Planning a Database Migration

1. Introduction

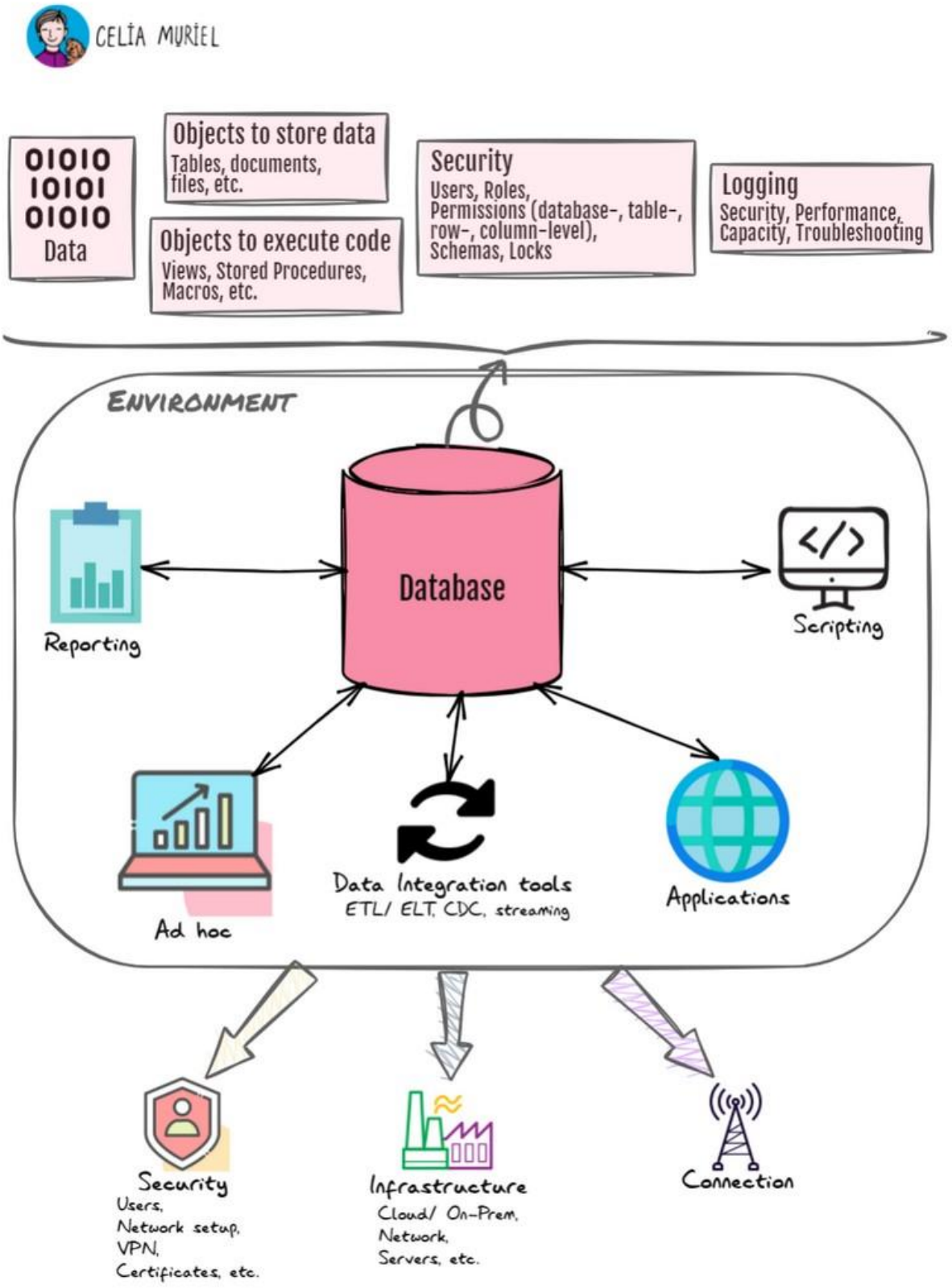
As a Solutions Architect specialised on Data, I have extensively worked on migrations: from on-premise to the Cloud, from one flavour of a vendor's database to other flavours, and from one database to a different one. In my experience, the secret sauce of a successful migration is **planning**.

