CHAPTER

Methodology and Design 3

INFORMATION IN THIS CHAPTER:

- Migration Options
- Methodology and Design
- Migration Services

Like any software development project, migration projects require careful planning and good methodology to ensure successful execution. When migrating from one database platform to another a good portion of the database design can be carried forward, especially the schema design for relational database migrations. It may be necessary to make some changes to the existing design and architecture if you want to leverage new features in the Oracle database in favor of older mechanisms, such as using database table partitioning and Oracle Real Application Clusters (RAC) to support multiple workloads instead of maintaining different databases in sync with database replication technologies. Therefore, a certain amount of rationalization is possible when migrating to Oracle from other databases, resulting in consolidation of many databases and schemas from the source database into a few schemas in Oracle. To enable such rationalizations and other optimizations in database design, it is essential to understand a typical migration project life cycle and the importance of each phase in it. Many factors affect the success of a migration project, such as availability of a skilled IT staff, tools, and technologies for migration, an understanding of the source and target database architectures, and realistic project planning. Identifying these factors before embarking on a new migration project can speed up execution and help you to craft efficient solutions for challenges encountered along the way.

MIGRATION OPTIONS

Of all the options for client/server application migrations compared in Table 1.4 in Chapter 1, database migration is the most common because it allows users to migrate to new database platforms, yet leave applications intact, with no changes to existing functionality and business rules, including the languages in which the applications were originally developed. This approach provides the easiest migration path to a new platform, and ensures business continuity in the new environment, along with fewer upheavals. In cases where applications have become very difficult to maintain and update with new features to support business requirements, they are rewritten in new languages that leverage the latest technologies and standards. Such migrations turn into completely new software design and development projects instead of just

Migrating to the Cloud. DOI: 10.1016/B978-1-59749-647-6.00003-X

platform migration efforts. In cases where the business functionality provided by an application is critical for the business and the goal is to make the application accessible from various channels (browser, mobile devices, etc.), the application is typically retooled to run in an environment which emulates a server to allow multiclient, multichannel access.

Legacy applications deployed on IBM mainframes and other proprietary and legacy platforms are usually retooled to run on open and distributed platforms using software that can mimic IBM mainframe environments or that can effectively provide the same capabilities on their own (e.g., Oracle Tuxedo).

Which migration option you choose will depend on your business requirements and constraints (e.g., time, cost, and feasibility). The easiest migration option is to Web service-enable an existing application so that it can interact with other applications on the Web or with applications deployed in a cloud environment. Of course, this approach also requires that these applications be modified to incorporate Web service capabilities by using either third-party solutions or native features, without requiring significant modification. Even if an application written in a language such as Visual Basic or PowerBuilder is not able to access other applications over the network, it can be modified to do so. It is also not necessary for every program comprising an application to be modified. Only programs that provide important reusable business services need to be identified and modified to Web service-enable them. Migration options such as database platform migrations and replatforming applications developed on legacy platforms are very popular because they are easier to execute and realize the benefits of such migrations quickly. On the other hand, migration options involving a complete rewrite/rearchitecture of an application or development of a completely new piece of software are less likely to be chosen due to the cost and time involved in executing such projects.

Lately, new technologies have emerged that aim to provide database transparency to applications. These technologies basically allow you to capture database calls issued by an application and then translate them on the fly to execute them against the target database. The goal of this approach is to significantly reduce application changes as a result of database migrations. However, organizations need to thoroughly test these technologies for performance and for accuracy of the converted SQL statements before deploying them in mission-critical environments. Some database vendors are making big claims that, using these technologies, database migrations can be completed within weeks. However, from the authors' experience, testing alone requires significant effort. These emerging technologies definitely have a role in reducing the overall migration effort in large migration projects by reducing changes to the application due to database migrations.

METHODOLOGY AND DESIGN

Regardless of the migration option chosen for implementation, a sound migration methodology and design needs to be in place before embarking on such a project.

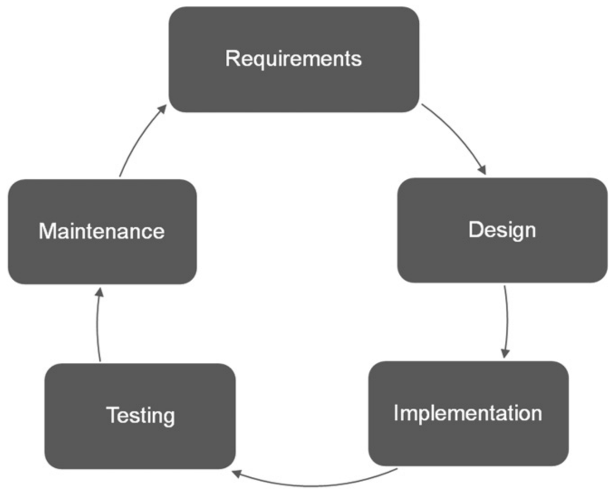

FIGURE 3.1

Waterfall Software Development Methodology

Traditional software development methodologies such as Waterfall consist of the distinct phases of requirements gathering, design, implementation, testing, and maintenance, as illustrated in Figure 3.1.

The Waterfall methodology can be applied to migration projects, but sometimes the requirements gathering phase is replaced with an assessment phase. For instance, for migrations involving a full rewrite or significant transformation to existing applications, the requirements gathering phase is applicable. In projects in which only the database platforms are being considered for migration, there is no need to gather business requirements again, since existing applications already have all the business rules and requirements taken care of. So, instead of requirements gathering, an assessment phase is used to understand the existing environment and determine the best way to migrate it to a new environment. Unlike Waterfall methodology, phases in a migration life cycle can be carried out in parallel (e.g., unit/functional testing of application and database components as and when they are migrated). Similarly, performance testing of some components can be performed before the application/database migration is complete.

Migration of one database platform to another requires some changes to existing applications due to the technical differences between various databases. Figure 3.2 illustrates a typical migration life cycle.

Let's take an in-depth look at each phase in the life cycle of a migration project.

