RMAN in the Trenches: To Go Forward, We Must Backup

By Philip Rice

In the process of implementing Recovery Manager (RMAN) over a period of several years, I had many questions, and some were not easy to answer. This article will share some of my discoveries, so you don’t need to stumble over the same rocks in the path.

Space: The Final Frontier

Before showing how feature choices affect space and time, it will be good to cover some basics. With RMAN, there are two backup format choices, both of which will check for corruption. Both will capture completion metadata, so RMAN knows how to handle recovery. An Image Copy is similar to an OS copy, except that an Oracle server session produces it. The Image Copy is suited to fast recovery, partly because compression is not possible. The Enterprise Manager 10g Suggested Backup Method creates an image copy with incremental changes rolled forward (i.e. it is refreshed daily), enabling a quick return of the database to its state at any point during the preceding 24 hours:

```sql
run {
  allocate channel oem_disk_backup device type disk;
  recover copy of database with tag 'ORA$OEM_LEVEL_0';
  backup incremental level 1 cumulative copies=1 for recover of copy
  with tag 'ORA$OEM_LEVEL_0' database;
}
```

The default Backupset format will be the focus in this article, looking at Incremental rather than Full backups. Incremental Level 0 captures datafile content, used with the RESTORE command. Compare the EM Suggested Backup with a sample Incremental Backupset command:

```sql
backup incremental level = 0 CHECK LOGICAL tag LVL0 database;
```

Incremental Level 1 captures block changes, used with the RECOVER command. The combination of 0 and 1 allows saving a significant amount of backup space. Compression allows even greater space savings. Full is similar to Incremental Level 0, except that Full will not allow capturing incremental changes.

For the moment, look at disk only, and ignore tape. RMAN offers two retention policy choices. REDUNDANCY 1 is the default, and this determines the number of backup copies you want at any time. The RECOVERY WINDOW gives a guaranteed time span for Point In Time recovery (PIT), but REDUNDANCY does not. The example in this article will use seven days:

```sql
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 days;
```

For the example, a Level 0 with a backup size of 50GB will be taken each Saturday night, and there are enough database block changes so that the Level 1 will take 10GB on each of the other six nights of the week. At the end of one week, 110GB of disk backup space is required. Bear in mind that a PIT guarantee of seven days is not a sliding window. On the second Saturday, the prior Level 0 must still be there to allow reaching back seven days. This holds true until after completion of the third Level 0. This example requires a maximum of (50*3) plus (10*12), for a total of 270GB. Now, look at a couple of simple choices that affect the space requirement.

For any database where Level 1 is of moderate size, the best use of space is a window that is one less than the multiple of Level 0 frequency. In the example, that would be a six- or 13- or 20-day Recovery Window. With that simple change, an older Level 0 will become obsolete prior to creation of the next Level 0. Figure 1 shows that the six-day choice minimizes the spikes of space use, reducing our maximum space requirement by 50GB.

```
      50 GB saved for 6 day vs. 7 day retention period

Day, with Level 0 every week

0 10 20 30 40 50 60 70 80 90 100
```

Figure 1

So far, the numbers only include space for one database. Now, consider one server with a half dozen databases, and the script backs up each one serially. The DELETE OBSOLETE command will do two things. It will remove any backup files that are no longer needed, and it will clear the entries out of the metadata, since that information is no longer needed. In the following pseudocode, take a look at DELETE placement:

```
for ORACLE_SID in [DBs to be backed up]
  BACKUP...;
  DELETE NOPROMPT OBSOLETE;
  [End loop]
```

Putting DELETE at the end needs an extra 50GB temporarily each time through loop, because the most recent Level 0 must be completed before the older one becomes obsolete. Be aware that the script works on just one ORACLE_SID at a time, so the second pass in the loop can not see an obsolete Level 0 from a continued on page 20
different ORACLE_SID in the first pass. Putting DELETE in front means that RMAN is not aware of the six obsolete Level 0s until the next day, which is (50*6), for a total of 300GB extra required. Just by changing the placement of the DELETE command, there is a difference of 250GB in this example. Using Flash Recovery Area (FRA) would be somewhat different, because space is cleared automatically when your quota is exceeded. However, the Recovery Window choice and the DELETE command placement ideas should be valid, because your quota allocation for the FRA could be less than it would be otherwise.

