EDB

The Complete Oracle to Postgres Migration Guide:

Move and Convert Schema, Applications and Data

AUTHORED BY:

Raghavendra Rao
Managing Consultant in
Professional Services

POWER TO POSTGRES

Contents

Introduction

Enterprises are increasingly making the move from Oracle databases to open source Postgres. This complete guide to migration gives you everything you need to know about moving your database from Oracle to Postgres.

For anyone on the fence or intimidated by the process of data migration, it recaps the advantages of Postgres over Oracle as a database system and the benefits of adopting Postgres, namely cost, flexibility and customizability. It then breaks the migration process down into individual phases (assessment, schema migration, functional testing, performance testing and data migration) and provides step-by-step instruction on each one. Key differences and incompatibilities between the two database systems are itemized to help users avoid common mistakes. Alternate migration strategies are also weighed, and a list of helpful free data migration tools is provided.

This guide is intended for anyone preparing to make the move from Oracle to Postgres, and should also provide assurance to any current Oracle users considering Postgres but concerned about the complexity of the move.

What is database migration?

Database migration is a process of moving definitions, data and stored procedures from one platform to another, and making application changes. Moving data involves selecting, preparing, extracting, transforming and applying that data from one database to another.

Migration to Postgres will undergo different phases—like picking the right schema, performing compatibility checks, converting incompatible objects, functional and performance testing, data migration and post-migration check.

Why migrate from Oracle to Postgres?

There are many reasons you might choose to migrate from Oracle to Postgres. Here are just a few of the benefits:

1. Cost: In addition to Oracle license costs, using Oracle databases incurs additional costs for features like partitioning and high availability, and expenses can add up quickly. Open-source Postgres is free to install and use.

2. Flexibility: Postgres has open-source licensing and is easily available from public cloud providers, including AWS. With Postgres, you're not at risk of vendor lock-in.

3. Customizability: Because Postgres is open-source, there are countless extensions and add-ons that can improve database performance markedly, and many of them are free to use. With Oracle, similar features quickly add up in cost.

That's not to say that migrating from Oracle to Postgres is not an involved process. Since the data migration is between two relational database management systems (RDBMS), it can be a challenging and time-consuming process due to heterogeneous structure/data types. As such, you want to be sure it's tackled and handled with the right tools. Follow the steps below and you'll be well on your way.

What are the Postgres advantages over Oracle database?

We can briefly say that Postgres advantages over Oracle database are in the following areas or features:

1. Application programming
2. Authentication
3. Extensibility
4. Languages
5. Localization
6. Performance
7. Scalability

1. Postgres vs. Oracle: Application programming

Oracle and Postgres both provide an application API for communicating with the database. However, Postgres is open source. Developers can directly access any Postgres component simply by including the header file in their project.

2. Postgres vs. Oracle: Authentication

Oracle has a built-in authentication system. Postgres relies on host-based authentication and can, therefore, support a wide range of authentication methods. This provides greater flexibility for authentication and the option to delegate the process.

3. Postgres vs. Oracle: Extensibility

Oracle has a mostly proprietary plug-in system, whereas Postgres' extension system is supported by the general community, so thousands of plug-ins are available.

4. Postgres vs. Oracle: Languages

While Oracle has a built-in programming language called PL/SQL, Postgres has not only PL/pgSQL but many others, as well as an extension system that allows users to create additional procedural languages as plug-ins, plus bindings for even more programming languages.

5. Postgres vs. Oracle: Localization

Oracle offers globalization support tools including a globalization development kit and unicode character support. Postgres' localization system services are built-in to provide automatic character encoding and collation support.

6. Postgres vs. Oracle: Performance

Because Postgres can create an unlimited number of nodes in a read cluster, the cost of any particular read operation can be reduced to close to nothing. And because of that, you can tune it differently for every workload. You can do this in Oracle too, but each node has an additional cost.

What are the Oracle to Postgres Migration Steps and Phases?

We can group Oracle to Postgres migration steps or phases as follows:

- Assessment
- Schema Migration
- Functional Testing
- Performance Testing

7. Postgres vs. Oracle: Scalability

Oracle of course has strong vertical read scalability, but Postgres can create a virtually unlimited number of nodes in a read cluster, depending on the resources you have available to dedicate to it.

1. Oracle to Postgres migration: Assessment

Oracle to Postgres migration: Assessment

What are the Oracle to Postgres pre-migration assessment steps and phases?

1a. Assessment
1b. Compatibility assessment
1c. Application code assessment
1d. Architecture assessment and cleanup

1a. Oracle to Postgres migration: Assessment

This is the first step in planning the migration and analyzing the application to estimate how easy or difficult it will be to migrate it from Oracle to Postgres. In this phase, a thorough analysis should be conducted on technology-related issues and to evaluate the compatibility of client, application server, data access and database features.

