumes for the Database

The volumes used in this example all have a size of 1 terabyte. FlashArray volumes are dynamically thin-provisioned; the physical storage they occupy grows and shrinks as data is added to and removed from them. When volumes are first created, they do not start to consume physical storage until hosts write data to them.

A FlashArray generates a unique serial number for each volume it creates. Volume serial numbers are needed to map volumes to ASM devices. After the new volumes have been connected to the database hosts (not illustrated), the administrator uses the **`purevol list`** CLI command, as shown in Dialog B2, to determine the volumes' serial numbers, which must be supplied when creating ASM Disk Groups.

```
pureuser@purearray> purevol list
Name              Size  Source  Created                  Serial
dg_dprod_oradata  1T    -       2016-06-27 16:18:31 PDT  2163F6A1D60810ED0001102B
dg_dprod_orafra   1T    -       2016-06-27 17:01:41 PDT  2163F6A1D60810ED0001102D
dg_dprod_oraredo  1T    -       2016-06-27 17:01:34 PDT  2163F6A1D60810ED0001102C
```

Dialog B2: Using the FlashArray CLI to Display Serial Numbers of Database Volumes

Alternatively, serial numbers can be displayed by the GUI in the **Volumes** view, as Figure B1 below illustrates.

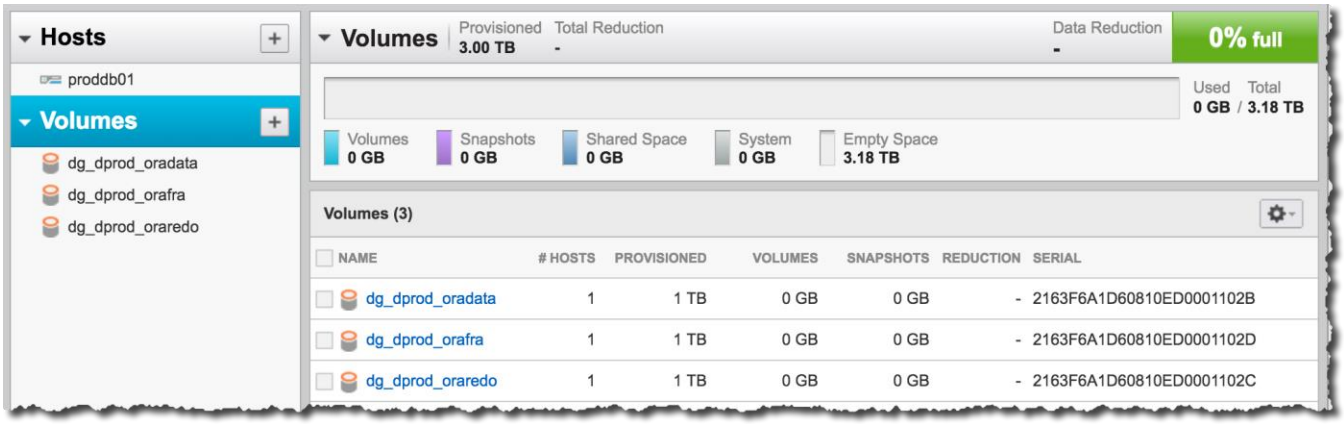

Figure B1: Using the FlashArray GUI to Display Serial Numbers of Database Volumes

SCSI storage device serial numbers are universally unique. Each one starts with a 36 bit (9 hexadecimal digit) vendor ID. Because the vendor ID is always the same for FlashArray volumes (3624a9370), it is omitted from the CLI and GUI displays to conserve display space.

Creating and Populating the ASM Disk Group

Dialog B3 shows statements that can be added to a Linux host's /etc/multipath.conf file to associate the volumes identified by serial number with the aliases . The statements must include the entire serial number, including the vendor ID portion (3624a9370 for Pure Storage, highlighted in gray in the dialog).

```
multipaths {
      multipath {
              wwid            3624a93702163f6a1d60810ed0001102b
              alias           dg_dprod_oradata
      }
      multipath {
              wwid            3624a93702163f6a1d60810ed0001102c
              alias           dg_dprod_oraredo
      }
      multipath {
              wwid            3624a93702163f6a1d60810ed0001102d
              alias           dg_dprod_orafra
      }
}
```

Dialog B3: **`/etc/multipath.conf`** Statements to Associate Volumes with Aliases

The aliases specified in Dialog B3 need not be identical to the volume names specified Dialog B1, but it is good practice to make the two identical for ease of identification, particularly in databases that use greater numbers of volumes for storage.

After editing the /etc/multipath.conf file, the administrator must restart the multipath daemon to cause the LUNs to be recognized, as illustrated in Dialog B4. Dialog B4 also lists the volumes, now identified by the aliases assigned in Dialog B3, as they appear in Linux Logical Volume Manager (LVM) paths. The aliases are used to identify the volumes in operations from this point forward.

```
[root@proddb01 ~]# systemctl restart multipathd.service
[root@proddb01 ~]# ls -ltr /dev/mapper/dg_dprod*
lrwxrwxrwx 1 root root 8 Jun 27 19:35 /dev/mapper/dg_dprod_oradata -> ../dm-28
lrwxrwxrwx 1 root root 8 Jun 27 19:35 /dev/mapper/dg_dprod_oraredo -> ../dm-31
lrwxrwxrwx 1 root root 8 Jun 27 19:35 /dev/mapper/dg_dprod_orafra -> ../dm-30
```

Dialog B4: Linux LVM Device Paths of Database Volumes

Dialog B5 shows the SQL commands for creating separate ASM Disk Groups to hold the database's datafiles, control files and redo logs, and fast recovery area. The commands also add the volumes created in Dialog B1 as ASM disks to the respective disk groups.

```
SQL> create diskgroup DATA external redundancy disk '/dev/mapper/dg_dprod_oradata' size 1T
attribute 'compatible.asm'='12.1.0.0.0','au_size'='1M';
SQL> create diskgroup CONTROL_REDO external redundancy disk '/dev/mapper/dg_dprod_oraredo' size
1T attribute 'compatible.asm'='12.1.0.0.0','au_size'='1M';
SQL> create diskgroup FRA external redundancy disk '/dev/mapper/dg_dprod_orafra' size 1T
attribute 'compatible.asm'='12.1.0.0.0','au_size'='1M';
```