Assessment

The assessment phase is akin to the requirements gathering phase of the Waterfall method for software development. Instead of gathering business requirements

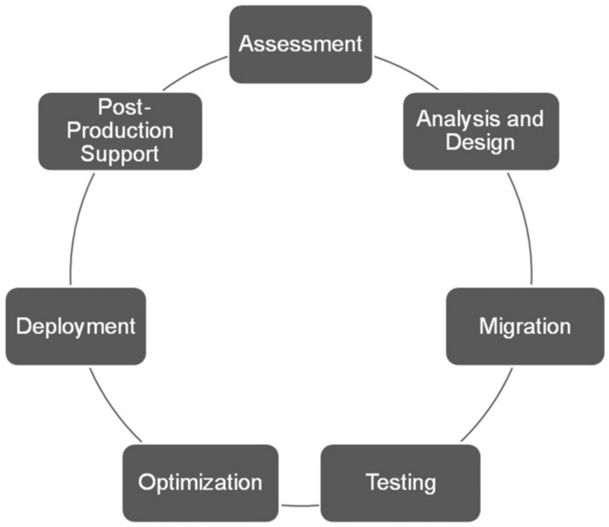

FIGURE 3.2

A Typical Migration Project Life Cycle

from a software development perspective, however, you collect information pertaining to project management (who will manage the project, and how), the potential cost of the migration, migration approaches, tools to use, and so on. In this phase, a detailed inventory of the application portfolio is created to assess the impact of database platform migration on the IT ecosystem, including other applications, integration services, reporting, and backup and recovery processes. For a thorough assessment, the following topics are typically of interest in this phase:

- **Drivers for migration (challenges, requirements)** It is very important to understand the drivers behind a migration effort. For example, if the customer's licenses for legacy technologies are expiring soon, their need to migrate to a newer platform is urgent. The expiry of official support and the extension of support for an existing database platform may be very expensive for such customers, so they must migrate quickly, but they may not need to transform their applications, so they would prefer very few changes to their applications as a result of the migration.
- **Inventory of current environment** Creating a detailed inventory of the current application portfolio really helps in terms of understanding the scope of a migration effort. This includes capturing information regarding the number of programs, scripts, and external interfaces involved. It also includes hardware and software configuration information, including operating system versions, database versions, features/functionalities in use, and similar information.

- **Migration tools/options** It is not uncommon to test different migration tools and technologies to assess their efficiency and accuracy. A high level of automation along with accuracy in migration can result in less time spent in migration and testing. Many customers conduct small-scale proof-of-concept projects to try different migration options, such as emulation or wrapper technologies, which allow applications to communicate with a different database without requiring code changes in the application. This can reduce the overall migration effort in cases where the application is simple and does not have demanding performance requirements.

WARNING

Always choose a migration tool or vendor that has been proven, and does not claim to support migration of any programming language or database on the fly. In almost all cases, when vendors spend a significant amount of time enhancing their tools instead of actually performing migrations, some modification to the tools is essential to address fringe cases of programming language or database feature usage that cannot be automatically converted. Establish verifiable success criteria for these migration tools and/or vendors so that the chances of failure are reduced during migration project execution.

- **Migration service provider** Businesses typically evaluate at least a couple of migration service providers if they do not have migration skills and staff in-house. In many cases, migration service providers utilize their own tools to perform detailed assessment and migration.
- **Migration effort estimate** This is usually provided by the migration service provider or database vendor and is the most common information businesses request when considering a migration project. We discussed how to estimate a migration effort in detail in Chapter 2. As we noted, the estimate depends on factors such as database and application size, components, and database complexity factors, among others.
- **Training requirements** Training requirements for existing IT staff on the new database platform need to be assessed to ensure that they can support the new environment effectively and can participate in the migration process if required. Therefore, it is important to identify appropriate training programs for the IT staff based on their roles in the organization. The most common training programs recommended for database administrators and developers who are new to Oracle database are:
  - Introduction to the Oracle Database
  - SQL, PL/SQL Application Development in Oracle
  - Oracle Database Administration
  - Oracle Database Performance Tuning

Knowledge transfer can also take place from the migration project team to the administration and development teams that will be responsible for maintaining the new system in the future.

- **IT resource requirement for the target database** Requirements for deploying the new database environment also need to be assessed. This assessment should include critical database features and functions as well as additional software that may be required to support the migration process and maintain the migration after it has been deployed. These resources typically include hardware, storage, and Oracle software, including the Oracle database and Oracle SQL Developer, and optionally, Oracle GoldenGate. For virtualization in a cloud environment, Oracle VM software can also be used.
- **IT resource requirement for the migration project** Resources such as the hardware and software required for performing migration tasks also need to be identified. Organizations may need to acquire new hardware and software to support the migration project, or they can provision these resources from a cloud service provider (Infrastructure as a Service [IaaS] and Platform as a Service [PaaS]).

Sufficient time needs to be allocated for this phase to have a complete and meaningful assessment of the migration process. It is not uncommon to see large IT organizations with tens of databases to migrate spending eight to 12 weeks performing a full assessment. When performing in-depth assessments to assist in migration projects, system integrators use an array of tools that capture exhaustive amounts of information from the source systems; this information helps them analyze the dependencies between an application's various components and the database, as well as the complexity of the migration effort. These tools analyze every application program and every line of code in these programs to paint a detailed picture. The following information helps a system integrator assess the impact of a database migration on an organization's applications:

- **Programs interacting directly with the database** This helps the system integrator to identify the number of programs that may require changes to SQL statements or changes to database-specific APIs.
- **Programs or other applications that execute transactions directly** This helps the system integrator to identify programs that may be impacted if there are any changes to transactional behavior in the target database (Oracle), such as:
  - Programs that have explicit transaction control statements in them (e.g., `COMMIT/ROLLBACK`). Typically, these programs maintain control over a transaction they initiate.
  - Programs that invoke a stored procedure to initiate a transaction, but have no control over the transaction. In this case, the stored procedure maintains control over a transaction.
  - Programs that issue explicit Data Manipulation Language (DML) statements (e.g., `INSERT/UPDATE/DELETE`), but do not control the full transaction. In many cases, a master program initiates these programs to execute database transactions and return results.
- **Programs or scripts that offload data from or load data into the source database** These programs will need to eventually be modified to include

changes such as use of Oracle database-specific utilities, associated commands, and parameters, as well as changes to any embedded SQL.

- **The number of management or database administration scripts** Identifying such scripts helps the system integrator estimate the effort involved in migrating these scripts by either rewriting them or discarding them completely to use Oracle-specific tools such as Oracle Enterprise Manager (OEM) for routine administration and monitoring tasks.
- **The type and number of external interfaces** All of these interfaces need to be further analyzed to estimate the migration effort.