1b. Oracle to Postgres migration: Compatibility assessment

One concern that is so basic that it can easily be overlooked when considering a move to Postgres is confirming that, if you do not control your own application, the packaged software application you are using certifies for Postgres. If not, you'll either need to convince your application's vendor to add Postgres support, or choose another application.

After verifying the source and target database compatibility, the following prerequisites should be met for data migration:

- Server resources (memory/disk space/network ports opened between source and destination)
- Operating system
- Data migration software and related drivers installed and configured

It should go without saying that your target server resources are large enough and have the scalability to handle the volume of data they will be receiving. If the volume of data is very large, then a purely online migration may not be advisable, and you should consider an export-and-reload approach. It may also be wise to follow a migration strategy that divides the migration into parts (see the section on migration strategies below.

1c. Oracle to Postgres migration: Application code assessment

The more your application code relies on Oracle-specific frameworks, as opposed to open classes, the more intricate your migration becomes. The amount of adjustment you will need to make depends on your application architecture and database connection layer. If you have Java code that uses generic JDBC classes rather than Oracle-specific ones, the conversion should be relatively easy. It should similarly be easy if you use an object-relational mapping (ORM) such as Hibernate or JCA. It could be trivially easy to switch the dialect from Oracle to Postgres, though some adjustments will still need to be made.

The situation is more complex if you use embedded SQL such as Oracle's Pro*C, dynamically built SQL, or link to Oracle-specific libraries such as OCI or the Oracle JDBC classes. Adjusting these requires a solid understanding of the underlying application logic and should be carefully tested.

1d. Oracle to Postgres migration: Architecture assessment and cleanup

The ease or difficulty of a migration can be significantly impacted by the setup and architecture of your database and its contents. Migration presents a good opportunity to clean up your architecture and database contents. Deprecate objects you no longer need, such as old temporary tables or backup copies of data—they're not worth migrating if no one cares about them. If you store large files, like images or PDFs in your database, consider whether you can separate them into a lower-cost storage option to reduce database size and resources needed for backup and restores. You may want to purge static historical data or move into an archival store option.

Moving from Oracle to Postgres also opens the possibility of separating online transaction processing (OLTP) and analytics into different warehouses, which can improve both responsiveness and analytics capabilities.

1e. Oracle to Postgres migration: Schema conversion

Following assessment, the next step in the migration process is to identify and address differences in schema and data formatting between Oracle and Postgres. It is crucial to make these adjustments prior to the data migration to avoid frustrating and time-consuming errors in Postgres.

Postgres supports ANSI SQL standard SQL syntax and data types, whereas Oracle does not support the same standard; additionally, it includes some non-ANSI SQL syntaxes. Using tools, unsupported objects should be identified and then converted manually with Postgres-supported syntax or feature workarounds.

2. Oracle to Postgres migration: Schema migration

Oracle to Postgres migration: Schema migration

What is schema in Oracle and Postgres?

A schema is also known as a “User” in Oracle and has the same name as the user. The default is for each Oracle user to have their own schema. In Postgres, these are not the same, and if you do not explicitly specify a schema, new objects will go to a public schema by default.

- Create a user and schema with the same name
- The first component in the schema search_path is $user, by default

One advantage to the set up in Postgres is that a user can create multiple schemas without having to create separate users, and can grant permissions for creating objects in those schemas to others.

What are the Oracle to Postgres schema migration tools?

There are a number of migration tools available that can help users automate schema conversion. Below are the tools you can try for free for converting the Oracle objects to Postgres.

- Ora2pg—a robust migration tool that connects to an Oracle database, extracts schemas and tables and generates SQL scripts that can be loaded into Postgres.
- Ora_migrator—an extension that uses an oracle_fdw foreign data wrapper to extract data from an Oracle database.
- Orafce—this extension allows you to implement a number of Oracle functions in Postgres. It also provides support for Oracle date formatting and additional Oracle data types.
- EDB Migration Portal—a web-based service for migrating from Oracle to EDB Postgres Advanced Server that features Assessment, Schema conversion and Compatibility reporting.

https://www.youtube.com/watch?v=V_AQs8Qelfc&feature=emb_imp_woyt

In addition, there are a number of commercial conversion tools available, including Amazon’s AWS Schema Conversion Tool (AWS SCT).

Oracle to Postgres migration tools comparison matrix

The below tools comparison matrix will give you a high-level picture of what they can do and how far they get you in the migration exercise to a specific target.

	Assessment	Migration of data objects	Migration of code objects	Migration Data	Approach	Target
EBD Migration Portal				Use EDB MTK	Native + Transformation	Postgres Advanced Server
EDB MTK					Native Only	Postgres Advanced Server
AWS SCT					Transformation Only	PostgreSQL
Ora2Pg					Transformation Only	

What are the schema differences between Oracle and Postgres to pay attention to when migrating?

When preparing for schema conversion, pay special attention to the following differences between Oracle and Postgres.

Columns when migrating from Oracle to Postgres

Until version 12, Postgres did not have any equivalent to virtual columns, so users were encouraged to change these to views when migrating. Now Postgres offers generated columns, which share many traits with Oracle's virtual columns.