Dialog B5: Creating ASM Disk Groups and Adding Disk to Them

The `external redundancy` parameter in the commands in Dialog B5 is particularly worth noting. By default, Oracle ASM mirrors the disks in a disk group. ASM Disk Groups on FlashArray storage should specify `external redundancy`, however, because (a) FlashArray RAID-3D provides more comprehensive protection than mirroring, and (b) the arrays deduplicate data across volumes, so Oracle mirroring of ASM disks incurs the performance burden of writing everything twice, but delivers no additional data protection because the array deduplicates the second copy and protects the first with RAID-3D.

DBAs must run the ASRU utility periodically to reclaim deleted files' storage, particularly automatically deleted archived logs. FlashArrays process unmap commands by voiding the unmapped sector ranges of their cbmap structures, so command latency is negligible. Actual storage reclamation occurs during background garbage collection.

Adding Volumes to ASM Disk Groups (*if necessary*)

If Oracle constraints or organizational considerations require the addition of volumes to a ASM Disk Group, the FlashArray administrator would create the volume(s) and determine their serial numbers as shown in Dialogs B1 and B2. The host administrator or DBA would associate them with aliases as shown in Dialogs B3 and B4, and the DBA would add them to the appropriate ASM Disk Groups, as illustrated in Dialog B6.

```
SQL> alter diskgroup DATA add disk '/dev/mapper/dg_dprod_oradata02' size 1T;
```

Dialog B6: Adding a Volume to an ASM Disk Group

Creating the Database

For a database to use FlashRecover snapshot backup for point-in-time restore, the **+DATA** Disk Group must contain only datafiles. When using the Oracle Database Configuration Assistant (DBCA) to configure database storage, the simplest way to do this is to place all database files in a common location (the **+CONTROL_REDO** Disk Group) initially, and manually move the datafiles to the **+DATA** Group toward the end of the configuration procedure. Figures B2 and B3 illustrate the DBCA setup wizard steps in which the DBA specifies the database storage configuration.

*If Grid Infrastructure will be used for RAC or Oracle Restart, the **+DATA** Disk Group must not contain Grid Infrastructure-related files such as the Oracle Cluster Registry (OCR) or Voting Disk files. These should be placed in the **+CONTROL_REDO** group.*

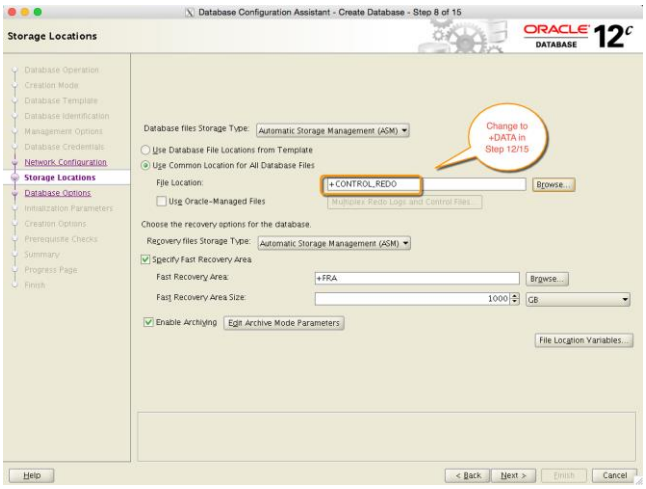

Figure B2: DBCA: Specifying a Temporary Common Location for All Database Files, Enabling Archiving and Specifying the FRA Location

Figure B2 illustrates the wizard step that specifies placement of all database files in the **+CONTROL_REDO** Disk Group (temporarily). The same step specifies use of a Fast Recovery Area (a near-universal practice among Oracle DBAs) located in the **+FRA** Disk Group. The step also enables redo log archiving (by checking the **Enable Archiving** box).

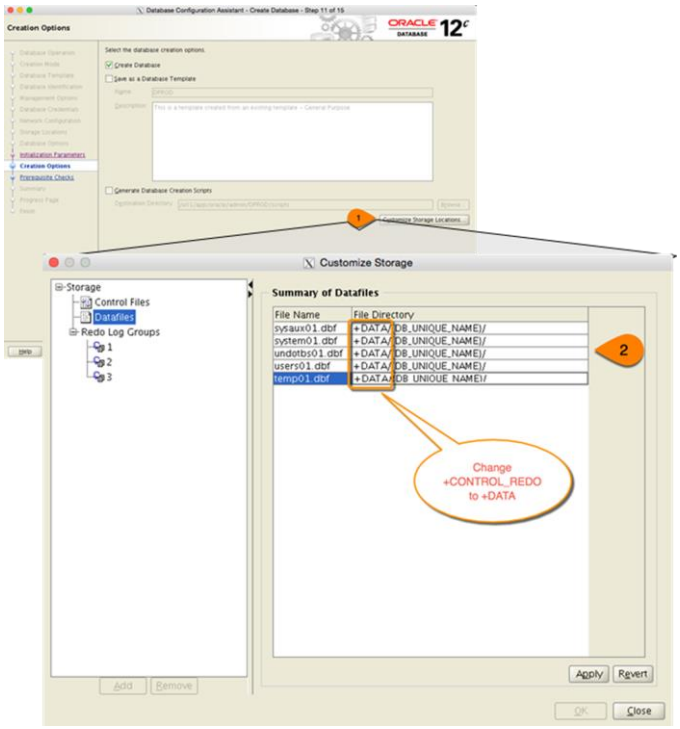

Figure B3: DBCA: Relocating Datafiles to the **+DATA** ASM Disk Group

Figure B3 illustrates the wizard step used to change the location for datafiles to the **+DATA** Disk Group. All other files remain in the **+CONTROL_REDO** Disk Group as specified in Figure B2.