It is best to capture as much information as possible in the assessment phase and to analyze it to thwart any technical challenges that may emerge during migration. This also has the benefit of building comprehensive documentation in the long run.

NOTE

The assessment phase tends to consist of an intense exercise during which crucial decisions affecting the migration project are made. Sometimes organizations spend months instead of weeks finalizing their choice of migration strategy, tools, and service provider. Many service providers offer in-depth assessment as a paid service for performing an inventory of the current environment and reporting on the dependencies among various applications, impact analysis, and feature/functionality usage. During the assessment phase, it is quite common for organizations to conduct pilot projects to prove that the choices that have been made will help the organization achieve its goal.

Analysis and Design

The analysis and design phase usually consists of determining the implementation details on the target (Oracle) database. Because of the differences in implementation of data types, security roles and privileges, transaction management, and SQL code, it is important to develop a plan that leverages appropriate features and functionalities in the Oracle database. Care must also be taken to ensure that the chosen features do not result in an increase in application code changes or a decrease in data quality in terms of truncation of data elements such as digits or milliseconds from timestamp data. The following are the most important issues that need to be addressed during this phase:

- **Database schema layout** It is very important to consider how to map the source database schema in Oracle as this can impact the applications and SQL statements embedded within it. Databases differ in how their database schemas, users, and objects are organized. Many databases support multiple databases under one database engine (governing processes). Under each database, objects can be organized in terms of schemas and users. Oracle, on the other hand, supports only one database per instance (or engine) and allows creation of multiple schemas within a database. This difference in database schema layout between Oracle and other databases can result in a collision of objects in the target database schema as other databases allow objects with the same name but with different structures to exist under different databases. As a result, new schemas may need to be created in Oracle and suitable modifications may need to be carried out in the applications to reflect the new database schema layout in Oracle.

- **Database object naming convention** Three major issues usually come up during database migrations to the Oracle database with respect to database object naming convention:
  - **Use of reserved words** Databases differ significantly in what they consider *reserved* words (i.e., words that cannot be used as object names or column names in database tables). This is because databases use these words internally in their software for data processing. During migration, it is possible that some database tables and their column names might run into this restriction on the target database (Oracle). Many migration tools, including Oracle's SQL Developer, provide information on such a possibility. These tools can also convert object names, using a predetermined convention, to an acceptable name in Oracle.
  - **Object name length restrictions** Oracle also imposes an additional restriction of 30 characters in terms of length of database object names. Some databases, such as Microsoft SQL Server, allow object names up to 128 characters in length. So, during migration to an Oracle database, object names that violate this restriction need to be dealt with. The Oracle SQL Developer tool can identify such cases and generate a report that can help developers keep track of areas where this issue will impact the application.
  - **Use of special characters in object names** Use of special characters such as # as the first character for object names in an Oracle database is not allowed; this is not an issue in other databases. As a result, such objects have to be renamed in the Oracle database during conversion, which may result in changes in the applications accessing those objects.
- **Data type mapping** All databases support a variety of data types to handle numeric, character, large object, XML, and timestamp data. Data types available for handling numeric, character, and timestamp data are mostly standard, but they differ significantly in terms of limits on data length and precision allowed. The most common issues encountered during database migration to Oracle are:
  - **Lack of a Boolean data type or BIT data type** The Oracle database does not have support for Boolean data types and BIT data types. So, while migrating to Oracle, these data types have to be converted to either a single-digit numeric or a single-character data type.
  - **Lack of proprietary data types such as `TIMESTAMP` in Sybase** Some databases allow creation of special columns in tables (e.g., `TIMESTAMP`) that the database updates as and when a record in the table is accessed. The Oracle database does not support any such data types similar to what the Sybase database offers in the form of its `TIMESTAMP` data type.

- **Locking behavior** Most relational databases require temporary locking of rows in tables when a user accesses the rows as a result of database query execution (e.g., SELECT statements). Because of this behavior, these databases may suffer from periodic lock escalations when under heavy load. Oracle, however, does not require row locking when simply reading rows. Instead, it supports true row-level locking and provides a read-consistent view of data. Along the same lines, other databases also allow reading of uncommitted data (a.k.a. dirty reads). This feature is typically implemented with the help of an isolation level in the database, which is traditionally used for reading data from tables without locking them. Oracle does not support this feature. Therefore, applications that use this feature need to be evaluated thoroughly to implement an efficient solution in Oracle.
- **Use of COMMIT/ROLLBACK in triggers** Oracle does not allow use of any transaction control statements such as COMMIT/ROLLBACK in triggers. Some databases, such as Sybase, allow partial control (commit/rollback) of SQL statements that are also executed within a trigger. As such, migrating from a database that supports this behavior to the Oracle database requires changes in the trigger code. SQL Developer automatically converts these statements to autonomous transactions where applicable, but this may also have an impact on the overall transaction flow in the application, and therefore appropriate measures must be taken for remediation.
- **Use of zero-length strings (empty strings)** Unlike Oracle, most databases support the notion of empty strings that have zero length but are not considered NULL (i.e., unknown). Oracle, on the other hand, does not support zero-length strings. In Oracle, if a column does not have any value, it is considered to have a NULL value. So, when migrating applications to Oracle from another database, it is necessary to convert any predicates that evaluate a table's column to a zero-length string to a comparison with a NULL. In Oracle, there is a difference between comparing a column to a NULL value and to an empty string and how the database engine resolves such a comparison. For example, the following query will not evaluate to TRUE in Oracle and will not return any rows, even if there is a row in the EMP table where the MGR column has no value (i.e., a zero-length string). Notice the use of the predicate involving a zero-length string.

  ```
  SELECT * FROM EMP WHERE mgr = '';
  ```

  However, the following SQL statement will evaluate to a TRUE condition in Oracle and return rows. Notice the predicate involving an evaluation for the NULL value.

  ```
  SELECT * FROM EMP WHERE MGR IS NULL;
  ```

- **Case insensitivity** This feature in a database allows data to be retrieved regardless of whether the search criteria match the data in the database (e.g., the data in the database may be in uppercase letters and the search criteria may be in lowercase or mixed-case letters). Obviously, this feature is implemented to enhance the user experience. Databases differ in terms of how they facilitate this feature in a database. Microsoft SQL Server allows enabling of case sensitivity by setting the `COLLATE` parameter at the database level or at the column level in a table. When you set this parameter at the database level, all tables support case-insensitive searches on them. In Oracle, this can be achieved in three ways.