Constraints when migrating from Oracle to Postgres

In both database systems the Primary and Foreign Key, Check, Not-Null and Unique, constraints all operate more or less the same way.

Identifiers when migrating from Oracle to Postgres

Oracle converts names of schema, tables, columns and functions to uppercase, unless the name is given in quotes, while Postgres converts them to lower case (also unless given in quotes). As long as the application consistently quotes or does not quote the identifiers, you should be safe.

Indexes when migrating from Oracle to Postgres

- B-tree and descending indexes should function in Postgres
- Reverse key, bitmap, and join indexes are not currently supported.
- Global index is not supported in Postgres

Partitions when migrating from Oracle to Postgres

Hash, List, and Range partitions should all work in Postgres following migration.

Tables when migrating from Oracle to Postgres

CREATE TABLE is mostly compatible, with the following exceptions:

- Postgres lacks global temporary tables. Use temporary tables (LOCAL TEMP) instead.
- Partitioning: Use Inheritance, Triggers, and CHECK Constraints for partition clauses.
- Storage clause parameters (INITRANS, MAXEXTENTS) are not recognized in Postgres and should be removed.
- For the Oracle PCTFREE parameter, replace it with Postgres' fillfactor.

Tablespaces when migrating from Oracle to Postgres

There are differences between Oracle and Postgres' versions of tablespaces, but they serve the same purpose and should work

.

Data types when migrating from Oracle to Postgres

The following chart lists notable differences between Oracle and Postgres data types.

Oracle	Postgres	EDB Postgres Advanced Server	Comment
VARCHAR2(n)	VARCHAR2(n)	VARCHAR2(n) VARCHAR2(n)	Be careful not to confuse 'n' in Oracle and Postgres data types. In Oracle it stands for the size in bytes; in Postgres it stands for the number of characters.
NVARCHAR, NVARCHAR2	VARCHAR or TEXT	NVARCHAR, NVARCHAR2, VARCHAR or TEXT	
CHAR(n), NCHAR(n)	CHAR(n)	CHAR(n), NCHAR(n),	Be careful not to confuse 'n' in Oracle and Postgres data types. In Oracle it stands for the size in bytes; in Postgres it stands for the number of characters.
NUMBER(n, m)	NUMERIC(n,m)	NUMERIC(n,m) NUMBER(n, m)	NUMBER type can be converted to NUMERIC, which is unlimited in size. However, the SMALLINT, INT BIGINT, REAL, and DOUBLE PRECISION data types offer better performance.
NUMBER(4)	SMALLINT	NUMBER(4) SMALLINT	
NUMBER(9)	INT	NUMBER(9) INT	
NUMBER(18)	BIGINT	NUMBER(18) BIGINT	

NUMBER(n)	NUMERIC(n)	NUMBER(n) NUMERIC(n)	If n>=19
BINARY_INTEGER, BINARY_FLOAT	INTEGER, FLOAT	BINARY_INTEGER, INTEGER, FLOAT	
DATE	TIMESTAMP(0)	DATE TIMESTAMP(0)	In Oracle the DATE type returns both date and time, whereas in Postgres the DATE type returns the date without the time.
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMPTZ	TIMESTAMP WITH LOCAL TIME ZONE TIMESTAMPTZ	Oracle has both TIMESTAMP WITH TIME ZONE and TIMESTAMP WITH LOCAL TIME ZONE data types. Postgres' TIMESTAMPTZ is equivalent to TIMESTAMP WITH LOCAL TIME ZONE. If these are confused it can introduce errors.
CLOB, LONG	TEXT	CLOB, LONG TEXT	Postgres' TEXT type is capable of storing up to 1 GB of text data.
BLOB, RAW(n), LONG RAW	BYTEA(1 GB limit) Large object	BLOB, RAW(n), LONG RAW BYTEA(1 GB limit) Large Object	In Oracle,the BLOB datatype is used for unstructured binary data and essentially has no size limit (up to 128 terabytes of binary data). Postgres' BYTEA data type can store up to 1 GB of binary data. Above that limit use Large Object (these are stored in a separate table).
NLS_DATE_FOR-MAT	DateStyle	DateStyle	These are parameters that set the format for how date information is displayed. The default style for Postgres' DateStyle is ISO. Oracle's default is inherited from the NLS_TERRITORY parameter.

What are the challenges while migrating from Oracle to Postgres?

In this section, we will discuss some of the challenges we might face while migrating from Oracle to Postgres. To address these challenges and limitations identified in Postgres, those objects should be rewritten manually with a workaround if they exist.

1 Constraints

While Oracle allows users to disable and enable constraints as often as you want, this is not generally recommended practice for any RDBMS, because it can lead to data corruption if not performed with proper care.

