Migration Overview

Release 1.0.0

EDB

Jun 07, 2022

CONTENTS

Organizations migrate their Oracle applications to use other database technologies for a variety of reasons including:

- Reducing costs
- License flexibility
- Open source initiatives
- Initiatives to modernize and consolidate applications and technologies
- The desire to move to new platforms and deployment environments

Due to its rich set of features, quality, and strong development community, Postgres is often chosen as the target database for a migration off of Oracle. EDB provides a distribution of PostgreSQL, EDB Postgres Advanced Server, with built-in compatibility for Oracle and a set of migration tools and services specifically geared toward helping organizations move off of Oracle more easily and with less impact to their business.

The purpose of this document is to help you plan and execute an Oracle to EDB Postgres Advanced Server migration. It identifies and describes the factors to consider when performing a migration, the steps in the migration journey, and the capabilities, tools, and services that EDB provides to help with migrations. Also covered are the key similarities and differences between Oracle and EDB Postgres Advanced Server and workarounds for known differences or incompatibilities.

CHAPTER

ONE

FACTORS TO CONSIDER WHEN MIGRATING

You'll want to think about these factors when analyzing and planning for your migration:

- Schemas
- Data
- Infrastructure
- Applications
- Operations

1.1 Schema migration considerations

The schema defines the structure of the database. Schemas include the definition for tables and other structures, like sequences, indexes, constraints, and views. Although the syntax that Postgres provides for creating these objects is often similar or identical to that of Oracle, there are differences that you might need to address when migrating them. There are also differences in how Oracle and Postgres organize and store schema objects that will be important to understand.

Data types are used in the tables and in the other objects that are used by the database. For a migration, you need to think about how data types in Oracle map to those in Postgres.

A database schema also contains code objects, including packages, procedures, functions, and triggers, that are written directly in the database. In Oracle, code objects are written in Oracle's PL/SQL language. PostgreSQL has its own built-in programming language that isn't compatible with Oracle's. However, EDB Postgres Advanced Server does provide a built-in implementation of PL/SQL, known as SPL. Although EDB's SPL covers a large percentage of the most commonly used Oracle PL/SQL constructs, it doesn't implement the full set of Oracle constructs. As a result, you might need to convert some Oracle code objects to work with EDB Postgres Advanced Server.

There are often syntax differences between how the source and target databases interact with the data. Oracle has certain conventions and keywords that it uses in its version of the Structured Query Language (SQL) for selecting, inserting, updating, and deleting information that aren't supported in Postgres that you'll need to be aware of.

1.2 Data migration considerations

While the schema provides the structure, the data is the content of the database. One thing you need to think about is how to move the data to the target database. The method depends on how much data you need to move and how much downtime you can afford. You'll need to decide whether bulk loading is right for you, meaning you'll get a snapshot of data from the source system, transfer it to the target system, and move on from there. Or, you might need to keep the source application and database up and running while migrating the data to the target database. Since the source database is still receiving updates, you want to be able to continue to move that data to the target database.

Fallback is another consideration. As you go through a migration process, you'll eventually complete the migration. As you finalize testing of that migrated system, you'll often want to have a fallback position in case the migration wasn't entirely successful. After beginning operations in the migrated system, you can fall back to the original source database and source application more easily if the legacy database is being kept in sync with data changes being applied in the migrated database. Your fallback plan must ensure that new data entered into the migrated database can make it back into the original source database in a timely manner.

For migrating data, in addition to the core tools we offer for pushing data from the source database to the target database, you might also use tools to support more advanced data migration scenarios, such as more complex extract-transform-load (ETL) needs. These tools pull the data out of the source database, transform it, and then load it into the target database.

Once the data is in the target system, you want to validate that the data that you put in the target system matches the data in the source system.

1.3 Infrastructure considerations

Databases reside on infrastructure, so you'll have to consider the hosting environment for the source database as well as consider what the target environment will be. For instance, if your current database is deployed on premises, you might decide to migrate to another database also running on premises. However, more and more organizations are considering moving from an on-premises deployment of a database to a cloud-based deployment. If a cloud-based deployment is the ultimate goal, you might decide to migrate the database directly from one running on premises to one running in the cloud. Or you might first migrate the database to one running on premises and then move the migrated database to the cloud.

If migrating to an on-premises database, either as the end state or as a stepping stone to the cloud, you need to make sure you have the server resources on hand to perform and test the migration and to host the operational database and related tools. If migrating to the cloud, your needs are similar, but you will likely have more flexibility to add and remove servers as needed to support the migration process.

Another factor to consider is that many legacy application databases were deployed on specialized hardware and operating systems. This factor might influence the types of tools and approaches used for a migration given that you'll likely be targeting server options. You'll need to make sure the tools you plan to use can either be deployed on these systems or can access them to retrieve the information to migrate.

When considering a migration to on-premises resource, the deployment options available for EDB Postgres Advanced Server are:

- Physical servers
- Virtual machines
- Cloud-native Postgres containers in Kubernetes
- Private cloud instances

Traditionally, databases were installed on physical servers to optimize performance. Over time, as virtualization technology improved, it became more common for databases to be installed on virtual machines. In the last few years,

containers have begun to emerge as a viable option for database deployments, especially as the industry has more or less standardized on Kubernetes as an orchestration framework and database vendors like EDB have developed Kubernetes operators for deploying and maintaining databases in containers.

For migrations to the cloud, the following are the deployment options for EDB Postgres Advanced Server:

- BigAnimal cluster
- Self-managed EDB Postgres Advanced Server installed on cloud-based server instances
- Cloud Native PostgreSQL (CNP)

More and more, organizations are interested in moving to a managed database service (also known as DBaaS) to take advantage of the benefits this service provides to businesses. BigAnimal is the EDB Postgres DBaaS offering available on the major commercial clouds that provides the option to easily deploy either PostgreSQL or EDB Postgres Advanced Server database clusters. If you want to migrate to the cloud but want or need more control over managing your database instances and the underlying servers, you can install and manage EDB Postgres Advanced Server on your own in cloud-based server instances. EDB's Cloud Native PostgreSQL (CNP) lets you deploy and manage PostgreSQL or EDB Postgres Advanced Server clusters in Kubernetes-orchestrated containers. Widespread adoption of containers for databases is still in its early stages. However, as applications shift toward microservices, the use of containers for databases is increasing. Kubernetes clusters can be stood up in the cloud. In fact, the major cloud providers even offer managed Kubernetes services.

When you have different source infrastructure and target infrastructure, consider that you might have optimized your source system based on the infrastructure. When you move to the target system, you might have to do similar optimizations but using different techniques.

1.4 Application migration considerations

An application sends information to the database and retrieves information from the database. Most applications that use a relational database management system (RDBMS) backend, such as Oracle or Postgres, issue database queries in the Structured Query Language (SQL). The essential components of the SQL standard were adopted by all major relational databases. However, each of them have features that deviate from the standard and are different from one another. As such, some conversion of the SQL code embedded in the application code, either statically or dynamically generated, is likely required as part of the migration efforts.

In addition, applications usually are tuned and configured to perform well to work with the legacy database system and operating environment. When you move from the old system to the new system, you might have to make some changes to your application to support running well against the new database and operating environment.

1.5 Operational considerations

Migrating a legacy database to new database technology results in the need to update operational processes and tools to work with the new database. In the case of migrations of Oracle to Postgres, any Oracle-specific tools that are being used will need to be replaced with Postgres-specific tools. Some of the major operational aspects to consider include high availability, backup and recovery, monitoring and management, security, and encryption.