1. Set up parameters pertaining to sorting and computational evaluations at the instance level (in the configuration file spfile.ora). The two parameters that control this behavior in Oracle are:
   - `NLS_SORT=BINARY_CI (Default: BINARY)`
   - `NLS_COMP=LINGUISTIC (Default: ANSI)`
2. To enable this feature in Oracle at the session level, the two parameters mentioned in the instance level configuration (i.e., the `NLS_SORT` and `NLS_COMP` parameters) can be set at the session level by issuing the following commands in the Oracle database using a tool such as SQL*Plus or some other database development tool:
   ```
   alter session set NLS_SORT = ‘BINARY_CI’;
   alter session set NLS_COMP = ‘LINGUISTIC’;
   ```

NOTE

To execute these commands the database user needs the `ALTER SESSION` privilege.

3. At the SQL Query level, this feature can be enabled by adding appropriate clauses in the SQL statement. For example:
   ```
   Select * from scott.emp where NLSSORT (''ENAME'','nls_sort='
   'BINARY_CI') =(NLSSORT('Miller','nls_sort=''BINARY_CI'''))
   ```

TIP

To ensure optimal performance for case-insensitive queries involving tables with large volumes of data, it is recommended that indexes also be created on the columns used for these queries, using the `NLS_SORT` clause as illustrated in option 3 (i.e., enablement at the query level). An index can be created as follows:

```
CREATE INDEX ENAME_IDX ON SCOTT.EMP (NLSSORT
      (''ENAME'', 'nls_sort=''BINARY_CI'''));
```

It is also possible to use the `UPPER()` function or the `NLS_UPPER()` function in SQL statements to convert the data fetched from tables as well as input identifiers into `UPPER CASE`. As discussed in option 3, this will require creating a functional index on the columns used for case-insensitive searches and modifying SQL statements to incorporate these functions. There may be other design issues specific to the source database in terms of how a particular feature or functionality has been exploited to facilitate a specific business requirement. Careful consideration and resolution to these design issues is essential to ward off any potential roadblocks during migration and post-migration efforts.

Migration

The amount of time it takes to complete the actual migration of objects and data from one database is relatively less than the amount of time it takes to complete an overall migration from assessment to production rollout. Migrations of one relational database to another are comparatively easier than migrations of a non-relational database to a relational database, because the organization of objects in a relational database is quite similar compared to non-relational databases such as hierarchical and network databases. All major relational database vendors also offer tools that provide robust migration capabilities in an automated fashion. Regardless of the level of automation and success factor of any migration tool, however, sometimes manual intervention will be required when migrating from one database to another. Database migration tasks can be divided into the following categories:

- Database schema migration
- Data migration
- Database stored program migration
- Application migration
- Database administration script migration

Of all the migration tasks listed, the application migration task requires the most manual effort, although new tools and technologies are being developed to facilitate this task. We will cover database schema migration and data migration tasks in more detail in Chapter 5, and we will discuss application migration in detail in Chapter 8. In this chapter, we will focus on best practices for executing these tasks.

Database Schema Migration

Database schema migration essentially involves migration tables, indexes, and views in a database. Relational databases are similar in terms of how their data is organized in tables and indexes, but they are different in terms of additional extensions to these tables and indexes that are designed to improve performance and facilitate development. Most migration tools can convert the database schema relatively quickly and accurately. Target-specific database schemas can also be generated from modelling tools such as Erwin. These are the most important things to consider during database schema migration:

- **Ensuring completeness of the schema** It is necessary to ensure that all objects from the source database have been migrated over to the target database. It is very common to have multiple schemas and databases to support an application. Having circular dependencies among multiple schemas and databases may result in errors during schema creation on the target database, as some of these dependencies may not exist when a particular schema is being migrated. After creating all the schemas in Oracle, all the objects that are marked as invalid need to be recompiled and verified to ensure that they are migrated successfully.

- **Tables with system functions as `DEFAULT` value clauses on columns** Many databases support having system functions as the `DEFAULT` value clauses on table columns. In almost all cases, these system functions do not exist in the Oracle database. As a result, some tables may not be created in Oracle, making other dependent objects invalid. It is recommended that you analyze the log resulting from the schema creation task, and isolate and rectify such errors.
- **Using clustered indexes** Clustered indexes in databases such as Sybase allow data storage in a physically sorted fashion to match the logical order (index). As data is added, it is sorted and stored in the order defined by the clustered index. This helps to reduce the time it takes to return the sorted data and to retrieve data by co-locating the index as well as the actual data in the same object. The Oracle database provides similar functionality with index-organized tables (IOTs). In IOTs, the primary key columns and the non-key data are stored in the same object. This helps users avoid having to look up data in tables separately, after index lookups, while executing a query in Oracle.
- **Creating database users and role assignment** Proper database roles and privileges on objects must be assigned to users. Schema and object-level privileges can be grouped into roles and assigned to users as needed. Creating roles and granting them to users can help in managing many object-level privileges.
- **Changing object names** Any changes to the database object names due to restrictions in the database, as discussed in the "Analysis and Design" section of this chapter, need to be identified and shared with all team members so that they can make suitable changes in their applications or other database components.
- **Partitioning database tables** Oracle allows large tables to be partitioned into smaller segments for management ease and for better performance due to the database query optimizer's ability to prune partitions during query execution, resulting in a reduction in the overall amount of data scanned. Based on data volume, performance, and manageability requirements, some tables may be chosen for partitioning. Although many relational databases support table partitioning, they implement this feature differently in terms of the methods allowed for partitioning, such as range, hash, and composite partitioning. Migration tools generally do not migrate partitioning-related information when migrating database schemas. Therefore, it is important to consider an appropriate partitioning strategy in Oracle after schema migration and before data migration.

Data Migration

After database schema migration, some representative data from the source database is migrated to the target database to enable testing and to ensure that the data migration scripts or tools chosen for the task are configured properly. The most common approach for data migration is undoubtedly the use of scripts that execute database utilities to export data from the source database and import it into the target database (Oracle), because they are easy to use and are free.