In Postgres, constraints are instead created as deferrable, and the SET CONSTRAINTS command can be used to defer them. The deferrable setting indicates the default time for activating the constraint. If the constraint in Oracle is not deferrable, it will need to be dropped and re-created as deferrable, though it is sometimes possible to alter the constraint without having to drop it. Note: to avoid potential errors or bad data, place the commands for dropping and re-creating the constraint in a transaction, marked with a BEGIN/COMMIT block, which will lock the tables during the transaction..

2 DELETE

The FROM clause for specifying the table is required with DELETE statements in Postgres but not in Oracle.

Oracle:

```
DELETE mytable WHERE column_name = 'column_amount';
```

Postgres:

```
DELETE FROM table_name WHERE column_name = 'column_amount';
```

3 Dropping database objects

In Postgres, permission to drop objects is restricted to only a database table's owner or a super user. It is not a grantable privilege, although membership in the role that owns the object can be granted. If an action in Oracle depends on this ability, it may need to be rewritten or reconfigured.

4 Dual Table

Because the FROM clause is mandatory in Oracle for every SELECT statement, FROM DUAL is used for SELECT statements where the table name is not necessary. Postgres does not require the FROM clause, so FROM DUAL is not necessary and can usually be omitted. If the Dual table is needed in Postgres, it can be generated as a view.

5 Empty Strings and NULL

In Oracle, empty strings have NULL values, but they are not considered NULL in Postgres.
In Oracle, you can check whether a string is empty or not using the IS NULL operator, but in Postgres, it would return FALSE for an empty string (and TRUE for a NULL one).

6 Federation to Foreign Data Wrappers

Oracle's Federation feature allows users to treat tables from other databases as local data. Postgres' foreign data wrappers are more versatile and allow you to connect to a wider range of data.

7 GRANT

The GRANT command behaves similarly in Oracle and Postgres. There are two basic variants—it can be used to grant privileges on a database object and to grant membership to a role. Not all privileges that are grantable in Oracle are grantable in Postgres. For example, granting the trigger privilege to a table allows users to create triggers; but, as opposed to Oracle, only the owner of the table can drop triggers.

8 Hierarchical queries

Postgres does not support the START WITH . . . CONNECT BY syntax that Oracle uses for hierarchical queries. Instead, Postgres uses WITH RECURSIVE.

```
SELECT
    business_name,
    city_name
FROM
    businesses bs
START WITH bs.city_name = 'BOSTON'
CONNECT BY PRIOR bs.business_name = bs.city_name;
```

Postgres:

```
WITH RECURSIVE tmp AS (SELECT business_name, city_name
FROM businesses
WHERE city_name = 'BOSTON'
UNION
SELECT t.business_name, t.city_name
FROM businesses t
JOIN tmp ON tmp.business_name = t.city_name)
SELECT business_name, city_name FROM tmp;
```

9 Joins with (+)

Oracle has a special shorthand (+) operator for performing left and right outer joins. Postgres lacks this feature, so the JOIN command would need to be supplied.

Oracle:

```
SELECT table1.firstname, table2.lastname
     FROM table1, table2
     WHERE table1.customer = table2.customer (+);
```

Postgres:

```
SELECT table1.firstname, table2.lastname
    FROM table1
    LEFT OUTER JOIN table2 ON table1.customer = table2.customer;
```

10 NOT NULL checking

To determine which columns in an Oracle table are NOT NULL, you would use the command CHECK (<column_name> IS NOT NULL).
Postgres instead has a NOT NULL constraint column named attnotnull in pg_attribute, the systems catalog where information about table columns is stored.

11 Packages

Postgres does not have packages, but, using schema architecture, functions and procedures can be grouped. Use the “orafce” migration tool library, which supports some of the standard packages, or EDB Postgres Advanced Server, which has built-in Packages.

12 PL/SQL to PL/pgSQL Conversion

Postgres’ PL/pgSQL procedural language is similar in many respects to Oracle’s PL/SQL. Both are block-structured, imperative languages, with similar formats for assignments, loops, and conditionals. The Postgres documentation has a thorough run-down of considerations that need to be made when porting from PL/SQL to PL/pgSQL.

13 Remote objects

To access remote objects, the DBLINK module or Foreign Data Wrapper(Oracle_fdw) can be used to access any other database.

ROWID, CTID and Identity columns

Postgres does not have an exact equivalent to the ROWID pseudocolumn in Oracle, which provides the address of a row in a table. CTID in Postgres is similar, except that its value changes every time VACUUM is performed. Instead, you can use identity columns, whose value is self-generated when a row is created and never changes. The value can be specified to be GENERATED ALWAYS or GENERATED BY DEFAULT. GENERATED BY DEFAULT allows the user the option to insert or update a value rather than use the system-generated value.

14 Sequences

Sequences have a different syntax in Oracle and Postgres and will need to be updated either manually or using a script.