After DBCA creates the database, the DBA must alter its parameters so that the database management system places any datafiles and redo log members that it creates in the future in the correct ASM Disk Groups (**+DATA** for datafiles and **+CONTROL_REDO** for redo logs). Dialog B7 shows the DBA commands to accomplish this.

```
SQL> alter system set db_create_file_dest='+DATA' scope=both;

System altered.

SQL> alter system set db_create_online_log_dest_1 = '+CONTROL_REDO' scope=both;

System altered.

SQL> alter system set control_file_record_keep_time=30  scope=both;

System altered.
```

Dialog B7: SQL Commands to Specify Placement of Any Future Database Files

The third command in Dialog B7 also sets the retention time for control file records at 30 days.

Once a database has been created, it is a best practice to verify that all of its datafiles, temporary files, control files, redo logs, and the SP file are placed as required by the FlashArray snapshot backup strategy. It is particularly critical that datafiles be located on separate volumes from control files and both online and archived redo logs. Figure B8 illustrates SQL commands for verifying file locations for the database in this example.

```
SQL> select name from v$datafile;

NAME
------------------------------------------------------------
+DATA/DPROD/system01.dbf
+DATA/DPROD/sysaux01.dbf
+DATA/DPROD/undotbs01.dbf
+DATA/DPROD/example01.dbf
+DATA/DPROD/users01.dbf

SQL> select name from v$tempfile;

NAME
------------------------------------------------------------
+DATA/DPROD/temp01.dbf

SQL> select name from v$controlfile;

NAME
------------------------------------------------------------
+CONTROL_REDO/DPROD/control01.ctl
+CONTROL_REDO/DPROD/control02.ctl

SQL> select member from v$logfile;

MEMBER
------------------------------------------------------------
+CONTROL_REDO/DPROD/redo04.log
+CONTROL_REDO/DPROD/redo03.log
+CONTROL_REDO/DPROD/redo02.log
+CONTROL_REDO/DPROD/redo01.log
SQL> show parameter spfile

NAME                 TYPE     VALUE
-------------------- -------- ------------------------------
spfile               string   +CONTROL_REDO/DPROD/PARAMETERFILE/
                               spfile.263.915668653
```

Dialog B8: Verifying Database File Locations

Creating and Populating a FlashArray Protection Group

FlashArrays take snapshots of individual volumes or of *protection groups* of one or more volumes. This example uses a protection group containing volumes **dg_dprod_orafra** and **dg_dprod_oraredo**. Because there is only one datafile volume, it can be snapped directly, so creating a protection group for it is optional, but is a best practice.

To create a protection group, the FlashArray administrator displays the **PROTECTION** tab of the FlashArray GUI and clicks the + to the right of the **Source Groups** heading in the navigation pane. This displays a **Create Protection Group** dialog in which the administrator enters a name for the group and clicks the **Create** button.

> *It is a best practice to name protection groups in a way that identifies them with the database and ASM Disk Groups they will snap. (The example uses **dprod_util_PG**.)*