Regardless of the tools and scripts used to perform data migration, migrations of very large databases require planning. When migrating very large databases (those with at least a few terabytes of data) it is important to have the right data migration strategy, have the appropriate tools, and, most importantly, use appropriate database features such as partitioning and compression. Migration of large databases is fraught with challenges, among them a narrow window of time and lack of system resources (e.g., staging areas for data files). The following data extraction and loading strategies can optimize the data extraction, transfer, and loading processes:

- Parallel extraction of data from the source database
- Loading of data into the target database in parallel
- Using multithreaded processes for data loading
- Avoidance of index maintenance during the data loading process
- Reduction of I/O operations and use of staging areas via named pipes for data transfer between source and target databases

The data migration task is less time-consuming than migration and testing of database stored programs and application migration. Data migration tasks can be categorized into the following three modes:

- **Offline data migration** As the name implies, data in this mode is migrated in a disconnected or offline mode (i.e., data from the source database is extracted into flat files and then loaded into the target database using native tools and scripts). Because the extraction and the loading processes are disconnected from one another, users can load data whenever they have some downtime in the database, or during off-peak hours. This is typically accomplished using native tools and scripts provided by the database vendors (e.g., Oracle SQL*Loader from Oracle, and LOAD/UNLOAD utilities provided by the IBM DB2 database).
- **Online data migration** Data migration in this mode involves connecting to the source and target databases using Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) drivers, or database gateways, and then migrating between them. As the name suggests, during the data migration, both databases have to be available for connections, and network connectivity between the two is required. Usually, this mode is used for smaller databases with workloads that aren't very heavy. Since data migration in this mode can generate additional load on the source system (CPU, memory, I/O operations), thereby impacting application performance, it is usually not recommended for large databases or heavily used databases.
- **Changed data capture (CDC)** CDC involves tracking changes occurring on the source database and then periodically replicating those changes to the target database. This is a very useful method for migrating very large databases with little or no downtime availability. CDC can be implemented in two ways:
  - **Using log mining** This is the most commonly used technique for implementing CDC. Log mining involves reading the online or archived transaction logs from databases and extracting transactions from them as they are

executed. After the changes are captured from the source, they are either stored in an intermediate file to be transmitted later to the target database, or immediately transferred and applied to the target database. This method is very popular because it introduces less overhead on the source system in terms of performance impact on existing databases and applications, and it is easy to set up and perform.

- **Using triggers** With this method, triggers are implemented in the source database to capture changes and write them to a staging table; the changes are then replicated to the target database. Creating new triggers on source database tables is a cumbersome process that is fraught with challenges. In addition, this method significantly impacts the performance of the source system, and as such, it is not the most popular method for capturing changed data.

Oracle offers a variety of tools that support all the data migration tasks we've discussed. It also offers rich features and products that can optimize the loading, organization, and retrieval of data from the database. Table 3.1 illustrates various Oracle products and technologies that can be used for data migration tasks as appropriate.

Chapter 4 digs deeper into each product mentioned in the table. In large migration projects, a combination of the products mentioned in Table 3.1 can be leveraged.

Database Stored Program Migration

The task of migrating database stored programs includes migration of stored procedures, triggers, and views which, in many relational databases, are used for implementing critical business logic. In databases such as Microsoft SQL Server and Sybase, stored procedures and triggers are used extensively by developers to support simple functions (e.g., the CRUD operations `CREATE`, `READ`, `UPDATE`, and `DELETE`). However, using stored procedures exclusively for CRUD operations can result in

Table 3.1 Oracle Offerings for Various Data Migration Tasks

Data Migration Method	Oracle Offerings
Offline data migration	Oracle SQL*Loader utility, Oracle External Table database feature Oracle SQL Developer and Oracle Data Integrator can generate the scripts for performing offline data extraction and loading.
Online data migration	Oracle SQL Developer, Oracle Data Integrator, Oracle Database gateways
Changed data capture (using log mining)	Oracle GoldenGate and Oracle Data Integrator (for DB2/400, Oracle)
Changed data capture (using triggers)	Oracle Data Integrator (for most databases)

inflexibility because the type of operation executed against a table is limited by the functionality implemented in the stored procedure.

Major tasks associated with stored program migration are:

- **Cleaning and optimizing code** Oracle SQL Developer and other migration tools support migration of stored programs very well. However, it is recommended that you test these converted stored procedures and triggers for accuracy and efficiency of the converted code. Developers can implement a simple business requirement in many ways, making it harder for tools to optimize all such coding techniques in the converted code. Stored procedures and functions with hundreds of lines of code or more should be verified and tested for efficiency in terms of database feature usage as well as optimized coding practices.
- **Handling errors in stored procedures and triggers** For applications that depend heavily on stored procedures and triggers, it is very common to see nested stored procedure calls. Automated migrations may not be able to handle error handling for nested stored procedure invocation. Therefore, it is necessary to pay close attention to error handling, especially for nested stored procedure invocations.
- **Using temporary tables extensively** Some database developers use temporary tables extensively to simplify queries and avoid writing a complex query involving several tables. Early versions of some databases also had restrictions on the number of tables that could be joined in a query efficiently. Therefore, migrating stored procedures with lots of temporary tables warrants a closer look so that they can be avoided and can result in simplified code that leverages the native features of an Oracle database. Typically, migration tools maintain a one-to-one mapping of temporary tables during migration from one database to another. But important stored procedures which are executed very often and have demanding performance requirements should be examined thoroughly to eliminate unnecessary temporary tables in the new environment.
- **Converting stored procedures into functions** The Oracle database does not support returning results to callers using the `RETURN` verb in stored procedures. This verb is only allowed in Oracle stored functions and not in stored procedures. However, it is very common to find Sybase and Microsoft SQL Server stored procedures using the `OUT` parameter as well as the `RETURN` verb to pass values and data to the caller. Converting these stored procedures into functions in Oracle also results in a different call signature (i.e., the syntax for executing a stored procedure versus executing a stored function is different because stored functions in Oracle *must* return a value).
- **Determining the impact of stored procedures returning result sets on Java applications (JDBC)** The Oracle database returns result sets to caller programs via explicitly defined `OUT` variables in stored procedures. However, other databases return multiple result sets implicitly, without having to declare variables to do so. This results in additional changes to Java programs when migrating to

Oracle, such as declaring additional variables, binding, and explicit access of these variables for result set data. We will discuss this issue in more detail in Chapter 7.