Oracle:

```
Mysequence.nextval;
```

Postgres:

```
nextval('mysequence');
```

15 SUBSTR

The SUBSTR function behaves differently in Oracle and Postgres. In Oracle the statement SELECT SUBSTR('ABC',-1) FROM DUAL; returns 'C', while the equivalent SELECT SUBSTR('ABC',-1); in Postgres would return ABC. The Orafce migration tool includes a SUBSTR function that returns the same result in both databases.

16 Synonyms

Postgres does not support synonyms. In place of Oracle's CREATE SYNONYM for accessing remote objects, Postgres has you use SET search_path to include the remote definition.

Oracle:

```
CREATE SYNONYM abc.mytable FOR xyz.mytable;
```

Postgres:

```
SET search_path TO 'abc.mytable';
```

17 SYSDATE

Oracle's SYSDATE function returns date and time (in the server's timezone). Postgres does not have a corresponding function, but there are a range of methods for retrieving the date and time for different purposes: statement_timestamp() gives current date and time from the beginning of the current statement; now() and transaction_timestamp() give the date and time from the beginning of the current transaction, and clock_timestamp() gives current date and time from the execution of the function.

18 TO_DATE

The to_date() function in both Oracle and Postgres return the date data type. However, Postgres' date data type provides the date (year, month, day), while Oracle's date data type value provides the date and time (year, month, day, hour, minute, second). To avoid this incompatibility, use Postgres' to_timestamp().

The solution for this incompatibility is to convert TO_DATE() to TO_TIMESTAMP(). If you use Orafce tool then it is not necessary to change anything because Orafce implemented this function so we get the same result as Oracle.

Oracle:

```
SELECT TO_DATE ('20180314121212','yyyymmddhh24miss') FROM dual;
```

Postgres:

```
SELECT TO_TIMESTAMP ('20180314121212','yyyymmddhh24miss')::TIMESTAMP(0);
```

19 Transactions

Oracle always uses transactions, but in Postgres, they must be activated. In Oracle, executing any statement starts a transaction, and it ends with the COMMIT statement. In Postgres, the transaction starts with the BEGIN statement, and it also ends with the COMMIT statement. Transaction isolation levels are the same in Postgres and Oracle, and Read Committed is the default isolation level for both.

20 Transaction error handling

Postgres is built to facilitate transaction management and error handling, with full ACID support and isolation levels. It is also capable of handling run-time errors and provides reliable error codes and messages to PL/pgSQL or application code. These are handled differently from Oracle, though, so changes still need to be made. Here are some tips for optimizing error handling for Postgres:

- Transaction control inside of PL/pgSQL is not permitted: you cannot commit or roll back a transaction inside a stored procedure. Commits and roll backs must be called from the application, so the application that calls the stored procedure must perform the transaction management—starting and committing or rolling back. The stored procedure executes within that calling transaction context. Obviously, if your existing database code has transaction management in procedures, it must be modified.
- When there is a run-time exception during a transaction, the transaction must be rolled back before you can execute another statement, because the transaction is aborted when it encounters the error. The application log will show the following error message:

```
ERROR: current transaction is aborted, commands ignored until end
of transaction block.

Be sure that you have an exception handler set up where errors might
be occurring and either to a savepoint or close the connection before
trying another database operation.
```

- Use a BEGIN…EXCEPTION…END block for exception handling so your code catch any errors that occur. This automatically sets a savepoint before the block and rolls back to it when it encounters an exception. Keep in mind that because exception blocks create a savepoint, they are expensive, so add them carefully.
- Map the error codes and exception types from Oracle to Postgres. While some error codes are the same in both, others are different. Your programming language affects this as well—for example, Oracle-specific JDBC exceptions need to be replaced with either generic cross-database exceptions or Postgres-specific ones.
- Ensuring your Postgres database handles transactions and errors correctly is a critical part of the migration process and usually requires a careful review of the database and application code.

For enterprises looking to retain some oracle functionality in Postgres and wanting to migrate faster without many code rewrites an option is to move to EDB Postgres Advanced Server. **EDB Postgres Advanced server** is Oracle compatible Postgres with native PL/SQL compatibility. The below video demonstrates how the Advanced Server handles Oracle queries: https://youtu.be/n1YCsrdDvdI

3. Oracle to Postgres migration: Functional testing

Oracle to Postgres migration: Functional testing

Functional testing after schema migration from Oracle to Postgres

Before proceeding further, it is important to test the converted schema on a sample dataset. A recommended approach is to load some sample data into Postgres from a source database development or testing environment where there's production sample data, and then set up an application connection using appropriate data access (drivers). After the application has connected to the database, allow it to do full functional testing on the converted objects with DMLs.

It is advisable to load the same sample dataset in both the Oracle and Postgres databases, then test both to make sure the SQL results are identical. Review and address any issues revealed by the functional tests.

4. Oracle to Postgres migration: Performance testing

Oracle to Postgres migration: Performance testing

Postgres performance testing after migration

Performance testing is important in the migration phase because some of the Oracle built-in transactions or features functionality might be slightly different in Postgres and application might see some difference. In this phase, we can capture all those differences and fix them at application, data access (drivers), and database level with proper tuning.