Also, when legacy databases are running on older specialized hardware and operating systems, a database migration might also require migrating the underlying host systems to modern platforms and operating systems. For on-premises migrations, this can result in the need to change your operations and maintenance procedures and tools to work with the new server platforms and operating systems. For migrations to the cloud, some of the operations and maintenance activities will be handled by the cloud service provider but others will still be your responsibility. Likely, those activities will require different tools and procedures from the ones you're currently using.

1.6 Some best practice considerations

The following are some best practices to consider when planning for your Oracle-to-Postgres database application migration:

- Don't skip or skim over steps in the migration journey. - Do the planning. - Identify and understand the functional and nonfunctional requirements. - Complete the solution design early to understand IT resource needs.
- Use development and test environments to help identify and resolve migration issues prior to migrating to a production environment.
- Get application developers and system administrators involved early in the migration journey to work with the database team.
- Set up an environment where the application can be tested with the database in a manner that closely approximates the production environment and usage.
- When migrating the database schema, defer loading indexes and constraints until after migrating the data.
- Engage the help of migration experts and PostgreSQL experts for more complex migrations.
- Ensure operations personnel are trained on the new system prior to post-migration operations.

CHAPTER

TWO

MIGRATION TECHNIQUES

At a high level, a database migration from Oracle to Postgres includes:

- Transforming the schema from Oracle's extended version of the SQL standard to a Postgres-compliant version, which is generally more compliant with standards
- Translating Oracle-specific data type definitions to Postgres-specific ones
- Rewriting queries and stored procedures
- Copying data
- Updating application APIs to use Postgres JDBC, .NET, ODBC drivers, as Oracle has extended the standard protocols with proprietary extensions
- Verifying that the migrated database meets all the nonfunctional requirements related to performance, manageability, high availability, and integration with enterprise security requirements.

You can perform the migration manually or in an automated fashion.

2.1 Manual approach for transforming databases

Executing these transformations manually is expensive and error prone. Except for very small databases without any business logic and extremely simple application logic, this approach isn't practical.

2.2 Automated approaches to database migration

Two automated approaches are usually used instead of a pure manual approach:

The *native compatibility* approach extends one database's capabilities and creates a native implementation of another database's specific extensions of the SQL standard, including the APIs and protocols. EDB Postgres Advanced Server is an enhanced version of PostgreSQL that offers compatibility with Oracle in the following areas:

- Oracle-specific and syntax-compatible database object types
- Oracle-specific data types
- Oracle-specific SQL extensions
- Oracle PL/SQL support as a built-in native procedural language
- Oracle data dictionary views (i.e., **ALL_**, **DBA_**, USER_ views)
- Oracle built in PL/SQL packages
- Oracle-like database drivers

- Oracle work-alike tools for DBAs EDB Postgres Advanced Server doesn't fully implement all Oracle-specific features in these areas. Nevertheless, the compatibility is extensive and covers many of the most commonly used Oracle constructs. As a result, the compatibility features that were implemented makes most migrations off of Oracle easier. The *translation* approach uses automated tools to rewrite (or translate) the Oracle definitions, queries, and stored procedures to Postgres-compatible versions. The EDB Migration Portal repair handlers use this approach to automatically convert a large number of objects that contain constructs that don't have a compatible implementation in EDB Postgres Advanced Server. The translation approach is the only approach available with open-source migration tools and those provided by other vendors to target pure open-source PostgreSQL. However, depending on the number of objects that need to be translated, the time and effort required to get to a pure open-source solution might not be supported by your schedule or resources.

While most migrations are greatly simplified through the combination of these two automation approaches, some objects usually still remain that require manual conversion. Refer to the workarounds in the *Migration Portal* Knowledge Base or contact EDB's Professional Services to help identify solutions for any manual conversions that remain.

CHAPTER

THREE

THE DATABASE MIGRATION "JOURNEY"

Migrating your database consists of a nine-step "journey."

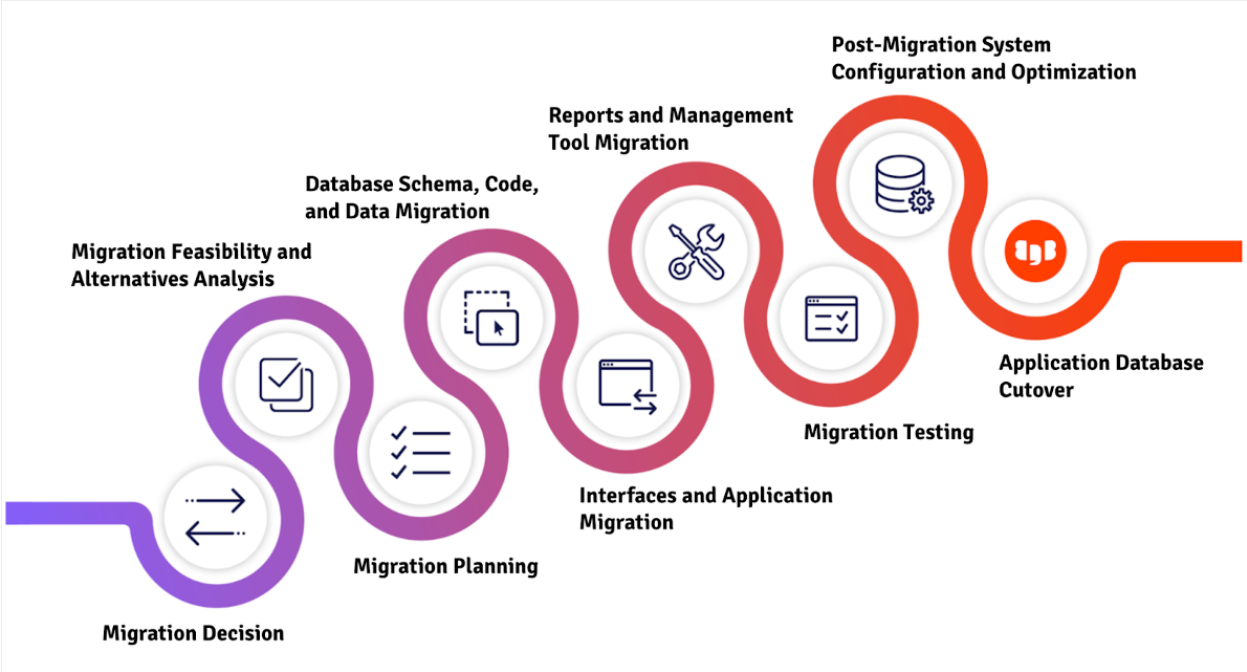

Fig. 1: Migration journey

3.1 1. Decide to migrate

First, your organization must make the decision to migrate. You might decide you're spending too much money on your legacy database system. You might have initiatives that your current database technology doesn't support well. Or maybe you want to move to the cloud or take advantage of open-source software. These are common factors that might lead to your business case for migrating.

Once you have your business case and your organization buys into it, your organization must commit to performing the migration. Commitment includes funding and resources to perform the migration.

3.2 2. Analyze feasibility and alternatives

Analyze your alternatives and look at the feasibility of migrating to certain databases.

3.2.1 Review your application portfolio to determine whether all of your applications will migrate

As part of this review, you'll look at the applications that have legacy databases that you can and want to migrate from. Align your migration priorities with IT strategies to ensure these are important enough business applications to consider for migration.

As part of reviewing your application portfolio, you might decide that you can retire some applications. There might be other applications in your portfolio that you want to migrate but can't because they're supported only on the existing database.

As you're identifying a candidate set of applications to migrate, keep in mind that the desire to migrate them must correspond to an organizational commitment to completing the migration. It will take time, resources, and budget.