Application Migration

Application migration or porting can result from either migrating an application from one environment to another due to a complete rewrite, or simply from an underlying database platform that is being migrated to a new platform such as Oracle. Typically, application development falls into two categories:

- **Customized application development** In this category, applications are generally developed in-house, by IT organizations, to support business functions. These applications almost always try to leverage all the native features of the database platform, as well as other IT systems in the organization, to drive maximum performance and tighter integration. As a result, applications tend to be heavily dependent on the database platform in which they were initially developed. As a result, any change to the database platform may result in changes to the applications. Features and functionalities leveraged by these applications also depend on the developer's skill set. Developers try to use the features they are most comfortable with. Once an application becomes obsolete due to a lack of the skills required to maintain its features, or due to the application becoming too brittle to add new features, the application is migrated to a new environment.
- **Generic application development (or packaged applications)** Typically, this category applies to independent software vendors (ISVs). ISVs develop generic application software that caters to a particular industry or a vertical market. They also tend to develop applications that do not depend heavily on the database. In fact, major ISVs offer versions of applications based on a particular database platform. Migration of a packaged application from one database to another involves installing and configuring the new version of the packaged application and importing the data and all the customizations from the original application. This is by no means a trivial task, because thorough testing needs to be done after the migration. From time to time, ISVs are forced to add support for new databases to their application software due to customer demand. They are also under pressure to maintain a single or as few codebases as possible to reduce the effort involved in managing multiple codebases, each catering to a different database, because this means that if they have to implement a new feature, they will have to modify all the application codebases in a similar fashion and ensure consistency across them.

From a migration perspective, customized applications are always migrated to new database platforms fully, because there is no need for them to support both the old and new database platforms in the long run. These applications can be changed to take full advantage of the new database platform. But ISVs need to support all existing database platforms, even as they add support for new databases. So, for them, it becomes a porting effort because they are simply adding more code to an existing application so that it will also work with the new database. ISVs try to reduce the application software codebase by using conditional coding practices such as conditional branches to a different piece of code, depending on the database platform on which it is deployed. Very large enterprise resource planning (ERP) software packages usually have separate codebases for each database.

As we mentioned when we were discussing the migration assessment phase, understanding the impact of database platform migration on applications is very important. Applications depend on the database platform in many ways:

- **Database-specific connection information** Every database requires certain information to establish a connection with it. In the event of a database change, this information has to be updated in the applications that connect to a specific database. If every single program in an application connects to the database directly, instead of relying on a central database access layer, this otherwise trivial task becomes a challenge. This task can be automated through the use of scripts from the operating system to search and replace appropriate connection strings in application programs.
- **Use of database-specific parameters** ODBC/JDBC drivers for database vendors have different parameters to support different requirements, such as transaction control, date/timestamp formats, and so forth. The Oracle JDBC driver, by default, enables `AUTO COMMIT` on a connection. This might create problems, especially when calling a database stored procedure in Oracle which leverages global temporary tables. Having set the `AUTO COMMIT` by default, the data in temporary tables will be deleted after any data manipulation statement (`INSERT`, `DELETE`, or `UPDATE`). To avoid this scenario, `AUTO COMMIT` for a JDBC connection should be explicitly disabled. For example:

  ```
  Conn.setAutoCommit(false);
  ```

- **Use of database-specific SQL statements** Using database-specific SQL statements with proprietary extensions requires changes when the database platform changes. It is a big challenge to identify how many application programs need to be changed because of their usage of SQL statements that do not conform to American National Standards Institute (ANSI) SQL standards or that are not supported by the Oracle database. In the assessment phase, there is a great deal of emphasis on identifying such programs and their database interactions in general (i.e., calling stored procedures, result set processing, embedded SQL usage, etc.).
- **Invoking database stored procedures and functions that return result sets** Applications using ODBC/OLEDB drivers generally do not need to be modified when the database is migrated to Oracle. However, as of the latest release of Oracle Database 11g R2 (11.2.0.1), Java applications using the Oracle JDBC driver invoking stored procedures returning result sets from the database need to be modified to accommodate Oracle-specific requirements in terms of

including bind variables for result sets, processing of multiple result sets, and similar functionality. Hopefully, these changes will not be necessary in future releases of the Oracle database.

- **APIs for manipulation of large objects** There are differences in JDBC APIs used for manipulating large objects in Oracle as compared to databases such as Informix.

Database Administration Script Migration

It is common to use scripts to automate general database administration tasks. Database administrators love to develop their own scripts to administer databases. However, using scripts to administer databases can complicate things because when script writers leave an organization, new administrators do not have full knowledge of how to use those scripts effectively. Scripts that are used for performance monitoring, and database administration tasks such as user maintenance, object maintenance, and database maintenance, need not be migrated to Oracle due to the availability of OEM, which can manage and monitor databases, application servers, and storage from a central console. Scripts that are used to extract or load data as part of batch processes executed from job schedulers may need to be migrated to make use of Oracle utilities (e.g., SQL*Loader, Data Pump, etc.) to perform similar functions. Oracle provides a rich set of tools to manage these processes—among them Oracle Data Integrator (ODI) and Oracle Warehouse Builder (OWB)—which don't require much coding. However, scripts leveraging native database utilities for data loading/unloading need to be ported to use Oracle database utilities instead.

Testing

Effort involved in testing the application and the database after migration usually is the largest contributor to the migration effort. Testing in a migration project usually comprises tasks such as data verification, testing of migrated business logic in stored procedures, functions, and triggers, testing of application interaction with the new database platforms, and testing of database maintenance scripts. Some of these tasks can be performed easily with the help of automated tools or relatively simple scripting. But some tasks, such as testing of database objects with business logic, can be cumbersome because of lack of automated testing tools. Also, any existing scripts that are currently in use in the source environment need to be ported to the new environment first, and they also need to be tested.