5. The nine steps of the migration journey

Oracle to Postgres migration: Data migration

Data migration from Oracle to Postgres

There are different approaches for data migration and tools available in the market.
Usually, they are classified in three ways

1. Snapshot: Data moved at once
2. Snapshot in Parallel: Data moved in chunks (schema or table)
3. Change Data Capture (Replication): Data loaded continuously.

For approaches 1 and 2, we need application downtime because data is being written one time from Oracle to Postgres, whereas in approach 3 data is loaded continuously, and there is a smaller downtime window. Pick the right data migration approach that fits in the downtime window.

Breaking down the strategies

Let's discuss these three data migration strategies in a little more detail.

Snapshot (One Big Bang)

In this approach, a snapshot of the source database state is taken and applied on the target database. Data is moved from Oracle to Postgres all at once. During the snapshot process, no WRITE operations are allowed on the source database. It's one of the cleaner and easier methods of data migration.

Pros

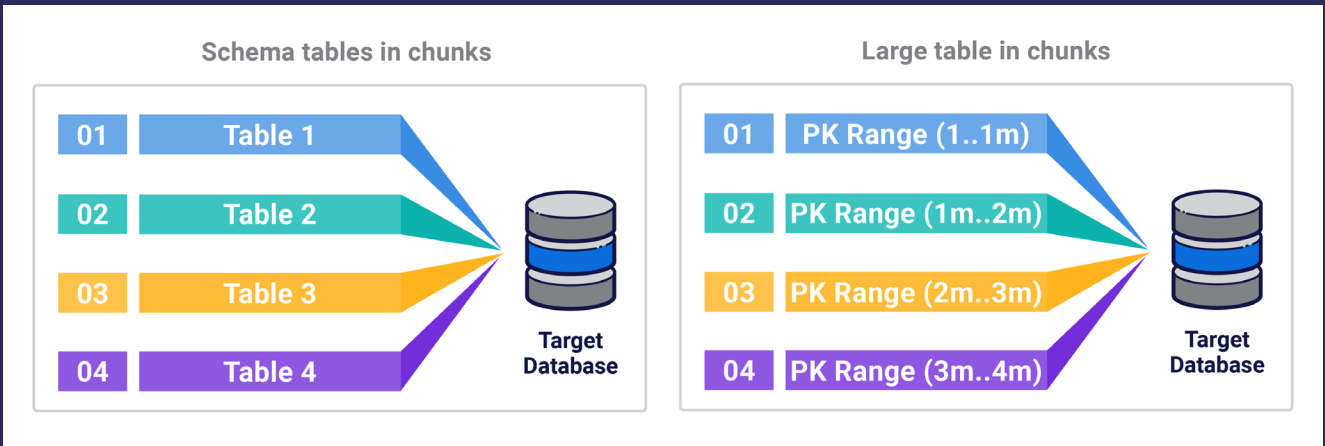
- Data movement will be completed in one g
- No Data Type challenges (LoBs)
- After snapshot, applications can start accessing the target database
- No special configuration required. Easy to manage.

Cons

- The application will be down during the snapshot.
- If a snapshot is interrupted, reinitiation is required.

Snapshot - Parallel in Chunks (Trickle)

Parallel in chunks is another type of snapshot approach where data objects are broken into chunks and snapshots are taken in parallel. Most of the tools support snapshot and the process are invoked in tandem. There are two ways to perform a snapshot in chunks: 1) table by table or 2) a large table split into small sets using primary keys or any unique row identifiers. In this approach, the snapshot duration and downtime window is greatly reduced. Good scripting skills required to prepare data migration tools for table or large table migration.



Pros

- Data moved at one time with less downtime
- Data moved in parallel - table by table or a large table in small sets

Cons

- Application downtime required (less compared to Big Bang approach)
- For large tables broken into small sets, primary key or unique row identifiers are mandatory
- Script required to adjust the parallel approach
- If a snapshot is interrupted, reinitiation is required.

5c. Change Data Capture (CDC Data Sync)