3.2.2 Evaluate which database technologies are the right choice as the migration target for each of your applications

In many cases, EDB Postgres Advanced Server is an ideal choice to replace Oracle. However, some applications have requirements that might be better addressed by other solutions.

After selecting the appropriate set of target database technologies, a follow-on step is to perform a more detailed assessment of how complex the migration will likely be. When EDB Postgres Advanced Server is a potential target, you can use the EDB Migration Portal to assess how compatible the Oracle schema and procedural objects in the database are with EDB Postgres Advanced Server. This information is important in the next step of the journey.

3.2.3 Prioritize applications

If you're considering migrating a large estate of applications, perform a first-level pass to identify the ones you want to move first. Are some so important that there would be more risk in moving them? You'll also want to look at the difficulty of the migrations and prioritize them based on that.

3.2.4 Decide on the hosting environment to target for your migration

See *Infrastructure considerations* for more information about the options you have for hosting the EDB Postgres Advanced Server database. Are you currently running on premises? Do you want to stay on premises or move to cloud? Or are you on the cloud and perhaps that hasn't worked out and you need to move back to on premises or to another cloud provider?

3.2.5 Identify potential migration problems

Although you'll perform a more thorough review of solutions to address specific compatibility issues that are encountered later in the journey, it's also important to identify potential migration problems in this early stage. For an Oracle-to-EDB Postgres Advanced Server migration, see 'Comparison of EDB Postgres Advanced Server with Oracle' for details.

The main goals of this step are to:

- Understand which applications are good candidates for migration.
- Analyze the different technology and vendor options for the migrations.
- Understand the big picture of the overall level of effort, time, and cost to complete all the migrations.

Also consider the cost of continuing on current technology and platforms versus migrating to new technologies and platforms.

3.3 3. Plan the migration

Planning your migration involves these processes:

- Prioritizing the applications to migrate. Once you've decided on the applications to migrate, prioritize the order in which you'll migrate them. Organizations often consider selecting the most difficult ones because they believe that if they can do the most difficult migration, then all the other ones will be easier. However, we usually recommend picking one that is moderately important to you—not necessarily the most valuable to customers—and also not the most difficult. This way you can learn from the process. Then select the next ones based on what you learned from that first migration and so on until you eventually get to the most valuable and most difficult.
- Identifying nonfunctional requirements, such as performance, availability, management, and integration.
- Designing the solution at a high level, including both the migration architecture and the post migration architecture. For example, select the database, the platform, the type of migration, your high-availability requirements, and so on.
- Estimating the migration effort. Out of your analysis comes a sense of how much time and resources are required to do the migrations.
- Preparing the migration environment. Obtain and install the needed software, and establish connectivity between the servers.

Giving enough attention to this step leads to a greater chance of success and can reduce the number of unwanted surprises late in the migration process. Form a migration team that includes someone with solid knowledge of the architecture and implementation of the source system.

3.4 4. Migrate database schema, code, and data

When it comes to an application database migration project, this is the step that often comes to mind first. In this step, the actual database is migrated from the source system to the target system. You'll need to perform these steps to migrate the database.

3.4.1 Move the schema

The schema provides the structure in which the data is stored and organized in the database. As such, you need to move the schema first.

When migrating a schema, there are usually differences in the schema objects that are present in the Oracle database and those objects that can be created in the target database. Due to EDB Postgres Advanced Server's compatibility for Oracle features, many compatibility issues are automatically addressed that other target database options can't address.

The main task during this step is to implement workarounds for those schema objects that aren't compatible or can't be converted automatically. Schema objects are created using a SQL-based data definition language (DDL). You can use EDB's Migration Portal to assess the compatibility of the Oracle DDL with EDB Postgres Advanced Server. It also performs some automatic conversions by way of its repair handlers, which are applied as part of the assessment process.

After performing a Migration Portal assessment, the next step is to verify the results of the DDL migration and resolve any problems identified in the assessment report.

3.4.2 Migrate the database functionality (i.e., database procedural objects)

This includes the code that exists in the database, such as procedures and functions that are called by the application or triggers that the database uses. Like the schema objects, the database code objects are also created using DDL. In addition, the code objects are closely tied to the schema objects, either referencing them or being referenced by them. Therefore, the database code objects are usually migrated at the same time as the other schema objects.

As with the other schema objects, the main activity in this step is to identify and implement workarounds for database code objects that aren't compatible or can't be automatically converted. Since Oracle's procedural language (PL/SQL) is different from other database native built-in languages, this can often be the most challenging task in a database migration, especially if the source database contains a large number of code objects or the code objects consist of many lines of complex code. This is where EDB Postgres Advanced Server's PL/SQL implementation, which natively provides many of the most commonly used PL/SQL features, becomes significant. These features can greatly reduce the effort required for completing this step. Migration Portal assesses Oracle database code objects in addition to the other schema DDL and has features to update and reassess incompatible objects.

3.4.3 Migrate the data in the database

When migrating the data, there are two basic options. The first is to move all of the desired data using a snapshot process (copy of all data at a particular point in time). This option is often chosen when the database is relatively small or, in the case of a production migration, the application downtime required to complete the migration is acceptable.

You can perform snapshot replication periodically to refresh the contents of the target database with the current contents of the source database. However, such periodic refreshes usually require removing all the previously copied data from the target system and migrating all of the data again. The second option is to migrate the data on an ongoing, continuous basis using change data capture (CDC) techniques. As part of every change data capture process, the initial load is done by some sort of snapshot. However, after the initial snapshot, a change data capture mechanism is used to identify later changes made in the source database and transfer the changes to the target database. The CDC-based data migration option is usually chosen when minimal downtime is required and it's important to keep the target system in sync with data changes being made in the source system.

You can use the EDB Migration Toolkit to perform snapshot-type data migrations and the EDB Replication Server to perform CDC-based data migrations from Oracle to Postgres. Although snapshot-based data migration and CDC-based data migration are the two main approaches, you can use other hybrid or custom approaches to provide solutions to special data migration needs that might not be addressed by the two main methods. For example, sometimes it helps to pull data by way of queries over database links or direct connections to the source database from the target database.

3.5 5. Migrate interfaces and application

Convert applications to work with the newly migrated Postgres database. The application is the code external to the database that interacts with the database.

Migrating the application involves:

- Updating the application connection strings and drivers (e.g., JDBC, ODBC, OCI, .NET). Applications that use open standard connectivity such as JDBC or ODBC normally require changes only to the database connection strings and selecting the EDB driver. Review the following driver-specific documentation for more information on installing the driver and configuring the application to connect to the EDB Postgres Advanced Server database: - EDB JDBC Connector - EDB ODBC Connector -

  EDB .NET Connector - EDB OCL Connector (an EDB version of the Oracle OCI

  driver)

For .NET applications, you'll need to update applications that use the Oracle .NET driver to replace calls to Oracle-driver-specific class names to their EDB driver equivalent names.

- Converting embedded SQL in the application. The SQL might have been built to work with the source database. Sometimes there's a specific syntax that Oracle supports that might not be supported in Postgres. In this case, you will need to convert that SQL so that it works with Postgres. EDB's compatibility for Oracle features built into EDB Postgres Advanced Server and into the EDB drivers reduces the amount of conversion required.
- Updating application code as needed. In some cases, an application can contain code that includes Oracle driver-specific calls that aren't available in the Postgres driver. You must update such code to implement Postgres-based workarounds.
- Migrating applications. Depending on the migration project, there might be some movement of the actual application—where it's hosted and its configuration—that needs to take place along with migrating the source code. In general, this won't result in any compatibility concerns. However, it's important to verify the application's database connectivity, functionality, and performance in its new environment. In addition, when migrating an Oracle application to Postgres, you might need different techniques, tools, and configurations to address connection pooling and load balancing needs.