This post explains the steps to plan for database migration. On following posts, I will discuss specific migration examples, such as migrating from PostgreSQL to MongoDB.

2. Migrating a Database is to Migrate an Ecosystem

As you can see in the diagram below, a database contains data, but it also keeps the objects structure to keep the data (tables, documents, etc.), objects to access or run operations on the data (as views, stored procedures, macros, etc.), the security rules to read or modify the data, and logging mechanisms for security, and performance and capacity calculations.

The database is accessed by reports, different applications, batch and ETL tools, ad hoc queries and scripts. All these elements are enclosed in an ecosystem with specific security and infrastructure, and where the connections among the different aspects (database, reports, ETL. etc.) are defined according to business needs and requirements.



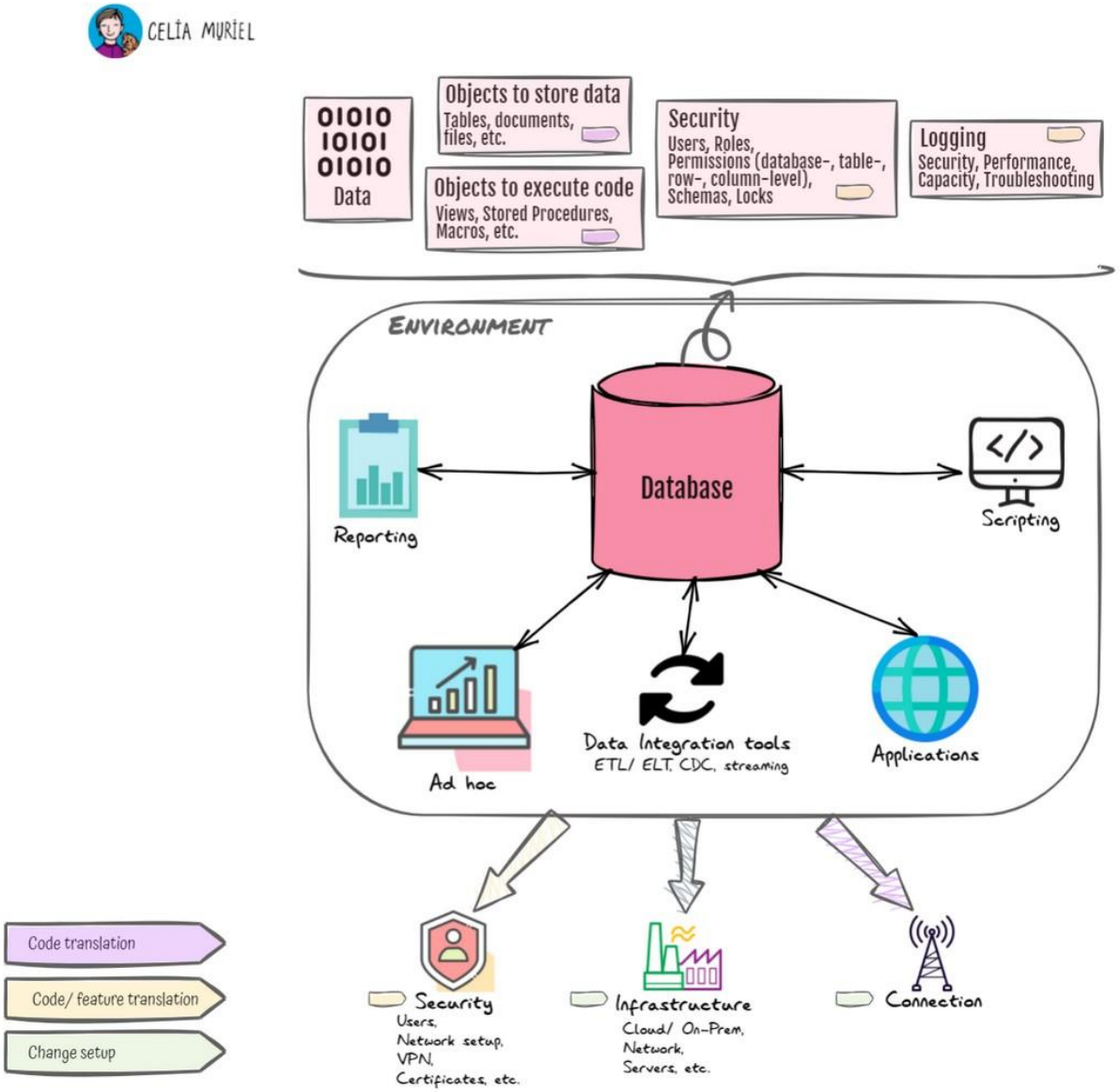
Even though the Connections can be considered part of the Infrastructure and the Security, let me keep them in the diagram above for the shake of a better representation of the steps to follow in a migration.