Let's take a look at the various tools and strategies that are used for each of these tasks:

- **Data verification** The easiest way to ensure that the data migrated from a database to Oracle is accurate is to monitor the data migration process closely and ensure that no errors are reported during the process. Even if the migration tools do not report errors, issues such as truncation of decimal values and character data fields may result due to improper sizing of the columns in the target database. Migration of Unicode data also needs attention. Oracle's GoldenGate Veridata can be used for side-by-side comparisons of data between two databases; however, it supports only a few databases (SQL Server, Oracle, Teradata, HP Enscribe, and HP SQL/MP). More information about Oracle GoldenGate Veridata is available at www.oracle.com/us/products/middleware/data-integration/059246.html. System integrators generally have their own toolsets that can assist with the data verification process.

- **Testing of database stored procedures and functions** Usually these objects are unit-tested as they are migrated for syntactic and semantic accuracy. However, after migration of an entire database, it is necessary to test the inter-dependencies among different database objects. Oracle SQL Developer provides features for unit-testing a stored procedure and/or function. However, sometimes it is very difficult to come up with all the possible combinations of parameter values that stored procedures or functions can accept. Therefore, having all the test cases properly documented and scripted can assist significantly in testing efforts.
- **Application testing** In most organizations, testing of custom applications developed in-house is performed by users who work their way through the user interfaces manually based on documented test cases. ISVs, on the other hand, usually have automated test suites available to test full applications. Testing tools such as the Oracle Application Testing Suite (OATS) can be used for functional and load testing of applications for scalability and performance, as it can assist in building test cases from scratch, especially for Web applications. OATS can record all interactions taking place via Web applications and replay them to test how the application interacts with the new database.
- **Database maintenance script testing** It is very important to test any scripts associated with database backup and recovery tasks. Most backup and recovery tasks can be automated using OEM, and scripts and commands used by OEM for performing these tasks can also be reused. Testing of these scripts is a manual process that needs to be carried out in an isolated environment. The Oracle database also provides alternative ways to perform database backups, such as using disk-based backups to automatically perform incremental backups and recovery operations.

If a system integrator is chosen to perform the migration, testing usually becomes a part of the service offering because it requires significant effort.

Optimization

Migrating applications from one database to another database sometimes results in poor performance. This occurs because these applications are highly optimized for a particular database system over long periods of time. OEM can be used to help resolve any performance issues post-migration. It is very common for organizations to set aside at least two to three months for performance testing after migration of mission-critical applications. Some performance issues can also be caught during the functional/integration testing phase. But some issues may crop up only under certain amounts of load on the system. Therefore, it is essential to test the performance of the new platform thoroughly to identify potential bottlenecks in the system and address them before the new platform is rolled out into production. Performance issues can arise due to any of the following reasons:

- **Insufficient system resources** Databases differ in their requirements for system resources such as memory, CPU, and I/O because they have been architected differently. Some databases use multithreaded processes, whereas Oracle is process-based on most platforms except Microsoft Windows. Oracle uses additional database structures such as `UNDO` segments that require additional I/O operations which other databases don't have. Hence, if the system on which the Oracle database is deployed is not sized properly, poor performance may result.
- **Bad SQL Query execution plans** To optimize queries, Oracle's SQL Query Optimizer depends on statistics collected for database tables and indexes during the normal course of database operation. However, if these statistics are stale because of bulk data changes made in a short period of time, or are absent for some reason, the SQL statements will perform poorly. OEM proactively monitors the performance of SQL statements and will alert database administrators of issues. It also runs jobs to collect database object statistics periodically to avoid such issues. However, it is possible that during the migration, some complex SQL statements that were converted to Oracle require additional indexes.
- **Underestimated workload or concurrent user population** Underestimating the peak workload or concurrent user population may result in under-sizing the system used for the Oracle database. This may also result in inadequate allocation of memory for the Oracle database engine in the form of a smaller shared global area (SGA).
- **Undersized Oracle database structures** For optimal performance, it is necessary to size Oracle database structures such as the temporary tablespace and the `UNDO` segment tablespace (a.k.a. rollback segments) appropriately, since the Oracle database automatically allocates space in them as needed. If they are undersized, performance will be poor because of frequent allocations in these tablespaces resulting in increased waits by the database.

Many of these issues are identified proactively by OEM, including recommendations for addressing them. After the database migration and preliminary performance testing are done, Oracle Real Application Testing (RAT) can be used to test the impact of various optimization scenarios (e.g., the use of new indexes, the effect of partitioning, compression, and encryption, etc.) in the new Oracle environment. RAT allows capture and replay of the workload on an Oracle database, and it is much easier to set up and configure than other tools on the market.

Deployment

Many tasks to be executed in the deployment phase get their input from the assessment phase and from the analysis and design phase. During these phases, the target system architecture and all necessary software components to be used in the new system are evaluated. Based on the outcome of these phases, new software and hardware systems are acquired. Work on this phase may begin early in the migration project, as many organizations have to follow certain business practices regarding acquiring new hardware and software.

Because this phase may involve acquisition of new hardware in addition to installing Oracle database software in many cases, additional measures may have to be taken to configure system resources as per Oracle database deployment requirements, such as configuration of shared storage and configuration of interconnected networking among the database server nodes that will be part of an Oracle RAC database. Common tasks executed in the deployment phase include the following:

- **Hardware configuration** This task includes configuring database servers, allocating storage, and configuring the network. Server configuration may also involve tasks pertaining to setting up a cluster to support the Oracle RAC database. In addition, the hardware configuration task involves setting up systems, networks, and storage at the disaster recovery site. Care must be taken when sizing the hardware and storage based on the workload profile of the application and the database. Policies and procedures need to be developed for deploying Oracle databases in a private cloud environment so that instances of Oracle databases can be provisioned as and when they are needed, and to consolidate Oracle databases onto the Oracle Exadata database machine platform.
- **Software installation and configuration** This task primarily consists of installing and configuring the Oracle software, and installing the migrated Oracle database schema on the systems deployed in the production environment to support the applications. After the database schema has been set up, database security roles and privileges need to be assigned to application users, along with access to the physical servers.
- **Initial data loading** After creating the database schemas in the production environment, the next task is to load the most current data from the source database. In cases where the source production database cannot be impacted with data extraction, the most recent backups are restored on a different server and the latest data is extracted from the backup. Then the data is loaded into the new Oracle database using tools and scripts that were chosen for the task during the analysis and design phase. It is also essential to ensure that desired indexes are created on all the tables in the database and that the latest table and index statistics are collected before the database is ready for users.
- **Testing of backup and recovery scripts and processes** It is very important to test all scripts and processes for database backup and restore operations. In many instances, administrators use a standard template to generate new scripts for the

new databases. However, these scripts need to be tested to ensure that they will perform as expected when it really matters; if they don't perform as expected, database recovery can be jeopardized. OEM allows configuration of backup and recovery tasks in an automated fashion which can help in avoiding any errors associated with manually scripting these tasks.