3.6 6. Migrate reports and management tools

In this phase, you'll migrate reports, DBA utilities, and scripts. Organizations often have reporting mechanisms and other management tools that also query and interact with the database in support of the core applications. Since some of these artifacts were built specifically to support Oracle-based processes, they might not all apply or be needed in the post-migration system. Thus, it's a good idea to first review the inventory of reports, utilities, scripts, and similar artifacts to determine the ones to leave behind, to replace, and to update.

Updating a script or utility often involves updating any incompatible SQL built for Oracle to be compatible with the target database. It might also involve updating them to work with the command line interfaces provided by the target database instead of the Oracle command line interfaces. For example, SQL*Plus is Oracle's primary command line interface, and psql is the one provided with Postgres. Although SQL statements can be issued in both interfaces, the other supporting commands and keywords they provide are very different. Many Oracle scripts and utilities are built using SQL*Plus commands. To get these scripts to run with psql, you need to convert them to to use psql commands instead.

Fortunately, EDB provides EDB*Plus, which is a SQL*Plus work-alike utility that understands some of the most commonly used SQL*Plus commands. This capability, coupled with EDB Postgres Advanced Server's other built-in compatibility for Oracle features, lets you run many existing scripts built for Oracle with EDB Postgres Advanced Server. EDB Postgres Advanced Server users also can use a version of psql that's extended to work with the compatibility-for-Oracle features built into EDB Postgres Advanced Server. After becoming familiar with psql, users often decide

to use it instead of EDB*Plus to take advantage of its capabilities. Therefore, you might still choose to convert your SQL*Plus-based scripts and utilities to psql-based ones.

Another popular interface provided by Oracle is SQL*Loader, which is a tool used to bulk load data into Oracle. EDB provides a work-alike utility called EDB*Loader that you can use to load data into EDB Postgres Advanced Server. Like EDB*Plus, EDB*Loader provides a familiar interface to Oracle users that you can use with EDB Postgres Advanced Server. Using this tool reduces the required changes to existing data loading procedures. Postgres native methods for loading data are also available to EDB Postgres Advanced Server users. If you choose these methods, you'll need to update any data loading processes based on Oracle-specific capabilities.

You might use other reports or management tools to retrieve information from the Oracle database. Assess the SQL queries issued by these tools to determine whether they're compatible with the target database. Assuming the queries apply to the new system, you'll need to update them if you identify SQL incompatibilities or other operational differences.

3.7 7. Test the migration

Whatever the size of your migration, you'll need to validate that the migrated application operates as expected. You'll be testing all along, but at some point, you need to do a formal set of tests to:

- Validate that the data was fully migrated to the target database and is consistent with the source database. - Verify the functionality of the database and the application. - Validate that the performance of the database and application are acceptable.

For this step, it's important to set up a test environment that allows you to adequately test the functionality and performance of the database and application. Run these application and database tests with the actual application or a copy of the application under workloads simulating the expected production workloads and in an operating environment similar to the production environment. The functional tests must provide enough coverage to exercise application code and interact with database objects that are used less often under normal workloads.

When it comes to data validation, the tools and processes used to compare the data in the source and target databases need to make it easy to identify differences. If differences do exist, you need to perform an analysis to understand why they occurred and whether some changes in the migration process are needed. Depending on how often the data in the target system is refreshed with data from the source system, you might need to perform data validation checks more than once. In the case of an ongoing CDC-based data migration, establish a plan for validating the data periodically.

In this step, it's also a good idea to set up and exercise the tools and processes you will use to meet availability requirements, perform backup and recovery operations, and monitor and manage the database. Doing so helps to identify any changes required to operational procedures or the configuration of the tools and related systems in preparation for their use in the production environment.

One of the main goals of this step is to identify possible issues to address before proceeding with a final migration to the production system. If you find any issues and correct them, you need to perform more testing. When you're satisfied with the results of the preproduction migration tests, you can perform the final preparation of the production environment.

3.8 8. Optimize and configure the post-migration system

In this phase, you'll ensure that all needed software is installed and update the production environment with changes identified and verified during the testing phase. You'll also configure and tune the system based on recommendations derived from previous performance testing results. You might need to adjust database parameters and application settings to prepare the system to run optimally. Also apply any query tuning-related updates that you identified and verified in the testing phase.

This phase is also the time to finish addressing your high availability, disaster recovery, and security requirements, including how users and applications authenticate to the database and who's authorized to perform each action. As needed, update standard operating procedures (SOPs) for ensuring these requirements are met and for monitoring and maintaining the database.

3.9 9. Complete cutover/go live

This phase consists of the following:

- Completing the migration of data into the production database. If data is being migrated from the source database using a CDC-based approach, disable further changes to the source databases to ensure that the new database becomes in sync with the old database. In general, you can continue to allow read-only queries of the source database if you want.
- Setting up a rollback option. Set up a rollback configuration so that the old database is receiving updates from the new database and it'll be there as a fallback position in case an issue occurs with running the applications with the new database. This way, you can go back to running the application against the original source database. Moving data back to the target database is often done through a change data capture mechanism.
- Configuring the system to begin using the migrated application with the new database. This is typically done only after verifying the new database has received all the final updates from the old database.
- Testing the new system until a go/no-go decision is made. Test the new application and database for some period of time. Once you're confident that the application is performing well in the new environment, you can declare that the application is a "go" for full production use.
- Finishing cutover to production. Once you complete the cutover and are satisfied with it, usually you'll remove the fallback position of sending the data changes back to the target data source database. At that point, you'll run only in the new production system.

CHAPTER

FOUR

EDB CAPABILITIES FOR THE MIGRATION JOURNEY

One advantage of performing the migration from Oracle to EDB Postgres Advanced Server is the experience EDB has in performing these migrations. For each phase, we offer tools and services that make migrations easier. We can also discuss your business requirements and help you with the decision to migrate and the approach that works best for you.

4.1 Capabilities used in a migration journey

Before identifying the tools and services that EDB provides to assist with a migration, it's worth reviewing the *capabilities* that are needed in each step of the migration journey. In the context of this discussion, the term capabilities can refer to either software tools or workforce skills and experience. While all the steps in the migration journey benefit from being performed by experienced individuals, many of them are more easily performed when an appropriate set of tools are used.

These are some of the essential types of tools and skills that are needed to successfully complete each step of the migration journey.

4.1.1 Analyze feasibility and alternatives

- Tools and ability to assess the feasibility of moving the source database to the target database
- Tools and ability to categorize migration complexity

4.1.2 Plan the migration

- Ability to estimate migration level of effort
- Ability to prioritize the order of migrations
- Ability to to define nonfunctional requirements and design a high-level solution

4.1.3 Migrate database schema, code, and data

- Database compatibility
- Tools and ability to convert incompatible objects
- Tools and ability to perform snapshot- or CDC-based data migration

4.1.4 Migrate interfaces and application

- Database driver/connector compatibility
- Tools to parse and assess application SQL
- Tools or ability to refactor application code and convert

4.1.5 Migrate reports and management tools

- Tools to parse and assess SQL in reports and utilities
- Ability to convert incompatible SQL

4.1.6 Test the migration

- Tools to help validate data consistency
- Tools to assist with functional and performance validation and ability to perform these activities

4.1.7 Optimize and configure the post migration system

- Tools and ability to assist with database, query, and application tuning
- Tools and ability to address and configure high availability, data recovery, security, authentication/authorization requirements

4.1.8 Complete cutover/go live

- Tools and ability to perform reverse data migration to support rollback
- Tools to help validate data consistency
- Tools and ability to operate and maintain the migrated system

4.2 EDB migration software and services

The table shows a mapping of the migration-related capabilities that EDB provides to the migration journey step they support. A summary description for each of the capabilities is provided after the table. More details can be found in the product documentation and description of package services on the EDB website.

