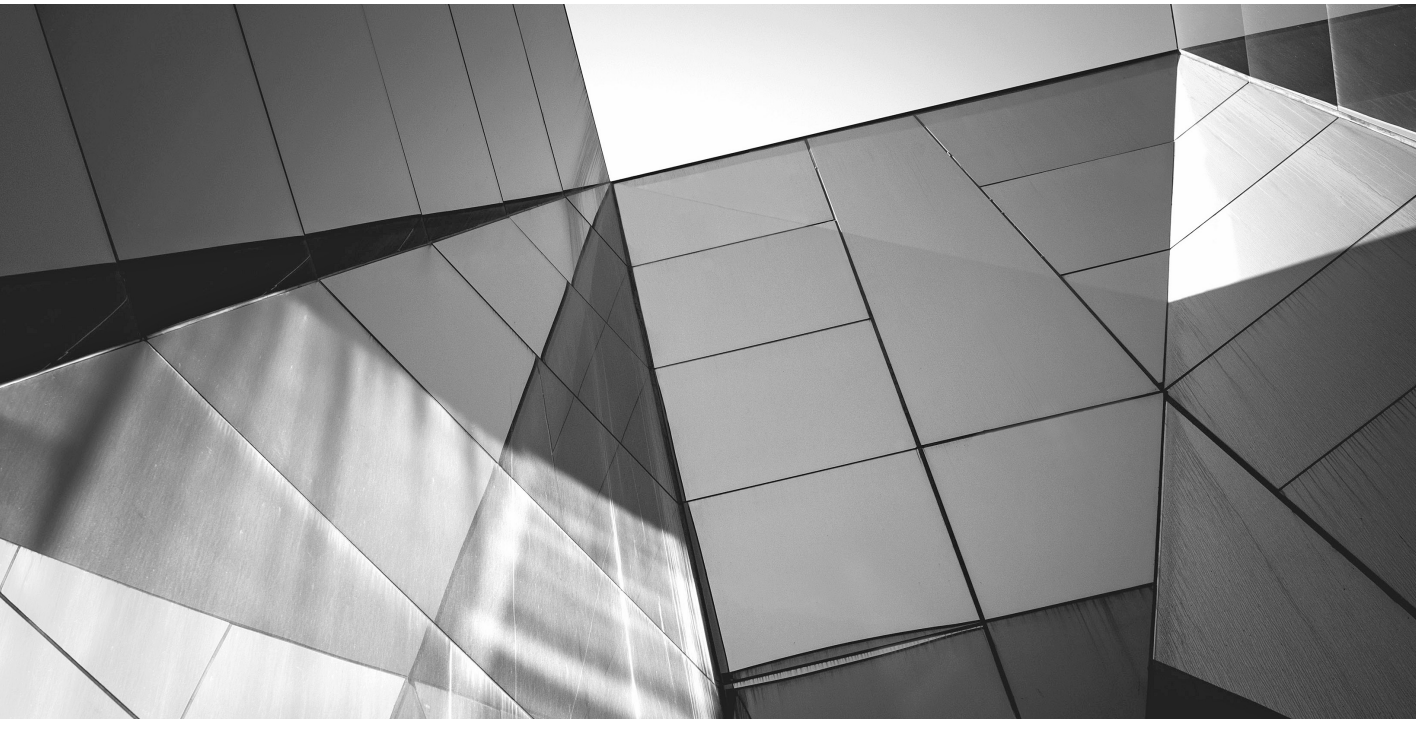

CHAPTER 14

Zero-Downtime Upgrades and Migrations Using Oracle GoldenGate

As the application requirements for database management keep getting more and more complex, the availability requirements also keep increasing, thereby shrinking the maintenance window. It is almost impossible to obtain downtime these days for performing routine maintenance activities on a database. Major upgrades like database versions from 10*g* to 11*g* will need a significant amount of downtime for successful accomplishment of intended upgrades that are subject to stringent user acceptance requirements. A typical database upgrade project will need an enormous amount of planning, testing, and application certification, which can run into weeks or months, depending on the complexity of the application. In addition to database upgrades, migrations or "re-hosts" also call for extensive downtime in order to achieve the end result. For example, an enterprise may choose to replace legacy hardware and consolidate multiple databases on a single Exadata database machine. Or it may choose to move from AIX-based servers to Linux-based commodity hardware. The process of migration of the data from legacy to new environment can take from a few hours to probably days depending on the size of the data. In this modern Internet/online world, can any business afford that kind of downtime, which directly leads to loss of revenue? Can an online retailer or an airline reservation site be down during a database upgrade or re-host activity?

Traditionally, Oracle database upgrades have been "in-place," meaning that the application and the database are brought down and the prescribed procedures to upgrade the database are followed on the production database. This procedure is "invasive." In case a failback is required, there is no option other than extending downtime and restoring the database to its previous state. As discussed in the introduction, this option is more and more likely to be hit with severe resistance from the business owners.

With the advent of Real Application Clusters (RAC) and Oracle Data Guard Technologies, rolling upgrades have been possible, meaning upgrading each node at a time without bringing down the entire cluster. Oracle Maximum Availability Architecture prescribes this, and it is a major step up from the options available before now.

Oracle GoldenGate can reinforce the rolling upgrade methodology in significant ways. By deploying Oracle GoldenGate and taking advantage of its real-time heterogeneous data movement capabilities, complex database environment upgrades and cross-platform database migrations can be achieved with near-zero downtime, making the transition transparent to the business and end users. The only downtime that will be experienced by the users will be during the application switchover.