- **Capture of changes (from source) and switchover** For databases that cannot afford downtime and are required to be available 24/7, it is essential that changes that took place during the initial data loading process or during the switchover phase be captured and applied to the new database to avoid missing any transactions affecting the integrity of the data in the new environment. Oracle GoldenGate can play a crucial role in this task of capturing changes in the source database and replicating them in the Oracle database after the initial data loading task is complete. It can continue to capture changes on the source database while the data loading operation is in progress on the Oracle database. The following steps are required in this process:
  1. Set up the changed data capture process on the source database.
  2. Extract the data from the source database.
  3. Load the data into the target database (Oracle).
  4. Apply the changed data captured from the source database to Oracle.
  5. Open the Oracle database for business use.

Post-Production Support

It is a common practice to support a newly deployed database environment with personnel who were involved in the migration process to troubleshoot any issues that may come up immediately after the new environment goes live. The following issues may arise during this time:

- Issues related to unfamiliarity with the new environment.
- Performance issues with SQL statements.
- Applications not performing as expected (missing functionality or incorrect behavior). This may happen if some components of the application were not migrated properly to the new environment and were not tested.
- Problems with data representation in the applications (formatting, Date/Time mask, etc.).
- Time required for database administrators and developers to become fully familiar with the new environment, including procedures for administering routine tasks.

MIGRATION SERVICES

Managing a large migration project requires careful planning, sufficient resources, and certain skill sets. Many system integrators have specialized practices focused on migrations, and have their own methodology, tools, and best practices.

For IT organizations that are planning to migrate many databases as part of a strategic move to consolidate database platforms, it makes sense to engage a system integrator to execute the project, because a system integrator can bring industrialized migration expertise to the table. It is also possible that the system integrator can simply augment his IT staff temporarily to enable him to perform migrations on his own. ISVs, on the other hand, can port their applications on their own because, for them, it is a matter of implementing new database functionality in existing applications as they have to support the other databases as well.

Migrating database platforms for packaged applications bought from ISVs, and installing additional software components of the application, such as the database access layer or, in some cases, new versions of the packaged application certified to run against an Oracle database, may be required. In such cases, it is best if either the ISV or an established system integrator with a dedicated practice focused on these applications is engaged to carry out the database platform migration. The application vendor may have a different codebase for each database that needs to be installed first. As such, migrating from one database to another without changing the application codebase will not work in those cases, and it may jeopardize the ISV's ability to support the application. For example, migrating a PeopleSoft human resources management system (HRMS) application deployed against an IBM DB2 database to a PeopleSoft HRMS application deployed against an Oracle database involves the following steps:

1. Install the PeopleSoft HRMS software and database schema for Oracle.
2. Migrate all the customizations in the PeopleSoft HRMS software from IBM DB2 to the Oracle database.
3. Migrate all the data from IBM DB2 to Oracle.
4. Test the new environment.

Migration of the database schema associated with a packaged application directly to Oracle using migration tools is not supported in most cases by the packaged application vendors. Many system integrators have practices dedicated to migrations and upgrades of a particular application package. Leveraging such system integrators can ensure seamless migration to the new environment.

SUMMARY

Migration projects of any size require careful planning. Assessing its current portfolio of applications can help an organization understand the challenges, complexity, and level of effort required to have its databases migrated to Oracle. Many tools facilitate database and application migrations. These tools differ in the level of automation they provide in migrations and in the accuracy of the migrated SQL statements. Performing proofs of concept with these tools to better understand their capabilities will be beneficial in the long run for large migration projects. Database migrations also have an impact on applications that are dependent on them.

Design-related issues in migrations will vary in complexity and nature within organizations depending upon feature/functionality usage. Some organizations set up policies to develop applications with the goal of being database-agnostic. In such cases, the impact of database changes will be minimal, as they would have avoided implementation of the proprietary features of a database. There is no substitute for comprehensive testing of the migrated/ported application along with the new database platform.

In this chapter, we discussed various migration approaches, tools, and the migration life cycle in depth. We also explored migration service options for different organizations (e.g., ISVs, IT organizations, etc.). The goal was to inform readers about potential choices in migration tools, and various design issues that should be considered. The next chapter will provide an overview of migration tools and technologies along with their strengths and weaknesses.

CHAPTER

Relational Migration Tools 4

INFORMATION IN THIS CHAPTER:

- Initial Database Migration
- Initial Stored Object Migration
- Application SQL Migration
- Unit Testing
- Performance Testing
- System Testing
- Production Rollout
- Global and Niche Service Providers

This chapter covers the products and tools from Oracle and third-party partners that can help to accelerate the relational migration and associated application component of your cloud migration. The chapter is organized by project phase to keep the content consistent with the way we covered the migration process in Chapters 2 and 3. Some of the tools and products discussed in this chapter can be leveraged across the migration phases in different ways. However, we will make sure not to duplicate details of the products and tools where this occurs. You will most likely use the following products and tools in your database migration effort, shown here in order of most-heavily to least-heavily used:

- SQL Developer Migration Workbench
- SQL Developer Application Migration Assistant
- Oracle Enterprise Manager Performance Packs
- Oracle GoldenGate, Oracle Gateways, Oracle Warehouse Builder, and/or Oracle Data Integrator

Most customers use as many Oracle tools and products as possible. There are several reasons for this, but the most pertinent reason is that the tool you will use the most, SQL Developer, is both free and fully supported by Oracle. SQL Developer Migration Workbench is a feature of SQL Developer. Throughout this chapter, we will use the term *SQL Developer Migration Workbench* when the database migration features of SQL Developer are being used. Another reason is that customers prefer to use tools and products from the same vendor to which they are migrating their database. This makes it easier from a training and support perspective, but it also means there is only one vendor to go to if issues are encountered.

We will be covering third-party tools which offer support for source databases that Oracle does not support and, in some cases, offer capabilities not found in