When reviewing the table, the following apply: **Bold=EDBtool** *Italic = EDB professional services package* [Bracketed = Compatibility with Oracle]

Journey Step	EDB Capability
Analyze feasibility and alternatives	• **Migration Portal** - *Migration Assessment Service*

continues on next page

Table 1 – continued from previous page

Journey Step	EDB Capability
Plan the migration	• *Migration Assessment Service* - *Enterprise Architecture Service* - *Solution Design Service*
Migrate database schema, code, and data	• [EDB Postgres Advanced Server] - **Migration Portal** - **Migration Toolkit** - **Replication Server** - **Database Links and dblink_ora Functions** - *Quick Deploy Service* - *Embedded Postgres SME Service*
Migrate interfaces and application	• [EDB Postgres Advanced Server] - **[EDB Database Connectors]** - *Embedded Postgres SME Service*
Migrate reports and management tools	• **[EDB*Plus]** - **[EDB*Loader]** - *Embedded Postgres SME Service*
Test the migration	• **LiveCompare** - **Postgres Enterprise Manager (PEM)**[^1] - *Embedded Postgres SME Service*
Optimize and configure the post migration system	• **Postgres Enterprise Manager (PEM)**[^1] - **Barman** - **Failover Manager**[^2] - **Postgres-BDR**[^3] - *Performance Tuning Service* - *Monitoring Best Practices Service* - *Backup Best Practices Service* - *Embedded Postgres SME Service*
Complete cutover/go live	• **Replication Server** - **LiveCompare** - *Embedded Postgres SME Service*

4.2.1 EDB Postgres Advanced Server

EDB Postgres Advanced Server is an enhanced version of PostgreSQL. Not only does it contain all the features of core PostgreSQL, it also includes built-in compatibility with Oracle and other enterprise grade capabilities, including enhanced security features. EDB Postgres Advanced Server provides compatibility with Oracle in the following areas:

- Oracle-specific and syntax-compatible database object types
- Oracle-specific data types
- Oracle-specific SQL extensions
- Oracle PL/SQL support as a built-in native procedural language
- Oracle data dictionary views (i.e., ALL, DBA, USER_ views)
- Oracle built-in PL/SQL packages

These features simplify migrations from Oracle. EDB Postgres Advanced Server compatibility features mean there are fewer things to convert because things that worked in Oracle will work in EDB Postgres Advanced Server. Although EDB Postgres Advanced Server doesn't fully implement all Oracle features, the compatibility with Oracle features that

are implemented are some of the most commonly used constructs. In addition, EDB continues to add compatibility features with each major release. See

> 'Comparison of EDB Postgres Advanced Server with Oracle' for information on compatibility between Oracle and EDB

Postgres Advanced Server.

EDB Postgres Advanced Server is available to install on various Linux platforms and Windows versions. See Platform Compatibility on the EDB website for the full list of supported platforms. In addition, you can deploy EDB Postgres Advanced Server as a BigAnimal service and in Kubernetes environments using the EDB Postgres for Kubernetes operator .

4.2.2 Migration Portal

Migration Portal is a free online service provided by EDB to assess the compatibility of Oracle database schemas with EDB Postgres Advanced Server. After you create a Migration Portal project and upload a SQL-formatted data definition language (DDL) file for the Oracle database, the Migration Portal analyzes the compatibility of the DDL objects, applies repair handlers to fix a set of known incompatibilities, and reports the results. You can then review the results, inspect all the DDL objects, update the objects with errors to fix the issues, and reassess the project in the Migration Portal. When the DDL in the migration project is fully converted (i.e., made 100% compatible), the Migration Portal can help load that schema into an EDB Postgres Advanced Server instance.

4.2.3 Migration Toolkit

Migration Toolkit is a Java-based command-line tool that supports several different database migration use cases. However, in the context of an Oracle-to-EDB Postgres Advanced Server migration, its primary use is to perform snapshot type data migrations. It provides a rich set of command options to control how and which set of source information is loaded into the target database. A one-time trial period is available, but long-term use requires an EDB database subscription.

4.2.4 Replication Server

Replication Server is a Java-based application that provides both a graphical user interface and a command line interface. It provides a robust data platform for replicating data from non-PostgreSQL databases, including Oracle, to Postgres in single-master mode. It provides a change data capture (CDC) based option for migrating data from Oracle as an alternative to Migration Toolkit. This makes it especially useful when application downtime is a concern. Replication Server also supports replication from Postgres to Oracle. As a result, you can use it to establish a fallback option in case issues are encountered during the final stages of the migration.

A one-time trial period is available, but long-term use requires an EDB database subscription.

4.2.5 LiveCompare

LiveCompare is a command line utility that you can use to compare any number of databases, including Oracle and Postgres databases, to verify they are identical. After comparing the databases, the tool generates a comparison report, a list of differences, and data manipulation language (DML) SQL scripts. You can apply the DML and fix the inconsistencies in any of the databases.

LiveCompare supports several different use cases, but in the context of migrations you can use it to validate data that was migrated from one database to another. When run after migrating data from Oracle to your new Postgres database, its comparison reports help you to know if your data was fully migrated or if you have data inconsistencies to address.

If you're migrating data back to Oracle for a fallback position, LiveCompare can make sure that the two databases have the same data.

You can use LiveCompare as part of a one-time trial period, but long-term use requires an EDB database subscription.

4.2.6 Oracle-compatible database connectors

EDB provides versions of PostgreSQL database drivers (connectors) that are enhanced to include compatibility with Oracle features to better support running applications originally built for Oracle to run on EDB Postgres Advanced Server. The following Oracle-compatible Postgres-based connectors are available:

- EDB JDBC Connector
- EDB ODBC Connector
- EDB .NET Connector

EDB also provides the EDB OCL Connector to allow Oracle Call Interface (OCI) based applications to run against EDB Postgres Advanced Server.

The table highlights some of the important compatibility features provided with the EDB enhanced connectors.

Oracle compatibility feature	JDBC	ODBC	.NET	OCL
PL/SQL Support	X	X	X	X
REF_CURSOR - OracleTypes.CURSOR	X	X	X	X
User-defined Exceptions - vendor code	X	X		X
Named Parameters - parameter names	X	X	X	X
Data Types - VARCHAR2 , STRUCT, ARRAYS	X	X	X	X
STRUCT - Enhanced Manipulation	X		X	X
Upper Column Names - (OPTIONAL)	X			
Multiple INOUT/OUT parameters	X	X	X	X

The EDB enhanced database connectors are available for use with EDB Postgres Advanced Server and require an EDB database subscription.

4.2.7 Other EDB software

4.2.8 EDB*Plus

EDB*Plus is a Java-based command line interface to EDB Postgres Advanced Server. EDB*Plus accepts SQL commands, SPL anonymous blocks, and EDB*Plus commands. EDB*Plus commands are compatible with Oracle SQL*Plus commands, so you can use the scripts you've already built for Oracle against the EDB Postgres Advanced Server database.

4.2.9 EDB*Loader

EDB*Loader is a high-performance bulk data loader that provides an interface compatible with Oracle databases for EDB Postgres Advanced Server. The EDB*Loader utility loads data from an input source—typically a file—into one or more tables using a subset of the parameters offered by Oracle SQL*Loader.