As shown above, migration is not only about moving data. There are more parts involved.

3. Overview of the Tasks for a Migration

If we consider the relationships of data with other elements in the ecosystem, we can infer that during a migration, we must work not only in moving data but also on tailoring other elements to the new ecosystem.

The diagram below shows where we must translate code, adapt features or change the setup of a tool.



4. Areas to Review to Plan and Design for the Migration

Below you have a list of the areas you need to cover to plan and prepare for a migration.

1. Decide the landing database and ecosystem according to your business requirements and needs.
2. Inventory the objects to migrate and estimate migration times, either manually or with scripts. Some vendors provide with tools to help to do this work, but you will benefit from reviewing the output, and deep dive in the different objects, setups and possibilities in the source.

3. Find the features and limitations of both the source and target databases. See what features you are using in the source that doesn't exist in the target and how you are going to work them around. Examples:
    - Number of concurrent sessions/ queries
    - Representation of timestamps
    - Date / time operations
    - Data types. E.g., INTERVAL in Teradata does not exist in other databases
    - Schemas – They don't exist in all databases
    - Permissions at row or column level
    - When mathematical operations are done, the type of rounding for decimal values.
    - Etcetera.
4. Project Management
    4.1. Identify stakeholders, team, task owners and teams' responsibilities
    4.2. Plan and schedule the migration. Include back down in case of failure
    4.3. Book resources on the appropriate timeframes
    4.4. Assigned specific resources for each task
    4.5. Define the communication channels: meetings, phone calls, critical moments, etc.
    4.6. Communicate to business, stakeholders, teams and task owners
5. Infrastructure
    5.1. Environment for source
    5.2. Target infrastructure
    5.3. Network details
    5.4. Application related infrastructure
    5.5. Compression – It is beneficial to move data in a network, but it consumes resources to compress and decompress
6. Security
    6.1. Follow the company's Security Master Plan and integrate the source and target ecosystems in this plan.
    6.2. VPN
    6.3. Network
    6.4. Data encryption: in transit, at rest
    6.5. Encryption methods and tools
    6.6. Requirements to achieve these tasks. E.g., encryption/ decryption may consume additional CPU
    6.7. Permissions to access the environments
    6.8. Setup schemas, roles, users in the target database
7. Applications (products, tools, ETL/ ELT, reporting, ad hoc, etc.)
    7.1. Which applications connect to the database.
    7.2. Check if the applications connecting to the database are compatible with the target ecosystem and if the vendors are going to support them.
    7.3. Find out if the applications need to be migrated to the target ecosystem, and if the vendor has tools, services and/ or recommendations to migrate. E.g., if you are going to migrate the database Microsoft SharePoint uses as metadata from SQL Server on-prem to Azure SQL Database, they offer a SharePoint Migration Tool, which not only migrates the metadata but

also upgrades the product to the appropriate Azure version. Another example is to migrate Informatica PowerShell from on-prem to the Cloud.

7.4. If the applications are developed in-house ones, check with the appropriate stakeholders if there is knowledge to migrate and who is going to make the necessary changes.