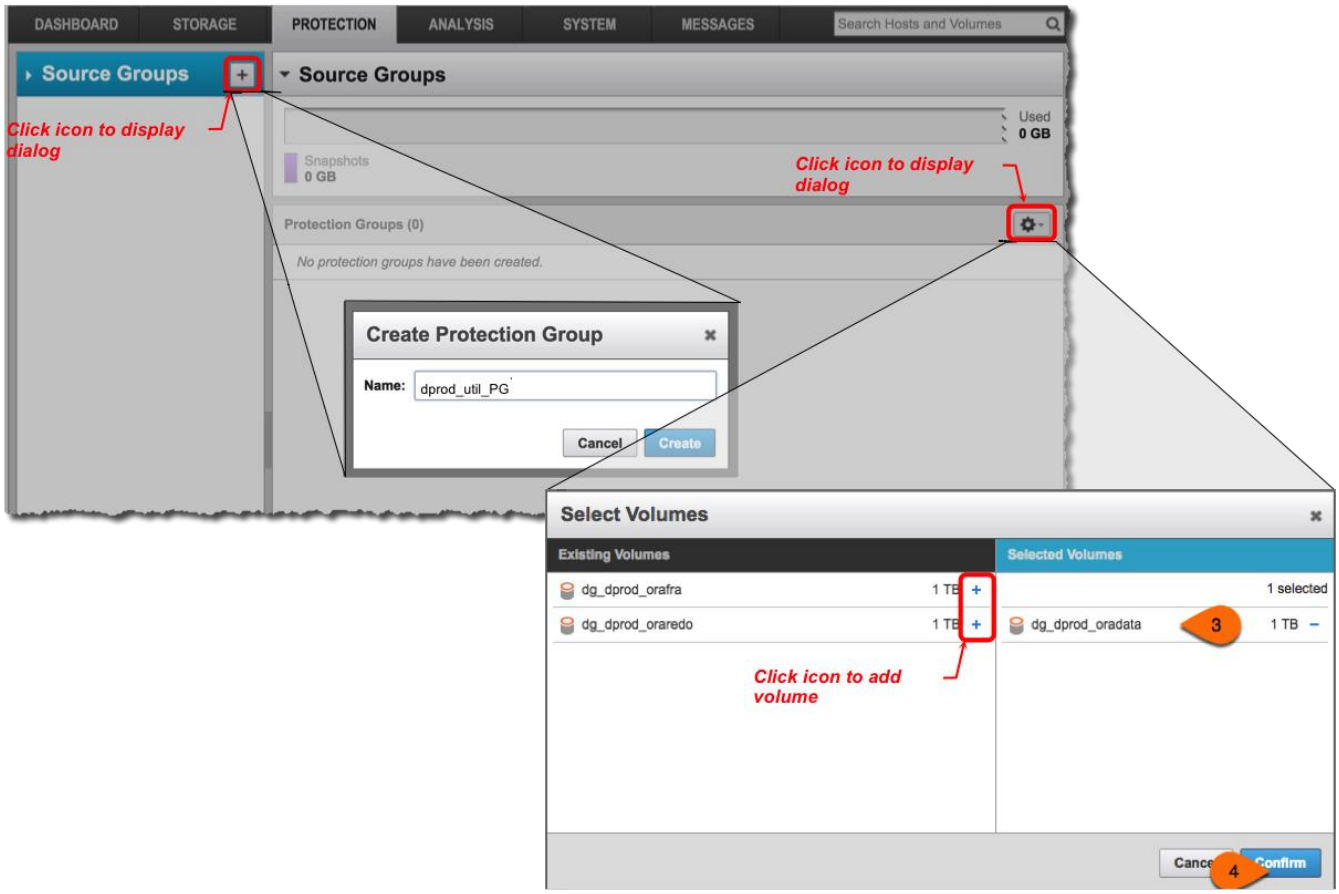

Figure B4: Creating a Protection Group for **db_dprod_orafra** and **db_dprod_oraredo** Volumes

To populate the protection group with volumes, select it, use the ⚙ icon to pull down the menu, and select **Add Volumes**. From the list displayed in the **Select Volumes** pop-up dialog, select the volumes to be added to the group (**dg_dprod_orafra** and **dg_dprod_oraredo** in this example), and click **Confirm**.

> *If the **+DATA** Disk Group contains more than one volume due to Oracle size limitation or for other operational reasons, be sure to add all volumes to its protection group to guarantee that snapshots capture all datafile contents as of the same point in time.*

Scheduling Protection Group Snapshots

FlashArray administrators customize snapshot schedules to meet RPO and RTO requirements. To customize a snapshot schedule, the administrator selects the protection group, uses the ⚙ icon to pull down the menu, and selects **Schedule**, which displays the pane shown in Figure B5.

The text boxes can be used to change the default hourly schedule to meet application-specific RPO and RTO requirements. The snapshot schedule shown in the figure specifies hourly snapshots of the **dprod_data_PG** group to be retained for a day (24 hours) from the time of creation. One snapshot per day (the "daily") is retained for an additional 7 days. When this schedule has been in effect for a week, the FlashArray contains:

- 24 hourly snapshots representing the state of the database's datafiles at each of the most recent 24 hours
- 7 daily snapshots representing the state of the database's datafiles at one point in each of the past 7 days.

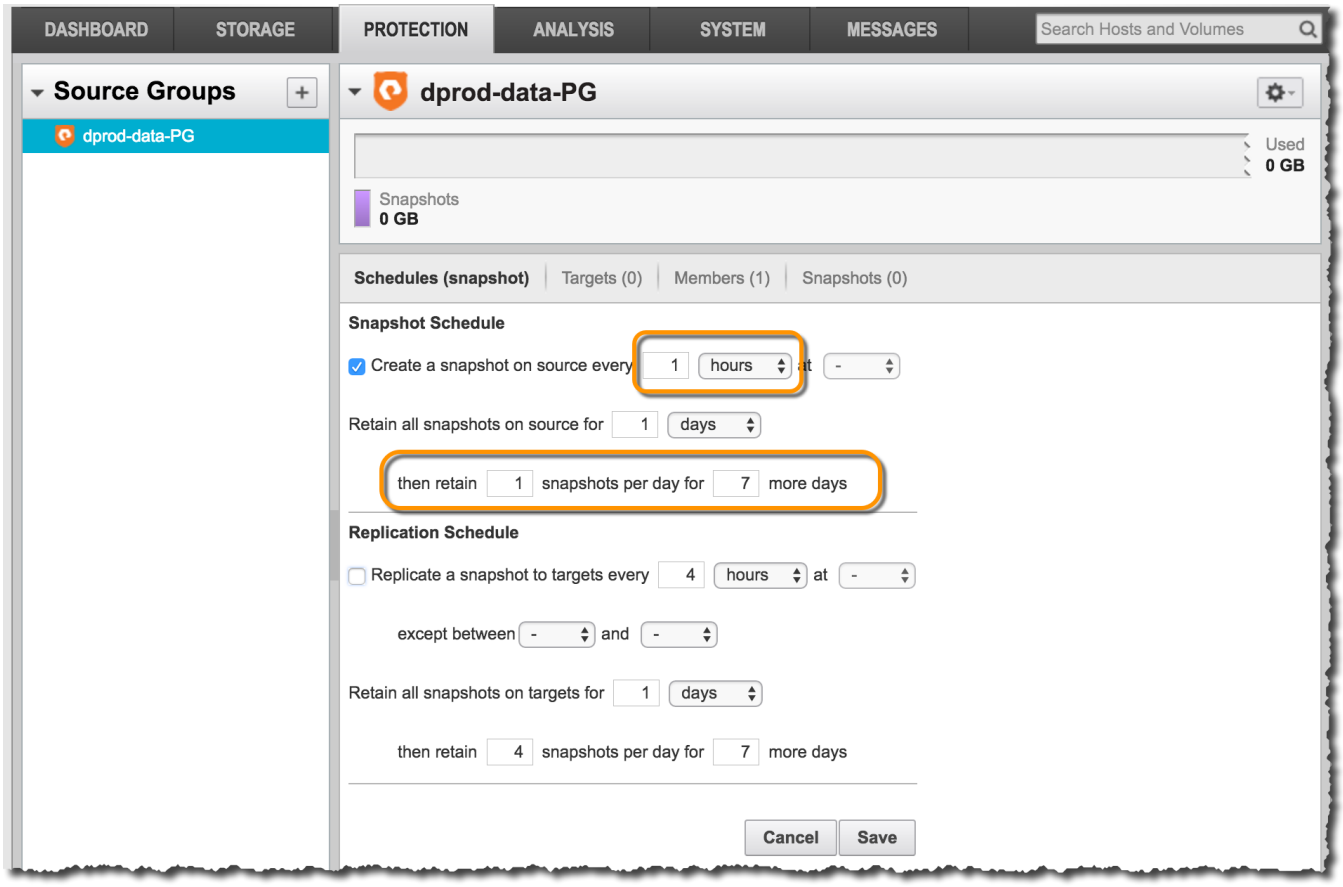

Figure B5: The Default FlashArray Snapshot Schedule

With this schedule it is possible recover the database:

If the desired recovery point is within the part 24 hours

By restoring the most recent snapshot that precedes the recovery point and running RMAN recovery to roll the database forward to the recovery point. The maximum roll-forward is 1 hour, so recovery is relatively fast.