4.2.10 Database links and dblink_ora functions

Database links and *Database links and dblink_ora functions* functions are Postgres-extension-enabled capabilities that let you pull data from Oracle to Postgres by way of SQL queries and database function calls. You can also use these technologies to help migrate data from an Oracle database. Both of these methods use the Oracle Call Interface (OCI) to connect to Oracle. After connecting, use a SQL statement to select the data from the "linked" Oracle database and insert the data into the EDB Postgres Advanced Server database.

When deciding to use one of these two options, the database link option is usually the preferred one as it's easier to use. However, the dblink_ora functions help you to migrate some data types that the database link option doesn't, such as BLOB and CLOB data. Also, it supports using Oracle-specific functions in the queries that are executed in the Oracle database. This can provide some flexibility in extracting certain types of information, like JSON data embedded in CLOB or BLOB columns. Both of these capabilities are provided with EDB Postgres Advanced Server, but you need to set them up to use them.

4.2.11 EDB Postgres Enterprise Manager (PEM)

EDB Postgres Enterprise Manager is a comprehensive database monitoring and management application based on the open source

pgAdmin 4 project, which is also supported by EDB. It provides a

graphical user interface for inspecting, monitoring, managing, and querying your PostgreSQL and EDB Postgres Advanced Server databases and the systems on which they run. It provides many reports, dashboards, probes, and alerts that you can use to monitor the health and performance of your databases. PEM also includes many database administration and database development features. PEM's features are often used as part of a database migration to monitor and tune performance and to update database object definitions as needed. You can use PEM to monitor and manage the Postgres databases that are under an EDB database subscription.

4.2.12 Failover Manager

Failover Manager is a high-availability solution for managing Postgres database clusters, enabling high availability of primary-standby deployment architectures using streaming replication. Failover Manager provides a Postgres primary database node automatic failover to a standby database node in case of a software or hardware failure. In the context of a database migration, it isn't used to perform or test the migration. It's an operational tool that you can use in the post-migration system to address certain high-availability requirements. You can use Failover Manager to manage primary and standby databases that are under an EDB database subscription.

4.2.13 Postgres-BDR

Postgres-BDR is a PostgreSQL extension providing multi-master replication and data distribution with advanced conflict management, data-loss protection, and throughput up to five times faster than native logical replication. It enables distributed PostgreSQL clusters with high availability up to five 9s. For more extreme high availability scenarios, we suggest Postgres-BDR versus EDB Failover Manager. Although it isn't a tool you use to perform an Oracle-to-EDB Postgres Advanced Server migration, if it's intended to be used in the post-migration system, you need to deploy it in the target system and test it to identify any application or database changes needed to support its use. Postgres-BDR requires an EDB subscription to use.

4.2.14 Barman (Backup and Recovery Manager)

Barman is an open-source administration tool supported and maintained by EDB for remote backups and disaster recovery of Postgres servers. Having good backup and recovery procedures and tools is essential to the post-migration operating environment. In addition, these tools are often integrated into processes to support migration testing and related development efforts. In addition to Barman, EDB also supports pgBackRest , a popular Postgres backup and recovery tool, as part of its subscription offerings.

4.2.15 EDB professional services

EDB offers the following professional services packages that are often used in support of a database migration:

- **MigrationAssessmentService** to perform a detailed analysis of your database. From that analysis, you'll learn how difficult the migration will be and how much time and effort it will take to perform the database migration.
- **EnterpriseArchitectureService** to help define your operational, infrastructure, and environment requirements for the migrated system and recommend architectures to support your availability and scalability needs.
- **SolutionDesignService** to recommend a Postgres database solution to meet your database strategy, service-level agreements, and infrastructure needs.
- **QuickDeployService** to help you understand how to deploy Postgres in a highly available, scalable, and secure way based on standard reference architectures and best practices.
- **PerformanceTuningService** to help tune your database to meet application needs.
- **MonitoringBestPracticesService** to help you establish good monitoring practices for operating and maintaining your post-migration databases.
- **BackupBestPracticesService** to help you establish good backup and recovery practices to ensure you never lose your data.
- **EmbeddedPostgresSubjectMatterExpert(SME)Service** to augment your staff with Postgres expertise to guide you through the various stages of a migration and post-migration operations.
- **Training** to help your staff become proficient at operating and maintaining your post-migration Postgres databases

For more information about these and other EDB packaged services see the Consulting Services page.

CHAPTER

FIVE

'COMPARISON OF EDB POSTGRES ADVANCED SERVER WITH ORACLE'

As you assess the feasibility of the migration from Oracle to EDB Postgres Advanced Server, you'll want to understand the similarities and differences between the two database technologies. EDB Postgres Advanced Server is an enhanced version of PostgreSQL that includes a number of built-in Oracle compatibility features. Although these compatibility features make it more similar to Oracle than core PostgreSQL, differences still remain. The intent of the comparison that follows is to give you a feel for the capabilities and features provided by Oracle that will migrate automatically or with minimal effort and those that will require more effort.

The following topics are covered in this comparison:

- *Comparison of general characteristics*
- *Comparison of database features*
- Comparison of database operations-related capabilities
- *Comparison of application development capabilities*
- *Comparison of nonrelational data support*
- *Notable differences*

Having this understanding will help you to decide if EDB Postgres Advanced Server is a good option to replace your Oracle database. This information is also useful in planning the work required to migrate your database to EDB Postgres Advanced Server. After you decide to migrate to EDB Postgres Advanced Server, you can refer to the Knowledge Base available in the *Migration Portal* **PortalWiki** tab for more detailed workarounds of common migration issues.

5.1 Comparison of general characteristics

Prospective users must understand a few foundational details when comparing Oracle's database with the EDB Postgres Advanced Server database. Having an understanding of these basic characteristics is especially valuable during discussions between Oracle experts and Postgres experts.

5.1.1 Terminology

Before looking at the compatibility issues in detail, be aware that there are differences in nomenclature used in many SQL-based products. The table shows some of these differences.

Oracle	EDB Postgres Advanced Server
Table or index	Table, index, or relation
Row	Row or tuple
Column	Column or attribute
Data block	Page—When block is on disk
Page	Buffer—When block is in memory

In addition, each instance of EDB Postgres Advanced Server is referred to as a cluster. A cluster is a collection of databases that is managed by a single program instance. It consists of a data directory that contains all data and configuration files. You can refer to it in two ways: by location of the data directory or by port number. A single server can have many program installations, and you can create multiple clusters.

5.1.2 General capabilities

Both Oracle and EDB Postgres Advanced Server are mature, enterprise-class, object-relational databases that meet the industry standards for atomicity, consistency, isolation, and durability (ACID) compliance. Both were developed from the same IBM research on System R and designed to solve many of the same problems. These database programs have many similarities.

General capabilities	Oracle	EDB Postgres Advanced Server (EPAS)
Design origin	Commercial implementation based on IBM's original research for System R	Academic implementation (UC Berkeley) based on IBM's original research for System R
Continuous development	Since 1979	PostgreSQL development started in 1986. EPAS development started in 2004. EPAS is based on PostgreSQL and continuously merged.
Object relational database	Yes	Yes
Processing architecture	Process based and thread based	Process based
Full ACID compliance	Yes	Yes
Multiversion concurrency control	Yes	Yes
Multitenant architecture	Yes	Yes
Automatic workload management	Yes	No
Enterprise database management	Oracle Enterprise Manager	EDB Postgres Enterprise Manager
Multicore support	Yes	Yes
Write-ahead durability	Redo logs	Write-ahead log
Disk-read buffering	Yes	Yes

5.1.3 Capacities

When considering a new database, you need to understand whether a new solution has the capacity to support existing application data designs, workloads, and anticipated growth. Applying the capacity of a new solution to your workloads and future applications means understanding how it supports data across multiple structures in the database.