Bring Tape into the Picture

It's tempting to just have RMAN do backup to disk, and let your Operations group handle copying to tape. The problem is that the RMAN metadata knows nothing about tape content. RMAN on Oracle Database 10g allows registering a backup set into the metadata, but that is not possible in Oracle9i Database. I've heard of two companies where only three days of backup could be kept on disk, and OS commands were used to copy to tape. To do recovery with a weekly Level 0, multiple days would need to be copied from tape to the original disk location, so files would match the metadata knowledge. With disk space for only three days, recovery would be difficult and time consuming at best.

The Media Management Layer (MML) is an API that lets RMAN talk to your tape software. Even though there is extra complexity and you might have extra licensing cost, the extra effort can be worth it for many situations. The BACKUP BACKUPSET command writes a copy from your backup set, so it avoids the need to read from your source datafiles again. This command can go disk to disk, or disk to tape, but not tape to tape. Keep that in mind if you're considering a Virtual Tape Library (VTL). Even though disk is the physical media for a VTL, RMAN would consider it to be a tape device. The implication is that you need VTL to be a secondary destination, not primary.

Have you Ever Met a Metadata?

Metadata for backup work is found in virtual tables (V$ views) in the controlfile. That repository can optionally be synchronized into real tables (RC views) in a separate catalog database. The catalog can potentially hold metadata for hundreds of databases. The CONFIGURE command is available as of Oracle9i Database that allows turning on autobackup of the controlfile. Even though that made a separate catalog database less crucial, Oracle recommends using a catalog even when a catalog is not required.

When you create a new controlfile, all previous metadata will be lost. How will that affect a RESYNC command? Testing showed that RESYNC is a one-way street, where controlfile information is brought into the catalog. Does that imply that a new controlfile will wipe out entries in the catalog? Fortunately, testing showed that isn't the case.

The catalog automatically captures a full set of metadata, but a good use of that information is not necessarily obvious. This section shows an implementation of trend analysis. For a site using the MML for writing to tape, disk and tape runtimes can show a trend that can be helpful in scheduling. Analysis led to the RC_BACKUP_PIECE view, which is based on just a few catalog tables. Listing 1 is a view that makes the RC view useful:

- To handle backups that occur in the evening, a CASE clause includes anything before 4 p.m. as part of the run from the prior night.
- A TRUNC on the start date allows grouping all runs into a single day.
- Other CASE clauses will push out row values for each day so values are displayed as columns instead.
- We want the “I,” “D,” and “L” backup_type values. Experimenting showed that “I” is Incremental, but just Level 1, not Level 0. “D” is considered Full, which includes Incremental Level 0 and also controlfile backup. It wasn’t worth fine tuning those into two separate display values. “L” is archive logs.
- The FROM clause joins on RC_DATABASE so the database name can be displayed. These two views are joined on a locally created table, which associates database names with the server name. This allows flexibility in reporting, by a single database, or by a single server, or all servers.
- The GROUP BY and ORDER BY clauses have the minimum necessary to make SQL happy; so there is nothing special there.
- A catalog can have different retention periods for different databases, so a persistent table is required for long-term reporting:

```
CREATE TABLE ucs_c_bkup_trend_details AS
SELECT * FROM ucs_c_bkup_trend_insert_vw;
```

On a daily basis, new rows are inserted into this table from the locally created view, and table content is kept for a year. This captures a good amount of detail, which can be rolled up in various ways.

```
CREATE OR REPLACE VIEW ucs_c_bkup_trend_insert_vw [...column aliases...] AS
SELECT
CASE WHEN to_char(p.START_TIME,'HH24') < 16 THEN
  trunc(p.START_TIME - 1) ELSE trunc(p.START_TIME) END AS bkup_date,
  sdl.SERVER_NAME, d.NAME, p.DEVICE_TYPE,
  nvl(max(CASE WHEN BACKUP_TYPE = 'D' THEN p.ELAPSED_SECONDS END),0) AS ARCH_secs,
  nvl(max(CASE WHEN BACKUP_TYPE = 'L' THEN p.ELAPSED_SECONDS END),0) AS LVL1_secs,
  nvl(max(CASE WHEN BACKUP_TYPE = 'I' THEN p.ELAPSED_SECONDS END),0) AS LVL0_secs,
  nvl(max(CASE WHEN BACKUP_TYPE = 'D' THEN p.BYTES END),0) AS ARCH_bytes,
  nvl(max(CASE WHEN BACKUP_TYPE = 'L' THEN p.BYTES END),0) AS LVL1_bytes,
  nvl(max(CASE WHEN BACKUP_TYPE = 'I' THEN p.BYTES END),0) AS LVL0_bytes,
  sdl.SERVER_NAME, d.NAME, p.DEVICE_TYPE,
FROM RC_BACKUP_PIECE p, RC_DATABASE d, UCS ФедерAL_DB_LIST sdl
WHERE d.NAME = sdl.NAME
AND p.DEVICE_TYPE IN ('D','I','L')
GROUP BY
(case when to_char(p.START_TIME,'HH24') < 16 THEN
  trunc(p.START_TIME - 1) ELSE trunc(p.START_TIME) END )
  , d.NAME, sdl.SERVER_NAME, p.DEVICE_TYPE, p.START_TIME, p.COMPLETION_TIME
ORDER BY
(case when to_char(p.START_TIME,'HH24') < 16 THEN
  trunc(p.START_TIME - 1) ELSE trunc(p.START_TIME) END )
  , d.NAME, sdl.SERVER_NAME, p.DEVICE_TYPE, p.START_TIME,
```