If the desired recovery point is more than 24 hours ago, but within the past 7 days

By restoring the most recent daily snapshot prior to the recovery point and running RMAN recovery to roll the database forward to the recovery point. The maximum roll-forward in this case can be as much as 24 hours.

Once the FlashArray administrator has saved a snapshot schedule, the array takes the first snapshot immediately. Snapshots are effectively instantaneous, capturing the contents of all of a protection group's volumes at the same instant. They are *space-saving*—when first taken, they occupy no data space. As applications overwrite data in a volume, the array preserves the overwritten data and charges it to the most recent snapshot of the volume. This technique makes frequent snapshots possible, thus providing both faster point-in-time recovery and longer recovery windows than are typically feasible with disk-based RMAN backup strategies.

Configuring RMAN

Once a snapshot schedule for the database's **+DATA** protection group that provides the required RPO and RTO has been created, the DBA must configure RMAN to match. With FlashRecover snapshot backup, no RMAN backups are made, but archived redo logs must be retained for a minimum of 7 days so that RMAN recovery can recover the database from a restored FlashRecover snapshot from 7 days ago.

Dialog B9 illustrates the RMAN commands to specify automatic archiving of redo logs with a 7-day retention period. The database management system retains archived redo logs for 7 days, and then automatically deletes them.

```
RMAN> configure archivelog deletion policy to none;

using target database control file instead of recovery catalog
new RMAN configuration parameters:
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE;
new RMAN configuration parameters are successfully stored

RMAN> configure controlfile autobackup on;

new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
new RMAN configuration parameters are successfully stored

RMAN> configure retention policy to recovery window of 7 days;

new RMAN configuration parameters:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
new RMAN configuration parameters are successfully stored

RMAN> configure controlfile autobackup format for device type disk to
2> '/u01/app/oracle/admin/product/12.1.0/dbhome_1/dbs/cf_bkup_dprod.%F';

new RMAN configuration parameters:

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO
'/u01/app/oracle/admin/product/12.1.0/dbhome_1/dbs/cf_bkup_dprod.%F';
new RMAN configuration parameters are successfully stored
```

Dialog B9: Specifying Oracle Log Retention Policy

The final command in Dialog B9 causes the database's control file to be backed up in the specified directory each time there is a structural change in the database (e.g., datafile additions). Control file backups are required to recover databases that undergo structural changes between the time of a restored FlashRecover snapshot and the desired recovery point.

The datafile volumes are protected by FlashRecover snapshots, which can be restored and rolled forward by RMAN using the database's archived redo logs. The 7-day retention period specified in Dialog B9 defines a *recovery window*, a 7-day moving interval for which datafiles can be recovered in the event of failure or corruption.

The remainder of this appendix illustrates populating the example database, simulating loss of a datafile, and combining a restored FlashRecover snapshot with RMAN roll-forward to recover the database.

Populating the Example Database

For this example, a freeware benchmarking tool called SwingBench was run to generate a schema for the database and populate the **+DATA** disk group with about 1.86 gigabytes of data:

Figure B6

The SwingBench schema selection pane in which one of several built-in database schemas is selected.

Figure B7

The pane for specifying the amount of data with which SwingBench is to populate the database. A size of 1.7 gigabytes is chosen for this example.

Figure B8

The acknowledgment pane, in which SwingBench reports that the database was successfully populated.

Dialog B10 on page 29 shows the sequence of SQL queries for view the storage occupied by the populated database, broken down by tablespace.

(The `EXAMPLE`, `SH`, `TEST`, and `TEST2` tablespace is an artifact of the SwingBench software.)

Dialog B11 on the same page shows the redo log history leading up to and during the SwingBench database population run. Logging for the run's transactions begins with log sequence 6, at around 7:01PM, and continues for 1 minute 40.8 seconds (Figure B8**),** finishing while log sequence 19 was active.

During the run, Oracle created 13 redo logs (sequence numbers 6-18, identifiable by the timestamps clustered over about 4-5 seconds), each containing 100 megabytes of redo information (a separately predefined Oracle parameter).