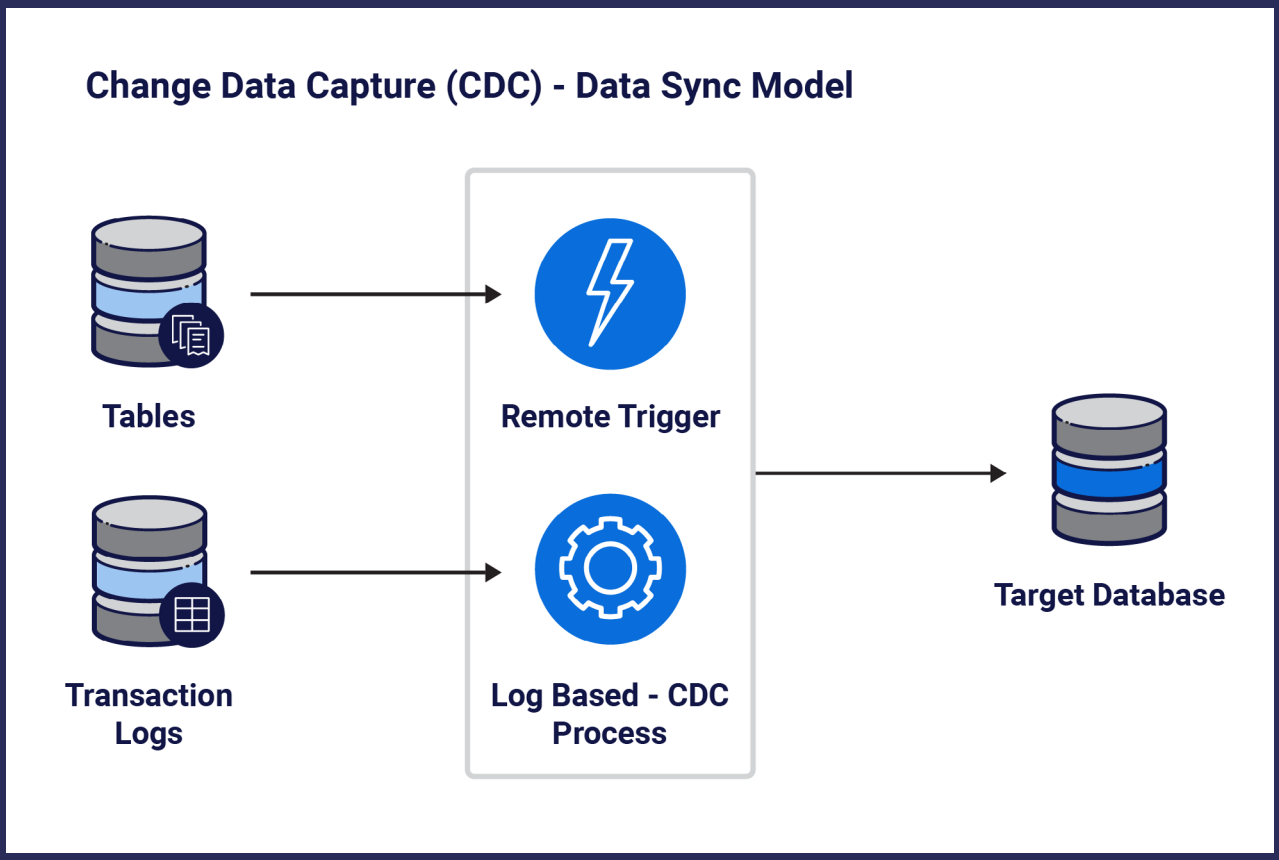
There are different traditional Change Data Capture (CDC) approaches that have been available for decades. In the CDC model data migration, the software is designed to determine/track/capture the data that has changed on the source database in real-time and replay the same on the target database. Today CDC model software is in high demand because they distribute data between heterogeneous databases with low-latency, reliable, and scalable data. Most common CDC approaches for migrating Oracle to Postgres databases are:

- Trigger-based
- Transaction Log-based

Trigger-based (CDC)—a remote trigger is created to capture DMLs (insert/update/delete) before or after and the sequence of transaction events is captured in changelog (shadow tables), then they are processed by the software to replay on the target database.

Transaction Log-based (CDC)—every database will have transaction logs (redos) to store all database transactions/events sequentially and be used in case of database crash recovery. Using native database transaction log plug-ins the transactions (DMLs) can be captured to change logs with some filters, transforms, and aggregations. Later, the sequence of captured DMLs will be replicated to the target database.

Both CDC approaches can be used to replicate data from Oracle to Postgres. However, each has its own merits and limitations, depending on the requirements we can choose one of the available approaches. There are very good software options available to support both CDC approaches.



Pros

- Data loaded continuously to target database (after initial snapshot)
- User can access source database while data loads to target database
- Data sync control (if interrupted, it can be resumed)
- Partial replication (Set of tables can be replicated)

Cons

- Need replication software
- In trigger-based CDC, there could be a slight performance overhead
- No Large Objects support
- Partial to small application downtime (switchover time)
- Only commercial/free to use tools available, no open source.

What Are the Free Tools Available for Data Migration?