Zero-Downtime Methodology

This section describes, at a high level, the overall approach to achieving a successful environment migration and/or upgrades with zero downtime using Oracle GoldenGate.

Let's call the database to be upgraded and/or migrated the "source" and the environment where the database is migrated to the "target." The basic architecture behind achieving a zero-downtime migration is nothing but setting up a one-way replication between the source and the target. The steps required to achieve that have been explained in the earlier chapters. Once the target is in sync with the source and ready, the application needs to be switched over to connecting to the target database. This will guarantee that the only downtime that needs to be scheduled is the time it takes to switch over the application to the new database.

If an upgrade is in the plan, the target database can be upgraded to the desired version and then the data resynchronized before cutting over to the target.

If a failback option is desired, which it often is, this concept can be extended and a reverse one-way replication can be set up after the cutover to the target. Now, the target becomes the source and vice versa. This role reversal ensures that changes made in the new database after cutover are applied to the old or the original source database, in case the business decides to fail back to it. The reasons for that can be any; for example, if some new bugs are discovered that were not in the test environment, or if there was a problem in the cutover for some reason. Once the business is satisfied with the new environment, the reverse replication can be discontinued.

Also, if Oracle GoldenGate is set up for bidirectional replication between the old and new environments and both systems support the application for transaction processing, then the application migration to the new environment can be achieved in a phased manner. This helps in eliminating the downtime related to application switchover too, and the transition to the new environment can be completely transparent to the end users.

As Figure 14-1 shows, the steps are self-explanatory, but they are summarized in the following list for clarification. Notice the difference between the database connections of the application server pre- and post-migration.

- Set up a one-way replication between the source and the target. The steps for setting this up are no different than described in earlier chapters. The source server will have the Extract and the Data Pump processes running with access to the source trail file.
- The target server will have the Replicat process running with access to the destination trail file.
- The initial load decisions can be similar to the ones to be made during a plain-vanilla forward replication setup. Export/import or direct load insert, backup/restore may also be possible provided endian compatibility exists.
- The target database can be a brand new database on the desired version or platform. For example, the source is on Oracle 9*i*/SCO and the target is on Exadata/Linux/11*g*R2.

- Oracle GoldenGate Veridata can be employed to verify that the data between the source and target are synchronized. It is optional, but recommended.
- Shut down the application and restart it to point to the new database. If the application is a client/server application, the switch can be affected by switching to a prestaged new tnsnames.ora file. For applications using JDBC, the data source would have to be changed, meaning that a new jar file may have to be deployed. All these steps can be staged ahead of time to minimize the cutover time.

Failback Option

This is the best advantage of employing Oracle GoldenGate: the ability to go back to an old or the original state. But why would anybody want to do that? There can be many reasons. New application bugs may be uncovered in the new environment, or the environment could be encountering unforeseen stability problems. Since the original environment wasn't altered in any way, the application can be just as easily pointed back to the original source database. But, if business wants to have all the

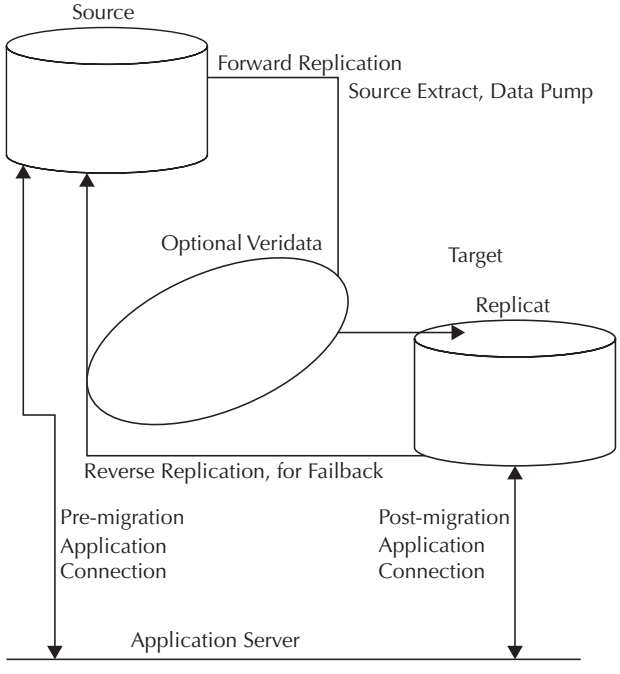

FIGURE 14-1. *GoldenGate architecture supporting zero downtime*

transactions since the cutover back in the original database after failing back, then GoldenGate will come to our rescue.

In the migration project design, it is best to keep in mind the possibility of failback. This high-level process needs to be slightly tweaked to achieve that.

- A new Extract process should be configured and started on the target server, which will act as the new source. The original source will now act as the new target with the Replicat process running on it.
- Essentially, it is a new one-way replication in the reverse direction. This should be done and verified before the application is restarted, during the original cutover process.
- With Oracle GoldenGate Veridata, it can be verified if the data is in sync.
- In the event of stability issues in the new environment, a reverse switchover of the application connection needs to be performed, very similar to the original process.
- Bidirectional data replication can also be used, to make the failback even simpler, but the design of the overall process can get a bit more complicated.

In addition to minimizing the downtime required for such migration/upgrade projects, we can see that we have simplified the whole process by eliminating a large number of steps and associated risks with each of them. Executing of upgrade scripts such as u***.sql—gone. Restore from backup for failback—gone. Downtime—nearly eliminated. Legacy character set conversions—accomplished. Endian conversion—accomplished. Cross-platform migration—easily achieved. Again, all of this has been accomplished with very little downtime.