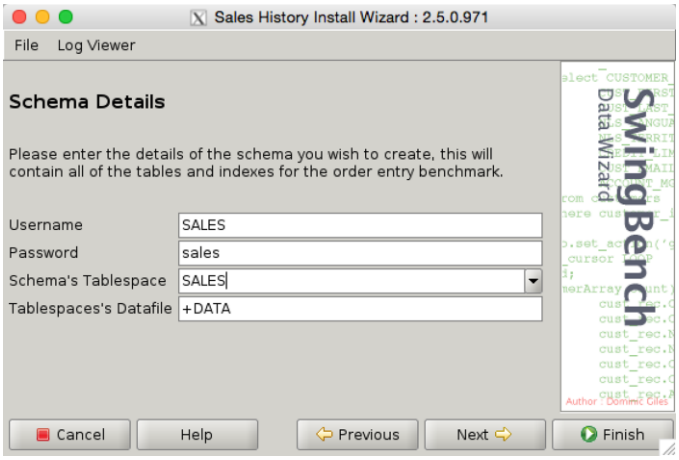

Figure B6: Entering Database Schema Details

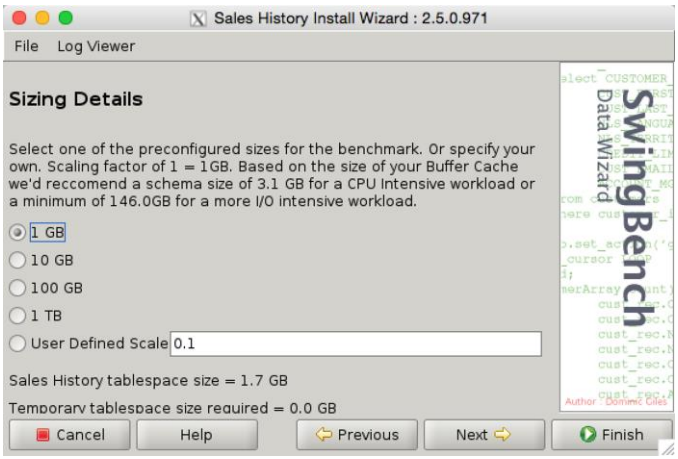

Figure B7: Entering Database Sizing Details

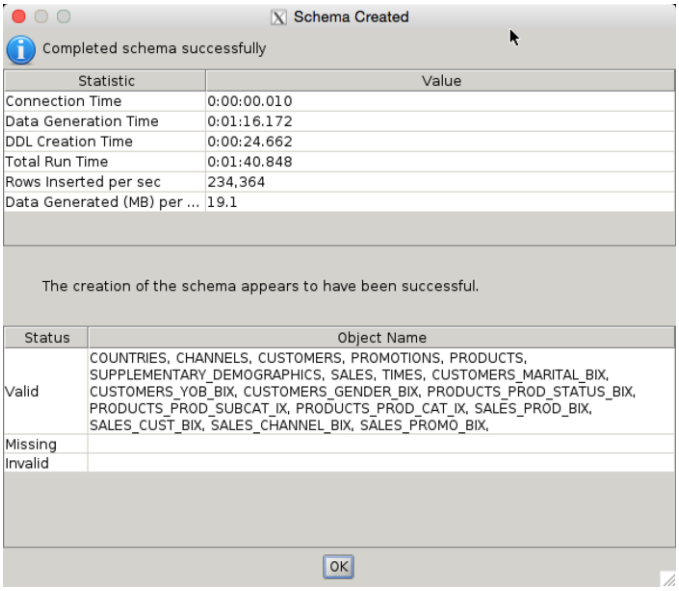

Figure B8: Successfully Populated Database

```
SQL> select sum(bytes)/1024/1024 mb from dba_segments where owner = 'SALES';

       MB
---------
 1,814.00

SQL> select TABLESPACE_NAME, BYTES/1024/1024 MB, FILE_NAME from DBA_DATA_FILES;
TABLESPACE_NAME       MB  FILE_NAME
--------------- --------- ------------------------------------
USERS                5.00 +DATA/DPROD/users01.dbf
UNDOTBS1           100.00 +DATA/DPROD/undotbs01.dbf
SYSTEM             790.00 +DATA/DPROD/system01.dbf
SYSAUX             620.00 +DATA/DPROD/sysaux01.dbf
EXAMPLE          1,243.75 +DATA/DPROD/example01.dbf
SALES            1,869.00 +DATA/DPROD/DATAFILE/sales.263.915735709
```

Dialog B10: SQL Queries to Display Database Storage Consumption by Datafile

```
SQL> select sequence#, first_time from v$log_history;


 SEQUENCE# FIRST_TIME
---------- -----------------
         1 06/28/16 00:21:51
         2 06/28/16 12:47:40
         3 06/28/16 12:48:03
         4 06/28/16 12:48:25
         5 06/28/16 13:22:09
         6 06/28/16 19:01:57
         7 06/28/16 19:02:02
         8 06/28/16 19:02:06
         9 06/28/16 19:02:11
        10 06/28/16 19:02:16
        11 06/28/16 19:02:21
        12 06/28/16 19:02:25
        13 06/28/16 19:02:30
        14 06/28/16 19:02:34
        15 06/28/16 19:02:39
        16 06/28/16 19:02:43
        17 06/28/16 19:02:47
        18 06/28/16 19:02:50
        19 06/28/16 19:02:56

19 rows selected.
```

Dialog B11: Redo Log History

Simulating Data Loss

The next step in the example simulates the loss of the **SALES** tablespace by dropping the datafile that contains it. The commands in Dialog B12 elevate the database access privilege level (**`sqlplus / as sysasm`**), drop the file and then attempt (unsuccessfully) to restart the database without it.

```
[grid@proddb01 ~]$ srvctl stop database -d dprod -o immediate

[grid@proddb01 ~]$ srvctl status database -d dprod
Database is not running.

[grid@proddb01 ~]$ sqlplus / as sysasm

SQL> alter diskgroup DATA drop file '+DATA/DPROD/DATAFILE/sales.263.915735709';

Diskgroup altered.

[grid@proddb01 ~]$ srvctl start database -d dprod
PRCR-1079 : Failed to start resource ora.dprod.db
CRS-5017: The resource action "ora.dprod.db start" encountered the following error:
ORA-01157: cannot identify/lock data file 2 - see DBWR trace file
ORA-01110: data file 2: '+DATA/DPROD/DATAFILE/sales.263.915735709'
. For details refer to "(:CLSN00107:)" in
"/u01/app/grid/diag/crs/proddb01/crs/trace/ohasd_oraagent_grid.trc".

CRS-2674: Start of 'ora.dprod.db' on 'proddb01' failed
```

Dialog B12: Simulating a Datafile Failure by Dropping It from the Database

As the error messages in Dialog B12 indicate, the database fails to start because the dropped datafile, **`+DATA/DPROD/DATAFILE/sales.263.915735709`** is not present.