There are open and free to use tools available for data migration in three different categories we have discussed in the blog. Below are the set of tools that we came across under those categories:

1. **Snapshot**
   - Ora2pg
   - Others (Full convert, Ispirer)

2. **Snapshot - Parallel in chunks**
   - Schema Tables
     - Ora2pg, ora_migrator
     - EDB Migration Toolkit
   - Large table in sets
     - EDB dblink_ora module
     - Oracle FDW(ora_fdw)

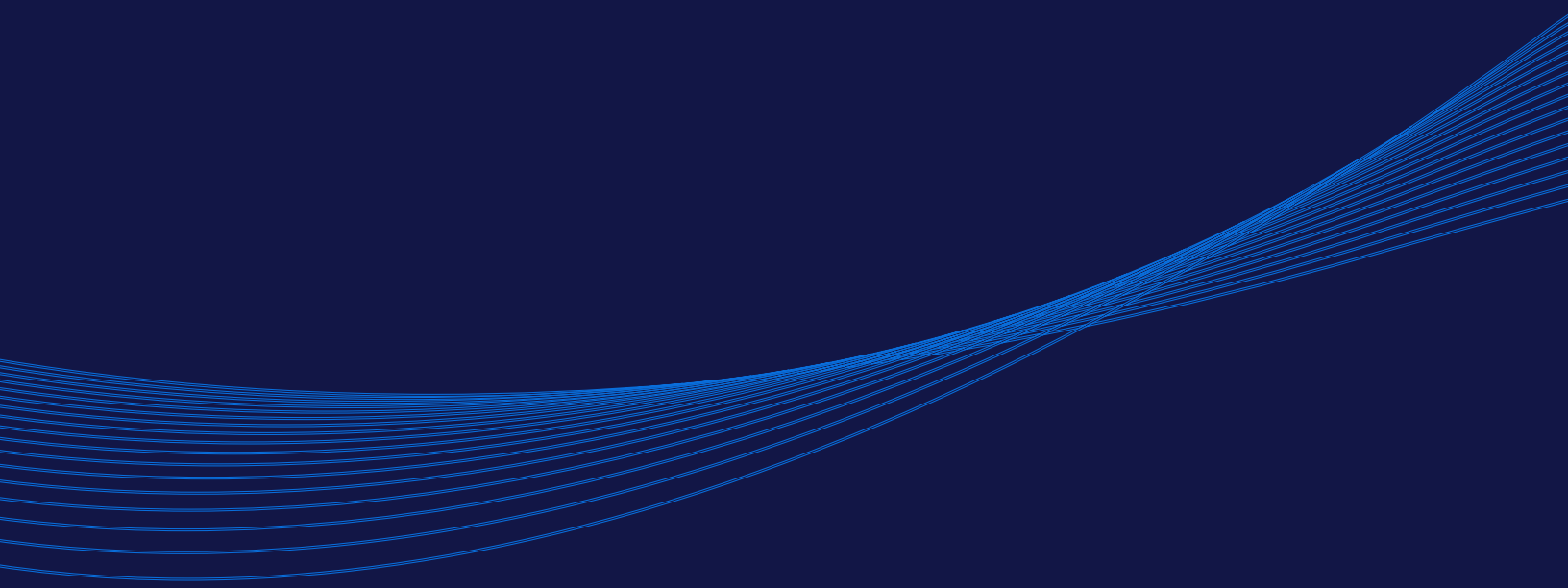
3. **Change Data Capture**
   - EDB Replication Server
   - Other tools (Oracle GG, DBConvert

	Snapshot	Snapshot in Parallel	Change Data Capture	Target
EDB Replication Server	●	●	●	Postgres Advanced Server/PostgreSQL
EDB MTK	●	●	●	Postgres Advanced Server/PostgreSQL
AWS SCT	●	●	●	PostgreSQL
Ora2Pg	●	●	●	PostgreSQL
Ora_FDW (extension)	●	●	●	PostgreSQL
Dblink_ora (extension)	●	●	●	Postgres Advanced Server

We hope this blog has provided you with information you need to begin planning an effective Oracle to Postgres migration!

Want to dive deeper?

Read our white paper “Replacing Oracle with Postgres: How to Successfully Migrate Your Legacy Databases”



About EDB

EDB provides enterprise-class software and services that enable businesses and governments to harness the full power of Postgres, the world's leading open source database. With offices worldwide, EDB serves more than 1,500 customers, including leading financial services, government, media and communications and information technology organizations. As one of the leading contributors to the vibrant and fast-growing Postgres community, EDB is committed to driving technology innovation. With deep database expertise, EDB ensures extreme high availability, reliability, security, 24x7 global support and advanced professional services, both on premises and in the cloud. This empowers enterprises to control risk, manage costs and scale efficiently. For more information, visit **www.enterprisedb.com**.

The Complete Oracle to Postgres Migration Guide:

Move and Convert Schema, Applications and Data

© Copyright EnterpriseDB Corporation 2022
EnterpriseDB Corporation
34 Crosby Drive
Suite 201
Bedford, MA 01730

EnterpriseDB and Postgres Enterprise Manager are registered trademarks of EnterpriseDB Corporation. EDB, EnterpriseDB, EDB Postgres, Postgres Enterprise Manager, and Power to Postgres are trademarks of EnterpriseDB Corporation. Oracle is a registered trademark of Oracle, Inc. Other trademarks may be trademarks of their respective owners. Postgres and the Slonik Logo are trademarks or registered trademarks of the Postgres Community Association of Canada, and used with their permission.

POWER TO POSTGRES