8. Disaster Recovery and High Availability
   8.1. Backup and Restore policies
   8.2. Geo-replication
   8.3. Recovery Time Objective and Recovery Point Objective
   8.4. High Availability needs: active-active, active-passive, none
   8.5. Tools to perform these tasks
   8.6. How to switch to a backup system/ redirect the workload
9. Governance
   9.1. Configure the target environment
   9.2. Collect statistics after the migration
   9.3. Adapt regular governance tasks to the target environment
   9.4. Identify new governance tasks which should be performed target systems as per best practices
10. **Define success criteria**
11. **Set expectations**. You move to another database to get specific features, but you are likely to trade off something. Databases and ecosystems have different characteristics once you migrate.
12. Migration
   12.1. Transform database objects
   12.2. Translate code run on the database
   12.3. Tools to translate code
      12.3.1. Sometimes the database vendors offer tools for free for database migrations from their competitors, or their databases on-prem to the cloud, as Microsoft and AWS. These tools are meant to migrate data and/ or database objects. You still need to work on the other elements of the migration. Also bear in mind that massive DDLs[1] translations may not give the most optimum DDLs.
      12.3.2. Other times there are tools which can be used to automate part of the code transformation (DDLs, data), as Datometry and Qlik Replicate (aka Attunity) for particular cases of competitive migrations.
      12.3.3. If there is not a tool to migrate DDLs and data, you have manually implemented the tools necessary to do these tasks.
      12.3.4. Additionally, there are tools to make the automatic translation on the flight of code from the reporting, tools, ETLs, scripting. E.g., Hyper-V from Datometry. It has pros (simplify the migration), and cons (reduce concurrency, the code to run on the target is not optimum).
      12.3.5. Plan manual reviews to account for best practices in the target database
   12.4. Identify manual conversion objects
   12.5. Define the strategy to move the data
      12.5.1. Plan service outage and move all the data at once (waterfall approach), or

[1] DDL = Data Definition Language

12.5.2. Move the data over a weekend, keep on working, and add incremental data (deltas) little by little until the migration is completed
12.5.3. Estimate time for every option, understand the business requirements to have service
12.6. Tools to migrate the data
12.7. Re-write/ build objects in target Microsoft Database
12.8. Migrate objects
12.8.1. Convert objects
12.8.2. Fix errors
12.9. Migrate data
12.9.1. Identify and fix migration errors
12.10. Convert scripts, jobs, triggers, stored procedures, etc.
12.11. Redirect the connections of all tools and clients from source to target right after the migration

13. Unit testing
13.1. Migration analysis and summary
13.2. Test schema and objects conversion
13.3. Reconcile data
13.4. Validate the converted objects between source and destination
13.5. Test migrate the database to ensure objects and data has successfully migrated
14. Applications
14.1. Test applications
14.2. Resolve any application related issues
14.3. Resolve any target database related issues
14.4. Confirm application is working fine on the new target
14.5. Validate success criteria
15. Connections
15.1. Once the migration is validated and accepted by users, change the connections to point to the target ecosystem.
15.2. Test the applications (ETLs, reporting, ad hoc, scripting, etc.) work correctly, and they run on the target database.
16. Documentation
16.1. Migration architecture
16.2. Database migration
16.3. Migration plan
16.4. Target ecosystem architecture

5. Assessment

To gather all the information to plan and design for migration, you will probably benefit from doing an assessment.

During the assessment, you will need to:

1. Inventory objects, features, applications, etc.
2. Sometimes vendors simplify the assessment with tools they provide for free.
3. Discuss with all appropriate stakeholders