Recovering the Database by Restoring a Snapshot and Rolling Forward

To restore the missing file:

- The DBA unmounts the (single-volume) ASM **+DATA** Disk Group
- The FlashArray administrator identifies the most recent snapshot taken prior to the SwingBench data load, and restores the volume's contents by overwriting the datafile volume's contents with it.
- The DBA uses the database's redo logs to roll the restored database forward, replaying the SwingBench data load up to the point of the "failure."

The first recovery step is to stop the **+DATA** Disk Group as in Dialog B13.

```
grid@proddb01 ~]$ srvctl stop diskgroup -g DATA -f
```

Dialog B13: Stopping the **+DATA** ASM Disk Group

> *For databases using file systems rather than ASM to organize storage, the DBA would log into a root account and unmount the file system.*

Next, the FlashArray administrator identifies the appropriate snapshot from which to restore an earlier version the missing volume. Dialog B11 indicates that the SwingBench data load started at 19:01. Figure B9 shows the FlashArray GUI view of the datafile volume and its snapshots. As the figure indicates, the most recent snapshot taken prior to the SwingBench data load is **dprod-data-PG.6.dg_dprod_oradata**, taken at 18:55.15.

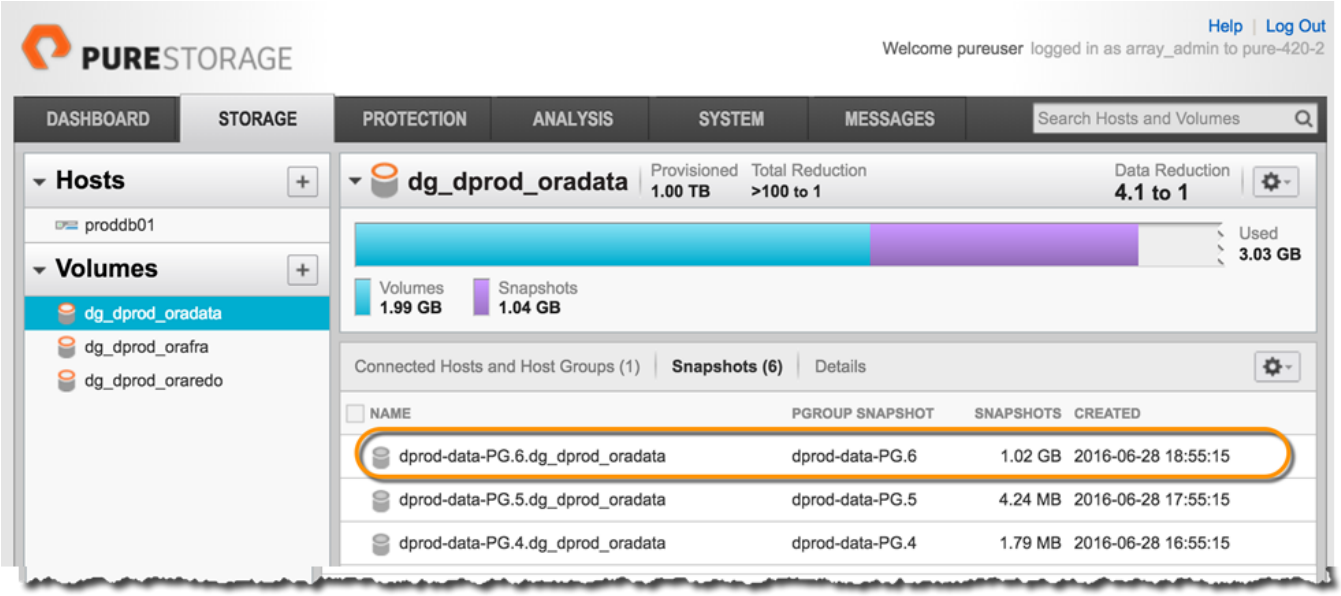

Figure B9: Selecting the Snapshot for Recovering Database

The FlashArray administrator copies the contents of the **dprod-data-PG.6.dg_dprod_oradata** snapshot to the **dg_dprod_oradata** volume, as shown in Dialog B14. Overwriting a volume from a snapshot typically takes less than a second because it essentially consists of replacing the volume's metadata with that of the snapshot.

```
pureuser@purearray> purevol copy --force dprod-data-PG.6.dg_dprod_oradata dg_dprod_oradata
Name              Size  Source            Created                  Serial
dg_dprod_oradata  1T    dg_dprod_oradata  2016-06-28 18:55:15 PDT  2163F6A1D60810ED0001102B
```

Dialog B14: FlashArray Command to Overwrite Volume Contents from a Snapshot

To minimize the possibility of erroneous overwrites, volumes cannot be overwritten from the FlashArray GUI. To overwrite the contents of a volume, the CLI command must include the --force option.

Dialog B15 illustrates the Oracle commands for restarting the **+DATA** ASM Disk Group, followed by restarting the database.

```
[grid@proddb01 ~]$ srvctl start diskgroup -g DATA

[grid@proddb01 ~]$ srvctl start database -d DPROD
PRCR-1079 : Failed to start resource ora.dprod.db
CRS-5017: The resource action "ora.dprod.db start" encountered the following error:
ORA-01113: file 1 needs media recovery
ORA-01110: data file 1: '+DATA/DPROD/system01.dbf'
. For details refer to "(:CLSN00107:)" in
"/u01/app/grid/diag/crs/proddb01/crs/trace/ohasd_oraagent_grid.trc".

CRS-2674: Start of 'ora.dprod.db' on 'proddb01' failed
```

Dialog B15: Failed Attempt to Restart the Database

For databases that use file systems rather than ASM for storage, the DBA would log into a root account and mount the file system on the restored volume.