Listing 1
This reporting offered a bonus. Even though I got an e-mail every day from the tape run that uses the MML, it didn’t provide very useful information. Graphing from my trend table made me very aware that I needed to turn on *Backup Optimization*. To clarify cause and effect, some background will be helpful.

*Restore Optimization* means that a file is not restored from backup if it is already in the correct location and expected information is found. This feature started with Oracle9i Database, and it is always turned on, so no action is required. This can be especially helpful when recovering from tape after a power failure, because copying large files from tape a second time could eat up a lot of time. Since this only deals with RESTORE behavior, my trend reporting does not have any direct relationship with this type of optimization. This feature is mentioned here because it might be easy to blur the two types together, even though they have different purposes.

*Backup Optimization* must be turned on with the `CONFIGURE` command. This takes effect when you use a channel of only one device type, and it applies to:

- `BACKUP DATABASE`
- `BACKUP ARCHIVELOG` with `ALL` or `LIKE` options
- `BACKUP BACKUPSET ALL`

This feature sounds good, but documentation hasn’t been very clear regarding when it should be used. The good news is that for each of the following situations, RMAN is smart enough to coordinate with your retention policy:

- A read-only or off-line tablespace would be good candidates, because they do not change.
- Another situation is if you decide to keep your original archives in place for several days, like this:

```
BACKUP archivelog all;
DELETE archivelog until time 'sysdate-3';
```

With optimization off, the `rc_backup_redolog` catalog view shows one backup copy for each distinct redo log sequence number (i.e. multiplex is ignored), but will make one backup copy for each day that archive remains on disk. I had this situation, because our site was making a transition on a couple machines to RMAN from the old style hot backups. I was aware of this behavior, so extra archives in my reporting/graphing made sense. But, the trend graphs brought up one more situation that needed *Backup Optimization* turned ON. The big surprise was that optimization needed to be ON for `BACKUP BACKUPSET`, when copying from disk to tape using the MML. Metadata views clearly showed knowledge of incremental level when copying to tape, so it seemed reasonable that `BACKUP BACKUPSET` would be treated the same as the original backup. But incremental backup sets remain on disk for many days until becoming obsolete, and another tape copy is created each day the backup sets are found. In our example with a Recovery Window of seven days, another Level 0 tape copy is created each day for a couple weeks! The extra space required on tape (and time to write it out there) can be greatly reduced when optimization is turned ON.

In Figure 2 through Figure 5, *Backup Optimization* for the first several days is OFF, and it is turned ON for the last few days. Figure 2 shows the time taken to create the backups on disk. Incremental times are fairly constant, but archive times are greatly reduced in the last few days.

![Figure 2](image)

**Figure 2**

Figure 3 shows a ratio of how much goes to tape vs. disk. There is a maximum of 16:1 on the left. After turning *Backup Optimization* ON so RMAN stops making extra Level 0 copies on tape, the graph shows a reduction to a steady 1:1 ratio.