Capacities	Oracle	EDB Postgres Advanced Server
Max. database size	Unlimited	Unlimited
Max. table size	4 GB x Block Size	32 TB
Max. row size	4 TB	1.6 TB
Max. field size	For BLOB(4 GB - 1) x DB_BLOCK_SIZE initialization parameter	1 GB
Max. rows per table	Unlimited	Unlimited
Max. columns per table	1000	250 - 1600 depending on column types
Max. indexes per table	Unlimited	Unlimited

5.2 Comparison of database features

To assess the feasibility of your migration, look at the tables that follow to understand how the elements you want to migrate are handled in an Oracle-to-EDB Postgres Advanced Server migration.

5.2.1 Tables legend

The tables use these words or symbols to denote compatibility:

Yes/No — Denotes whether the feature or characteristic is supported in the database.

✓ — The feature operates in a manner compatible with Oracle, allowing you to continue using or migrate your existing Oracle skills, program code, or data.

5.2.2 Tables and partitioning

The range of constructs in the database and how much flexibility DBAs have in organizing these structures can affect performance as well as maintenance and other operational requirements. The ability to partition a database improves performance, for example. Organizing data into distinct structures and distributing them across the infrastructure also improves manageability, availability, and load balancing. Materialized views allow DBAs to replace slow, resource-intensive runtime queries, complex joins, or lengthy scans of data with simple, faster reads from prejoined, presorted, and stored results.

Oracle Enterprise	EDB Postgres Advanced Server	Notes
Temporary tables	Partial	Global temporary tables aren't supported in Postgres.
Views	Yes	

continues on next page

Table 4 – continued from previous page

Oracle Enterprise	EDB Postgres Advanced Server	Notes
Materialized views	Yes	• Postgres doesn't provide an automatic refresh feature. Triggers can be used as part of a Postgres solution. - Postgres doesn't provide an incremental refresh option, only full refresh.
Partitioning	Yes ✓	
Partition by range	Yes ✓	
Partition by hash	Yes ✓	
Partition by list	Yes ✓	
Subpartitioning	Yes ✓	
Interval partitioning	Yes ✓	
Partitioned indexes	No	
ANSI constraints	Yes	
Tablespaces	Yes	In Oracle and Postgres, tablespaces are used to organize and specify where tables, indexes, and other database files are stored. However, unlike Oracle, in Postgres, tablespaces are directories in which automatically generated table, index, and other database files are stored.
Index organized tables	No	In Postgres, you can cluster a table by an index, providing similar performance boosts when reading data from a presorted structure.
Sequences	Yes ✓	• In Postgres, some keyword options aren't available. - Some difference in maximum sequence value.

5.2.3 Data types

Data types provide ways for a DBMS to define, implement, and use information in the system and put constraints on how data is interpreted by the database when multiple data types are in use. EDB Postgres Advanced Server has strong compatibility with Oracle data types and is highly extensible, allowing it to quickly support new and emerging data types and workloads as they become popular.

Data types	Oracle Enterprise	EDB Postgres Advanced Server
Type system	Static + dynamic (through ANYDATA)	Static integer NUMBER NUMBER ✓, DEC , NUMERIC, SMALLINT (16-bit), INT, BINARY_INTEGER, PLS_INTEGER, INTEGER (32-bit), BIGINT (64 bit)
Floating point	BINARY_FLOAT, BINARY_DOUBLE	BINARY_FLOAT ✓, BINARY_DOUBLE ✓, FLOAT, REAL (32-bit), DOUBLE PRECISION (64-bit)
Decimal	NUMBER	NUMBER ✓, DEC, DECIMAL, NUMERIC

continues on next page

Table 5 – continued from previous page

Data types	Oracle Enterprise	EDB Postgres Advanced Server
String	CHAR, VARCHAR2, CLOB, NCLOB, NVARCHAR2, NCHAR, LONG (deprecated)	CHAR ✓, VARCHAR ✓, CLOB ✓, NCLOB ✓, NVARCHAR2 ✓, NCHAR, CHARACTER, TEXT, CHAR, VARYING, CHARACTER VARYING, VARCHAR
Binary	BLOB, RAW, LONG RAW (deprecated), BFILE	BLOB ✓, RAW ✓, LONG RAW ✓, BYTEA (no compatible type for BFILE)
Date/time	DATE, TIMESTAMP (with/without TIMEZONE), INTERVAL	DATE ✓ TIMESTAMP (with/without TIMEZONE), INTERVAL ✓, TIME (with/without TIMEZONE)
Boolean	Not Available	BOOLEAN
ROWID	ROWID	ROWID
XMLTYPE	XMLTYPE	XMLTYPE
Key-value	Requires NSWLDB which is a separate database program	Yes, is integrated into the core database
JSON	Use VARCHAR2, CLOB, and BLOB with is_json check constraint.	JSON and fast binary JSONB with 58 JSON operators, functions and relational json converters
Spatial / Geospatial	Yes	Yes
Other	IMAGE, AUDIO, VIDEO, DICOM	ENUM, POINT, LINE, LSEG, BOX, PATH, POLYGON, CIRCLE, CIDR, INET, MACADDR, BIT, UUID, XML, arrays, composites, ranges, custom
Data domains	Yes	Yes

5.2.4 Indexes

To provide optimal performance for the wide range of supported data types and new workloads using those data types, the database must also support a wide variety of indexes. Postgres is somewhat unique in this regard, especially its GiST index, which allows for easy development of specialized indexes for new data types.

Indexes	Oracle Enterprise	EDB Postgres Advanced Server	Notes
B-Tree	Yes	Yes	
Hash	Yes	Yes	
Expressions	Yes	Yes	
Partial	Yes	Yes	
Reverse	Yes	No	In Postgres, a functional index can be used to reverse the order of a field.
Bitmap	Yes	No	Use of BRIN index might be suitable in some use cases.
Block Range Index	Yes	Yes	
GiST Easy creation of specialized indexes	No	Yes	
GIN Custom inverted indexes	No	No	
K-nearest-neighbor	Yes	No	Available in Oracle using the package DMBS_DATA_MINING and Spatial option.
Full-text search	Yes	Yes	

continues on next page

Table 6 – continued from previous page

Indexes	Oracle Enterprise	EDB Postgres Advanced Server	Notes
Spatial	Yes	Yes	Available in Postgres using free PostGIS extension.

5.2.5 SQL capabilities

EDB Postgres Advanced Server strongly conforms to the ANSI-SQL:2008 standard. It also has transactional DDL, which supports backing out even large changes to DDL, such as table creation. While you can't recover from an add/drop on a database or tablespace, all other catalog operations are reversible. This feature is often used for protection when doing complicated work like schema upgrades. If you put all such changes into a transaction block, you can make sure they all apply atomically or not at all. This lowers the possibility that the database will be corrupted by a typographic or other such error in the schema change, which is particularly important when you're modifying multiple related tables where a mistake might destroy the relational key.