The database restart in Dialog B15 fails because the control file (in the **+CONTROL_REDO** ASM Disk Group) indicates the database state at the time of stopping (Dialog B12), whereas the datafiles in the restored **+DATA** ASM Disk Group represent a state prior to the SwingBench data load (Figures B6-B8). The disparity is resolved by RMAN roll-forward using the redo logs. Dialog B16 shows the commands for starting an RMAN recovery to the most recent committed transaction, and a snippet from the RMAN console log as it rolls the database forward.

```
oracle@proddb01:DPROD]$ rman target /

Recovery Manager: Release 12.1.0.2.0 - Production on Tue Jun 28 19:26:42 2016
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.
connected to target database (not started)

RMAN> startup mount

Oracle instance started
database mounted

Total System Global Area    6442450944 bytes

Fixed Size                     6870952 bytes
Variable Size               2113931352 bytes
Database Buffers            4311744512 bytes
Redo Buffers                   9904128 bytes

RMAN> recover database;

Starting recover at 28-JUN-16
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 5 is already on disk as file
+FRA/DPROD/ARCHIVELOG/2016_06_28/thread_1_seq_5.260.915735717
```

…additional console output lines not shown…

```
name=+FRA/DPROD/ARCHIVELOG/2016_06_28/thread_1_seq_16.271.915735767 thread=1 sequence=16
media recovery complete, elapsed time: 00:00:08
Finished recover at 28-JUN-16

RMAN> alter database open;

Statement processed
```

Dialog B16: RMAN Roll-Forward Recovery Commands

Restoring a FlashArray volume from a snapshot typically takes a second or less. Roll forward time is a function of the rate of database change, the time between snapshot and recovery point, and host-array I/O bandwidth. The 8-second recovery time highlighted in Dialog B16 suggests that FlashArray administrator-DBA interaction and typing commands is likely to have taken longer than the snapshot restorate and RMAN roll-forward. By comparison, recovering a 1.8 gigabyte database from an RMAN backup would typically entail 30-60 minutes of downtime.

Recovering a Database Whose Structure Has Changed[2]

If a database's structure is altered after a restore snapshot was taken, for example by adding a datafile, the RMAN **`recover database`** command shown in Dialog B16 would fail because the control file indicates the new datafile, but it is not present in the ASM Disk Group restored from the snapshot. Dialog B17 illustrates this scenario. In this case, the options are to restore a later snapshot that reflects the changed database structure, or to change the structure of the restored database,

[2] Refer to Application Brief AB-160501, *Recovering Oracle Databases from FlashRecover Snapshots*, for additional information regarding recovery options available with different versions of the Oracle database management system.

as the example of Dialog B17 illustrates. The Oracle document "Recovering Through an Added Datafile: Scenario" listed in the References on page 5 contains additional information.

```
RMAN> report schema;

Report of database schema for database with db_unique_name DPROD

List of Permanent Datafiles
===========================
File Size(MB) Tablespace           RB segs Datafile Name
---- -------- -------------------- ------- ------------------------
1    790      SYSTEM               ***     +DATA/DPROD/system01.dbf
2    0        SALES                ***     +DATA/DPROD/DATAFILE/sales.263.915735709
3    620      SYSAUX               ***     +DATA/DPROD/sysaux01.dbf
4    100      UNDOTBS1             ***     +DATA/DPROD/undotbs01.dbf
5    1243     EXAMPLE              ***     +DATA/DPROD/example01.dbf
6    5        USERS                ***     +DATA/DPROD/users01.dbf

List of Temporary Files
=======================
File Size(MB) Tablespace           Maxsize(MB) Tempfile Name
---- -------- -------------------- ----------- --------------------
1    60       TEMP                 32767       +DATA/DPROD/temp01.dbf
2    8192     TEMP                 8192        +DATA/DPROD/TEMPFILE/temp.261.915713221

RMAN> recover database;

Starting recover at 28-JUN-16
using channel ORA_DISK_1
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of recover command at 06/28/2016 19:41:26
RMAN-06094: datafile 2 must be restored

RMAN>  alter database create datafile '+DATA/DPROD/DATAFILE/sales.263.915735709' as
'+DATA/DPROD/sales.dbf';

Statement processed

RMAN> recover datafile '+DATA/DPROD/sales.dbf';

Starting recover at 28-JUN-16
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 5 is already on disk as file
+FRA/DPROD/ARCHIVELOG/2016_06_28/thread_1_seq_5.260.915735717
```

…additional console output lines not shown…

```
archived log file
name=+FRA/DPROD/ARCHIVELOG/2016_06_28/thread_1_seq_16.271.915735767 thread=1 sequence=16
media recovery complete, elapsed time: 00:00:17
Finished recover at 28-JUN-16
```

Dialog B17: Recovering a Database Whose Structure Has Changed

Thus, by properly coordinating FlashRecover snapshots and Oracle control file and redo log management, the overhead of backing up a database can nearly be eliminated, even when a database's structure has been altered after the restore snapshot was taken. With the right FlashArray snapshot schedule, coordinated with the Oracle redo log retention policy, recovery times can be as little as tens of seconds, regardless of the recovery point chosen.

Danny Higgins is a Database Architect based in Sinagpore, covering the APJ region. He has been with Pure Storage since 2014.

Prior to developing a taste for all things orange he spent 14 years as an Oracle database specialist in the banking industry. He began his career as a developer, and has worked through various roles including Production Support DBA, Database Engineer, and Database Architect, in which capacity he designed and built private cloud DBaaS platforms.

Danny was introduced to Pure while directing a program to evaluate flash storage for Standard Chartered Bank's 2nd generation DBaaS platform. He was so impressed with the FlashArray that he ~~bought~~ joined the company. Danny has unique insight into the challenges faced by infrastructure teams that manage very large database environments, and uses his experience to advise customers on modern flash-based solutions.

Danny holds a Bachelor of Science degree in Computing from Manchester University in the Northwest of England.

Blog: http://www.purestorage.com/blog/author/dannyhiggins

Twitter: @Danny_Higgins75

PURESTORAGE

Pure Storage, Inc.
Twitter: @purestorage
www.purestorage.com

650 Castro Street, Suite #260
Mountain View, CA 94041

T: 650-290-6088
F: 650-625-9667

Sales: sales@purestorage.com
Support: support@purestorage.com
Media: pr@purestorage.com
General: info@purestorage.com