![Figure 3](image)

**Figure 3**

Figure 4 shows tape runtimes, demonstrating in a different way that there are greatly reduced times after *Backup Optimization* is turned ON. Notice how catalog reporting gives a gentle surprise. Clock time was less than execution time, which was not the case for writing to disk. RMAN did not directly cause this, because the API tells the tape vendor *what* to write to tape, but does not tell *how* to make that happen. Examination in the catalog showed that numerous tape processes were used, which reduced the clock time.

continued on page 22
those datafiles, so level 1 would not come into play until doing RECOVER rather than RESTORE. RECOVER...VALIDATE syntax is not available, but it's possible to use the VALIDATE command instead of RESTORE...VALIDATE, to catch remaining pieces. KEY column values are necessary, which can come from LIST BACKUP SUMMARY, or from a query against your V$ or RC metadata views. With that KEY column value in the VALIDATE command, testing shows any or all backups can be examined, including level 1 and archivelog backups. An alternative is RECOVER DATABASE TEST, but documentation for that command says that the TEST clause can be used "only if you have restored a backup taken since the last RESETLOGS operation." That seems to be unduly limiting. Testing resulted in a message saying that the system datafile was in use.

The initialization parameters db_block_checking and db_block_checksum will detect datafile corruptions, as reads and writes are occurring. Similarity in these names can be confusing, but they are not interdependent. When block checking is on, blocks are examined for internal consistency. This block checking is always enabled for the system tablespace, but off by default for other tablespaces. When checksum is on, corruption caused by underlying I/O systems can be detected, e.g. bit flips. A checksum is computed and stored in the block header, before it is written to disk. When db_block_checksum is set to TRUE (default in Oracle9i Database) or TYPICAL (default in Oracle Database 10g), the checksum is verified upon a block read. For strongest possible corruption protection with RMAN backups, a White Paper has recommendations:

- In the initialization parameter file, set DB_BLOCK_CHECKSUM=TRUE. Since TYPICAL/TRUE is the default, that means you can simply not set this explicitly.
- In BACKUP and RESTORE commands, do not specify the MAXCORRUPT option (i.e. no action on your part), do not specify the NOCHECKSUM option (i.e. don’t override your setting), but do specify the CHECK LOGICAL option. It would be grand to see CHECK LOGICAL as a possible clause in a CONFIGURE command, but so far the option must be specifically included in command scripts.

These parameters have some overhead. You can turn on db_block_checking (overriding the default for non-system tablespaces) with LOW, MEDIUM, FULL in Oracle Database 10g, with one to 10 percent overhead. TRUE (backward compatible from Oracle9i Database) is the same as FULL. Documentation says that is the recommended setting “if the performance overhead is acceptable.” How about overhead for db_block_checksum? The default TYPICAL setting incurs only one to two percent additional overhead. The FULL setting in Oracle Database 10g means four to five percent overhead. What do we get for that overhead? When a block is read from the buffer cache in memory, the checksum is verified. If it does not pass, an ORA error results, and the update/delete change is not applied. In that case, it catches in-memory corruptions and stops them from making it to the disk. Otherwise, the change is applied, and the checksum is recomputed and stored into the block header in the buffer cache. At some point, that block with the newly computed checksum gets flushed to disk.

Setting db_block_checksum to FULL will not prevent corruption. But, if blocks are not already corrupted on disk, it could discover the problem early. In my prior employment, our site had a hardware flaw introduce corruption one night during a batch run. This was a processor flaw rather than a disk problem. By the time redo wrapped around far enough to expose this problem, it was well into the next morning. After days of limping along before the hardware vendor found the cause, the financial loss was valued at...

Corruption Detection – No Politicians Involved!

If corruption is detected during a backup, the default is to stop immediately. To override, a SET MAXCORRUPT statement would be required for each datafile. This is intended for very limited use, when your priority is finishing the rest of backup vs. repairing corruption. BACKUP VALIDATE will expose any other corruptions, and repair can be done.

During backup, corruption is detected in datafiles, not the backups. How can you check that your completed backups are good? RESTORE...VALIDATE is an option, but as mentioned in an Oracle Press book, validation is not a comprehensive test.” RESTORE DATABASE looks at headers in the level 0 backup, which is used to get datafiles. Level 1 has changes applied on top of...
$1 million in 1999 dollars. If the corruption in my scenario was introduced in the buffer cache, FULL could have prevented corruption from going to disk, and it would have been discovered during the batch run. With potential cost for not capturing in-memory corruption, there is value in testing to see if the overhead is acceptable. What if blocks in my scenario were already corrupted on disk? Reading (TYPICAL or FULL) or updating (FULL setting) of those on-disk corrupt blocks would reveal the corruption. Scheduling a backup after batch jobs could discover problems incurred during those jobs, since a backup will read blocks and report a problem as soon as it is found.