SQL capabilities	Oracle Enterprise	EDB Postgres Advanced Server
Union	Yes	Yes ✓
Intersect	Yes	Yes ✓
Except	Yes	Yes ✓
Inner Joins	Yes	Yes ✓
Outer Joins	Yes	Yes ✓
Inner Selects	Yes	Yes ✓
Merge Joins	Yes	Yes ✓
Common Table Expressions	Yes	Yes
Windowing Functions	Yes	Yes
Parallel Query	Yes	Yes
Query Hints	Yes	Yes ✓
Transactional DDL	No	Yes
Alter Session	Yes	Yes
Dynamic SQL	Yes	Yes

5.2.6 SQL extensions

Oracle has a number of SQL extensions that are popular with Oracle users. While not standard to the SQL language, they provide utility and convenience to DBAs and developers. EDB Postgres Advanced Server supports the ones that EDB customers ask for most often.

Oracle Enterprise extension	EDB Postgres Advanced Server
DUAL	Yes ✓
DECODE	Yes ✓
ROWNUM	Yes ✓
SYSDATE	Yes ✓
SYSTIMESTAMP	Yes ✓
NVL, NVL2	Yes ✓
(+) Syntax for Outer Joins	Yes ✓

5.3 Comparison of application development capabilities

Databases are a foundation of today's data-driven enterprise, and applications are increasingly data intensive. Vendors in turn work to continually enhance their database solutions to support the needs of application developers who seek the flexibility to make choices and simple ways for executing complex tasks. For example, databases that can provide support for multiple server-side languages for triggers, functions, and stored procedures give developers the option to choose their language for both client, middle-tier, and database server programming. Object-oriented capabilities like user-defined object types allow the database to store real-world representations of data, thus making development easier, quicker, and more understandable.

Application development	Oracle Enterprise	EDB Postgres Advanced Server
IDE	Yes SQL Developer	Yes Postgres Enterprise Manager
Database server programming language	Yes PL/SQL (block structured language)	Yes EDB SPL (PL/SQL compatible) (block structured language)
Additional programming languages for database server stored procedures, triggers, and functions	Yes Java, C, .NET (C#, Visual Basic)	Yes PL/pgSQL (PostgreSQL's procedural language), C, C++, PL/Perl, Python, PL/Tcl
JDBC support	Yes	Yes EDB JDBC Connector
ODBC support	Yes	Yes EDB ODBC Connector
.NET support	Yes	Yes EDB .NET Connector
OCI support	Yes	Yes ✓ EDB OCL Connector
PL/SQL debugger	Yes SQL Developer	Yes Postgres Enterprise Manager
Stored procedures	Yes	Yes ✓
Named parameter notation for stored procedures	Yes	Yes ✓
Triggers	Yes	Yes ✓
REF cursors	Yes	Yes ✓
Anonymous blocks	Yes	Yes ✓
Bulk collect/bind	Yes	Yes ✓
Associative arrays	Yes	Yes ✓
Nested tables	Yes	Yes ✓
VARRAYS	Yes	Yes ✓
Hierarchical queries	Yes	Yes ✓
Parallel query	Yes	Yes ✓
PL/SQL supplied packages	Yes	Yes (See *EDB Postgres Advanced Server-compatible package support*)
PRAGMA RESTRICT_REFERENCES	Yes	Yes ✓
PRAGMA EXCEPTION_INIT	Yes	Yes ✓
PRAGMA AUTONOMOUS_TRANSACTION	Yes	Yes ✓
User-defined functions	Yes	Yes
User-defined objects	Yes	Yes
User-defined exceptions	Yes	Yes ✓

5.3.1 EDB Postgres Advanced Server-compatible package support

EDB focuses on the most popular functions in packages. For some packages, not all Oracle functions are supported. For specific details, refer to the *EDB Postgres Advanced Server* documentation.

Package name	Package description
DBMS_ALERT	Functions that allow asynchronous notification of database events by way of an alert. Using this package and triggers, an application can notify itself whenever values of interest in the database are changed.
DBMS_AQ	Database-integrated asynchronous message queuing provides a flexible mechanism for integrating applications across the enterprise by communicating activities and exchanging a variety of information payloads.
DBMS_AQADM	Provides procedures to create and manage queues and queue tables.
DBMS_CRYPTO	Provides functions to encrypt and decrypt stored data.
DBMS_JOB	Was replaced by DBMS_SCHEDULER but is included for compatibility with older Oracle applications.
DBMS_LOB	Functions that allow access to and manipulation of Large Object values.
DBMS_LOCK	Provides a function interface to Lock Management services.
DBMS_MVIEW	Provides procedures to manage and refresh materialized views and their dependencies.
DBMS_OUTPUT	Allows sending messages from stored procedures, packages, and triggers for application or debugging use.
DBMS_PIPE	Functions that allow two or more sessions in the same database instance to communicate with one another.
DBMS_PROFILER	Provides functions to profile stored procedure workloads and identify performance bottlenecks.
DBMS_RANDOM	Useful functions to generate random text, numeric, and date values.
DBMS_REDACT	Redaction prevents a user from seeing all or portions of sensitive data.
DBMS_RLS	Implements row-level security functions in the database, blocking users from seeing each other's data in the same application.
DBMS_SCHEDULER	Job scheduler functions for creating and executing unattended repetitive tasks inside the database.
DBMS_SQL	Permits the use of dynamic SQL in procedures to allow applications to run SQL statements with unknown parameters (such as table name) until runtime.
DBMS_SESSION	Functions with the ability to enable and disable roles.
DBMS_UTILITY	A collection of functions for getting information about various runtime operations and metadata from the database.
UTL_ENCODE	Functions to perform Base64 encoding and decoding of data intended for transport between hosts.
UTL_FILE	Allows database procedures to read and write operating system text files in an I/O stream fashion.
UTL_HTTP	Functions that enable you to make HTTP calls to access information on web servers.
UTL_MAIL	Provides functions to create, manage, and send email from the database including attachments, CC, and BCC.
UTL_RAW	Functions supporting manipulating raw data types.
UTL_SMTP	Provides functions for sending email via SMTP according to the RFC821 specification.
UTL_TCP	Provides procedures and functions to enable PL/SQL applications to communicate with external TCP/IP-based servers using TCP/IP.
UTL_URL	Functions for escaping and "unescaping" URL strings.

5.4 Comparison of nonrelational data support

Nonrelational data	Oracle Enterprise	EDB Postgres Advanced Server
Spatial/location/graph	Yes	Yes
JSON support	Yes Text based	Yes Text and high-performance binary based
Key-value store	NoSQL database	Yes
Support for XML namespaces, DOM, XQuery, SQL/XML, and XSLT	XML DB	No
Compression (tables, files, network, and backups)	Yes	No Postgres performs compression of TOASTED rows
Partitioning	Yes	Yes
Hadoop integration	Yes ETL via Data Integrator Application Adapter for Hadoop	Yes Real-time join with relational data with HDFS Foreign Data Wrapper
MongoDB integration	Yes Golden Gate adapter	Yes Read/write/join with MongoDB Foreign Data Wrapper
Cube, rollup, and grouping sets	Yes	Yes
Transportable cross-platform tablespaces	Yes	No
Full-text search	Yes	Yes
Advanced compression	Yes	No

5.5 Notable differences

A number of differences between Oracle and EDB Postgres Advanced Server are either not yet addressed or are worth noting because of their frequent use.

Oracle Enterprise	EDB Postgres Advanced Server
MERGE	No Postgres UPSERT statements can be used but have different syntax from MERGE. MERGE syntax planned for PostgreSQL 15
Advanced queuing	Yes
Nested procedures/functions	Yes
Pipelined functions	No Pipelined functions are used for table functions. Table functions can be implemented in Postgres via SETOF returning functions. In Postgres, data is returned only after the function completes.
Empty string = NULL	No Empty string = !NULL
Performs many implicit data type conversions such as a number to a string	Partial Many data types need to be explicitly cast to the other data type or an error occurs.