Example of an Agenda to Discuss with Stakeholders to Plan a Database Migration

1. Understand the high-level vision of the company, business areas, and the project scope
2. Review the current infrastructure setup details (Development, Test, Production)
   - 2.1. High Availability, Disaster Recovery
     - 2.1.1. Backup policies, Dual Systems
     - 2.1.2. Geo-replication
     - 2.1.3. Tools
     - 2.1.4. Recovery Time Objective and Recovery Point Objective
     - 2.1.5. Etcetera.
   - 2.2. Capacity
   - 2.3. Bandwidth
3. Details of following as-is current environment:
   - 3.1. Understand end-to-end Solution Architecture
   - 3.2. Understand data use cases
   - 3.3. Understand data availability requirements details
   - 3.4. Challenges faced with existing platforms. E.g., performance, capacity, product features, application functionality, etc.
   - 3.5. Performance details
   - 3.6. Understand load/query performance expectations
   - 3.7. Reporting and consumer layer. Products, Applications, Technologies, 3rd party solution integration and interface details
   - 3.8. Complete Data flow in & out
   - 3.9. Ingestion layer. ETL/ELT Architecture overview including data sources, data movement scripts/packages
   - 3.10. High-level database design overview (data model, logical/physical, table design, data distribution details)
   - 3.11. Understand code artefacts
   - 3.12. Any Service Level Agreement in terms of business needs
   - 3.13. Users and Security details
     - 3.13.1. Industry regulations and company's policies which must be followed to handle the data
     - 3.13.2. Security setup in the source database
     - 3.13.3. Security requirements from the business, and standardisation policies
   - 3.14. Administration and Operations details
     - 3.14.1. Database version
     - 3.14.2. Physical server or virtual machine. Operating system, number of CPU cores, RAM, disk space, IOPS, hypervisor (when it applies)
     - 3.14.3. The maximum number of users which connect to the system at rush hours. Do they expect this number to increase?
     - 3.14.4. Customer Data Space. Expected growth with current applications, and with applications planned to be delivered
     - 3.14.5. Definition of the execution windows in the system. E.g. when the business windows happen, batch window, business window, maintenance window, if they have different

processing during the weekend when they do housekeeping (collect statistics, backups, etc.)

3.15. Customer dependencies details (systems, people, in-house process)

3.16. Current support setup details

4. Success Criteria of the new platform

5. Plans to add further functionality or improve the existing one

6. Applications (products, tools, 3rd ETL/ ELT, reporting, ad hoc, etc.)

6.1. Check with vendors if they support their tools on the target database

6.2. Limitations on the target ecosystem

7. Run tools and scripts to inventory the database.

6. Architect and Design the Migration

Once you have a clear idea of the database, the ecosystem and the business needs and requirements to migrate, as per the guidelines shown above, it is time to architect and plan the target ecosystem and the migration in detail.

This part varies significantly from one project to other, and I'll show in further posts several examples depending on different requirements, technologies, volumes of data and applications.

However, I want to highlight in this section the importance of defining the strategy to migrate the data and the ecosystem, i.e. waterfall approach or migrate deltas.

When you have large volumes of data and/ or you can't interrupt the service to business for more than a few hours, you want to take a delta approach (a delta means moving incremental data). In this case, you make an initial migration, and then, over the next timeframes where applications modify data in the source database, you move the deltas of data which has been modified in the source until you can complete the migration. This approach is slow and it requires you have the means to identify the modified (inserted, updated or deleted) data in large tables, but it allows to keep service over time.

In a waterfall approach, you prevent applications and users to access the database until you complete the migration, and move all data at once. Bear in mind that you will need to move data, objects, applications, tools, scripts, do the initial maintenance tasks on the database (collect statistics, build indexes, etc.) before allowing users to access the target database, and that could be several weeks. On the other hand, it simplifies the way to move data.

7. Define the Migration Plan, Responsibilities and Timeframes

Finally, you must define the migration plan, responsibilities, timeframes and ensure resources are available as per your schedule. Then you will be ready to start your project.

Below you have an example of a High-Level Migration Plan. It focuses on the tasks to migrate data, database object and security. **Add the relevant tasks to migrate your applications or business features**, and itemise in more detail the database tasks, as well as the strategy you have chosen to migrate the data and the ecosystem (waterfall, deltas, etc.). Additionally, decide the execution timeframes as per your assessment.