Flashback

In Oracle9i Database, we only had Flashback Query, which uses Undo. In contrast, Oracle Database 10g Flashback Database is physical, and uses Flashback Logs. Thus, it allows rewinding a database back in time, doing so faster than a PIT recovery. Think about rewinding, and imagine a business losing hours of transactions(!) Here are several scenarios when this feature could be desirable:

- Saving the SCN to a spool file or creating a restore point before running a high-risk batch job would allow backing out to the starting point. This scenario seems to be the weakest. For one thing, risk is at least somewhat reduced if the batch job is well tested. Also, it may not be realistic to isolate so that no other transactions are run during this batch job time span.
- Flashback allows easy conversion of a physical standby database to a reporting database and back to a standby. Activation of a standby database can be reversed.
- In a Test or Development database, we have a known starting point for tests, and rewinding would provide test data in a predictable state.
- A standby can be reverted to an earlier time, which could allow examination and manipulation in two different time periods. This standby would allow recovery of corrupted objects.

To use any of these scenarios, a Flashback Recovery Area (FRA) is required on disk. At a minimum, flashback logs must go in the FRA. Archives can be placed outside of the FRA. However, the Advanced User’s Guide recommends “that you use flash recovery area as an archived log location because the archived logs are automatically managed by the database.” Similarly, controlfiles and online redo logs do not need to be in the FRA. But the Concepts documentation says, “All files necessary to recover the database following a media failure are part of flash recovery area.” Those two file types are included in the “list of recovery-related files in flash recovery area,” along with backup pieces or datfile copies.10 When all file types in that list are placed in the FRA, it means increased potential for read/write contention when there is heavy log switching. The scenarios above can make the FRA worthwhile, but contention should be considered when implementing the Flashback Database feature.

Oracle Database 11g

As of the time of this writing, the Oracle Database 11g release is relatively recent, so I have not dabbled in the pond of new features. However, there are some intriguing points in documentation:

- There is improved integration with Data Guard, including Block Change Tracking support for standby.
- A physical standby or duplicate can be created without a pre-existing backup.

- IMPORT CATALOG allows merging into another catalog. Prior to this, importing into a new schema was possible, but merging multiple catalogs into one was not an option.
- Substitution variables can be used in scripts. This substitution might be helpful for something like tags, and presumably it would have value when stored scripts are used. Unix shell script variables are offering comparable functionality, so this feature is not likely to benefit my situation. It would be grand to see a quantum leap in language functionality. Since RMAN is directly tied into the Oracle engine, imagine the potential if RMAN could use the PL/SQL language that is already present.
- It looks like the Oracle Flashback Transaction Backout feature will offer a nice level of granularity.
- Flashback Data Archive with Flashback Query will allow accessing data from far in the past.

Acknowledgements:

I appreciate the excellent support from Timothy Chien, Oracle RMAN senior product manager, who has clarified many points during preparation of this material. Thanks to Bill Wagman at UC Davis, who acted as a reviewer prior to presentations at NoCOUG in 2007, used as the basis for this article.

References

1. http://download.oracle.com/docs/cd/B19306_01/server.102/b14196/backres003.htm#sthref662
2. http://download.oracle.com/docs/cd/B19306_01/backup.102/b14194/rcmsyn-ta009.htm#sthref141: “By default, RMAN creates all backups as backup sets, on tape or on disk.”
3. This recommendation was one of many points clarified by Timothy Chien, RMAN senior product manager. Here are a couple of situations where a recovery catalog is required rather than simply recommended:
   - As shown in 10.2 CREATE SCRIPT documentation, a global stored script is “available for use with any database registered in the recovery catalog.”
   - As shown in 10.2 ALLOCATE CHANNEL documentation: “You must use a recovery catalog when backing up a standby database.”
5. “Oracle Backup and Recovery” by Smith and Haisley
7. http://download.oracle.com/docs/cd/B19306_01/server.102/b14257/init-params039.htm#REFRN10029

About the Author

Philip Rice has been in the computer field since 1980 and began working with Oracle in 1991. He is now an Oracle DBA at the University of California Santa Cruz, where the school mascot is a banana slug. Philip can be reached at price@ucsc.edu.