Task No.	Task	Owner	Notes	Dependency (Task No.)
	Scope of the Project/ Assessment			
1	Discuss Overall Goal, Objective and Scope			
2	Project Scope (what needs to be migrated to target infrastructure for the project, databases, database objects, application, database size, migration strategy, etc.)			
3	Gain a high-level understanding of what the reporting and ETL does and also any dependent ETL, Interfaces, etc.			
4	Identify stakeholders, team, task owners a teams' responsible			
5	Project Infrastructure related Scope: Project Environment for the source (on-prem), Target Cloud Infrastructure, Network bandwidth, Application related infrastructure.			
6	If any security permission is required to give VPN/network access to the on-prem target environment and transfer over VPN/network - contact should secure it			
7	If any security permission is required to migrate data from source (on-prem) to target (Cloud)			
8	Project Team Readiness: provide dedicated resources throughout the duration of the Project (Infrastructure, databases, network connectivity issues, etc.)			
9	Discuss how the project migrated database will be tested using Applications, Reporting, etc. (identify the project application's screens and related database objects, metrics as response times, etc.)			

Task No.	Task	Owner	Notes	Dependency (Task No.)
10	Project Infrastructure readiness: Setup the project target Infrastructure including source database, VPN/network connectivity, target Credentials, setup applications which will be used to test target database			
11	Success Criteria of the Project			
	(i) The application should work the same as it is working currently in new target environment, (ii) Criteria to rollback the migration			
	Architect, Plan and Preparer Paperwork			
12	Use the Assessment details to Architect and Plan the migration			1-11
13	Use the Project scope details and prepare paperwork (contract, approval process, etc.)			1-11
14	Contact details for Paperwork			
15	Do all paperwork for approval			
16	Approval			1-15
17	Book resources as per the agreed timeline to start the project (dates)			1-15
18	Complete setup the Project Infrastructure			1-12
19	Complete setup the project's database and applications			1-12
20	Complete Setup VPN Connectivity			1-12
21	Cloud Subscription - with enough Cloud credit/cost for the project			1-12
22	Internet connectivity - Bandwidth / VPN, etc.			1-12
23	Open database ports			1-12
24	Test connectivity to Infrastructure: databases, VPN, etc.			

Task No.	Task	Owner	Notes	Dependency (Task No.)
	After all the above 23 steps are completed schedule the project kickoff			
	Kickoff and develop Project			
25	Project kickoff conference call			12-24
26	Setup tool for migration			1-25
27	Setup Global Project settings			1-26
28	Create Source and Target Metadata			1-27
29	Convert schema			1-28
30	Identify & fix schema errors			1-29
31	Migrate data (if you choose a delta approach, you will need to add several steps in a loop to add the data in the different deltas as per your plans)			1-30
32	Identify & fix data migration errors			1-31
33	Convert triggers, stored procedures, and the other objects in the database used to execute code			1-32
34	Convert SQL Scripts			1-32
35	Housekeeping tasks to initialise the database (collect statistics, build indexes, etc.).			1-32
36	Application migration, setup/ configuration to work with the new target database – all rest of the ecosystem			1-35
	Unit Testing			
36	Migration analysis and summary			1-36

Task No.	Task	Owner	Notes	Dependency (Task No.)
37	Test schema conversion			1-33
38	Reconcile data			1-35
39	Validate the converted objects between source and target			1-39
	Manual Conversion			
40	Identify manual conversion objects			1-39
41	Re-write/ build objects in target database			1-40
	Testing			
42	Test migrate the database to ensure objects and data has successfully migrated			1-41
43	Test every element in the ecosystem individually, and the integrated ecosystem with all the elements working together			1-42
	Handover migrated target environment for application testing			
43	Configure the connections for the different tools in the source ecosystem to point to the target database			1-42
45	Resolve any application (ETL, reporting, scripting, etc.) related issues			1-43
46	Resolve any target database related issues			1-43
47	Confirm the applications are working fine on the new target			1-46
48	Validate success criteria			1-47
	Document and close the project			

Task No.	Task	Owner	Notes	Dependency (Task No.)
49	Document ecosystem migration summary			1-48
50	Final presentation and closure			1-49