OpenSCG

White Paper

Oracle to PostgreSQL Migrations

Table of Contents

Introduction

Migrating from an Oracle database to PostgreSQL frequently gives organizations benefits that range from lowered costs to a better technical architecture. The OpenSCG Migration Factory speeds the process while reducing risk.

This white paper explores the process for migrating an application from an Oracle database to PostgreSQL. It is intended for people who have already made the wise choice of moving to PostgreSQL and are exploring what is involved in a migration project.

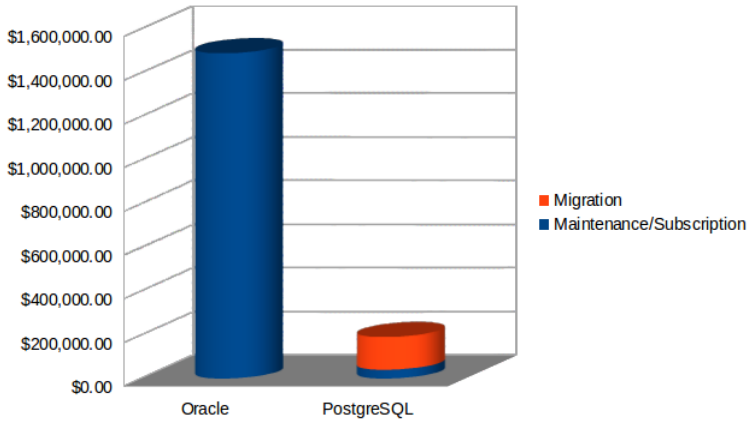
Executive Summary

Relational databases, particularly Oracle, are mission critical systems for any organization, but being locked into a particular vendor is a risky and costly situation. Expensive annual maintenance fees combined with onerous licensing restrictions are compelling reasons to consider switching your underlying database technology. Migrating to a proven open source alternative such as PostgreSQL overcomes those challenges, but an Oracle to PostgreSQL migration can be an involved undertaking requiring special skills and extensive experience. Mission critical applications can be very complex with small maintenance windows. To have a successful migration, it is imperative that your migration project has a comprehensive plan from the initial application assessment to running the new database in production.

OpenSCG's Migration Services combine the automated tools with the hands on experience to move your most important databases to PostgreSQL. Unlike in-house DBAs and developers who may do an Oracle to PostgreSQL migration once a decade or lifetime, the OpenSCG team migrations mission critical Oracle database to PostgreSQL everyday. OpenSCG's Migrations Services delivers Oracle to PostgreSQL migrations in a quick, reliable and cost effective manner.

Benefits of Migrating to PostgreSQL

The most obvious benefit of migrating to PostgreSQL is cost. Even after the Oracle licenses are purchased, the ongoing maintenance costs are significant. Oracle's list pricing is based on a per-core model with additional costs for features like partitioning and high availability. For example, the three year total cost of ownership of an existing 32 core environment with partitioning for scalability and Active Data Guard for high availability can be nearly $1,500,000.



A possibly more compelling long-term benefit of PostgreSQL is the ability to create technical solutions based on need not licensing. It is all too common for people to deploy things sub-optimally to save money on licenses instead of the right technical solution. That could mean not updating to newer hardware because the newer servers all have more cores than the previous generation. Or, it could result in many application servers all connecting to a single database server sometimes over slow WAN links. There are many cases where you need to do the wrong thing technically to optimize for licenses.

When to Migrate

All too often people look to migrate to PostgreSQL from Oracle when they are up for renewal. At that point it is too late. A migration does not happen overnight. The most time consuming and intensive part of a database migration is actually testing the application. So in fact, the ideal time to migrate your database is in conjunction with something that requires testing your application. If you are upgrading hardware, changing data centers or moving to the cloud, you need to do a full test anyway. It is quite common to see people moving from on-premise Oracle to PostgreSQL in the cloud. Combining both projects saves a significant amount of time and money.

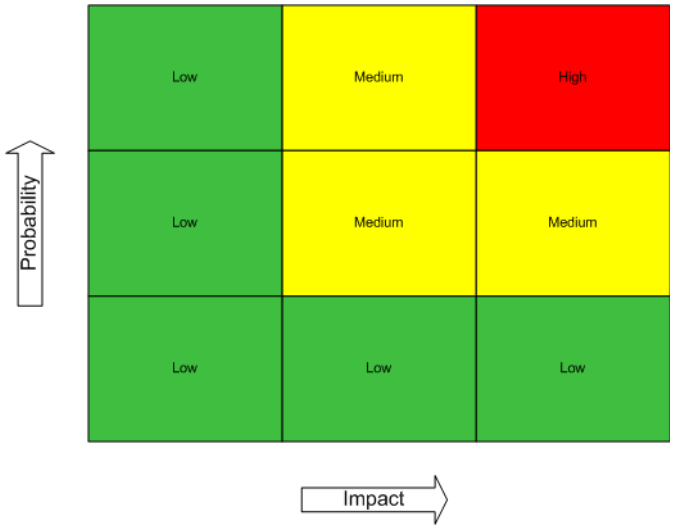
While it's ideal to time your migration with other infrastructure changes, the driver for most companies is still license renewals. Nevertheless, if you miss the opportunity to migrate to PostgreSQL before the renewal is necessary you need to

sign a new contract with Oracle, there are still benefits to migrating to PostgreSQL. Many times the freed up licenses from the migration can be reallocated to other applications in desperate need for upgraded hardware. After the migration, just apply those renewed Oracle licenses to another server that you are keeping on Oracle.

Common Database Migration Challenges and Risks

Effort

Some migrations can be a time consuming and involved project. Are there enough resources to execute the project without taking away from the necessary things to run your business?



Proficiency

A deep understanding on both Oracle and PostgreSQL is necessary to migrate an application properly. Migration projects are not done everyday by in-house DBAs and developers. Will they need help in identifying all possible dependencies before it affects the project time line?

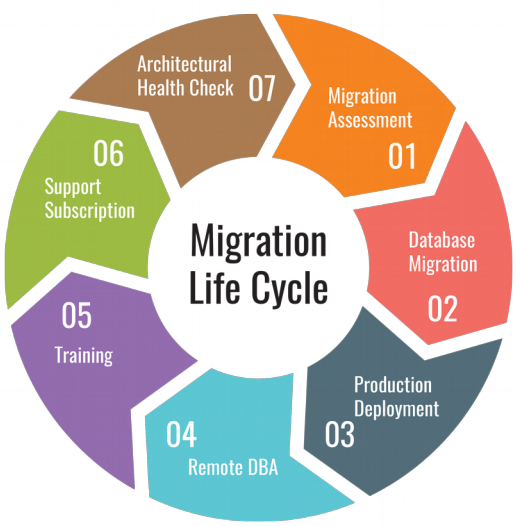
Schema Quality

Database systems can grow and evolve over time. By nature, they are constantly changing, which means they are constantly being administered. How many stale tables are left around? Are your stored procedures under source control? Is there a golden master copy of the schema?

Technology

Some migrations are for older applications where the original developers are no longer around. Do you have the proper test cases to ensure they are migrated correctly?

Migration Life Cycle



Migrating an application is more than just converting some stored procedures to a new syntax. The full life cycle spans several months to a year after you have moved to production. The behavior of a PostgreSQL database is different than Oracle and it is not until your application is running in production that you really understand the maintenance required on your new database. You can make logical assumptions based on experience, but the exact details come to life with real workloads. Things like your ideal checkpoint interval, the frequency you need to rebuild your indexes, and your optimal backup schedule as all things that are fine tuned over time.

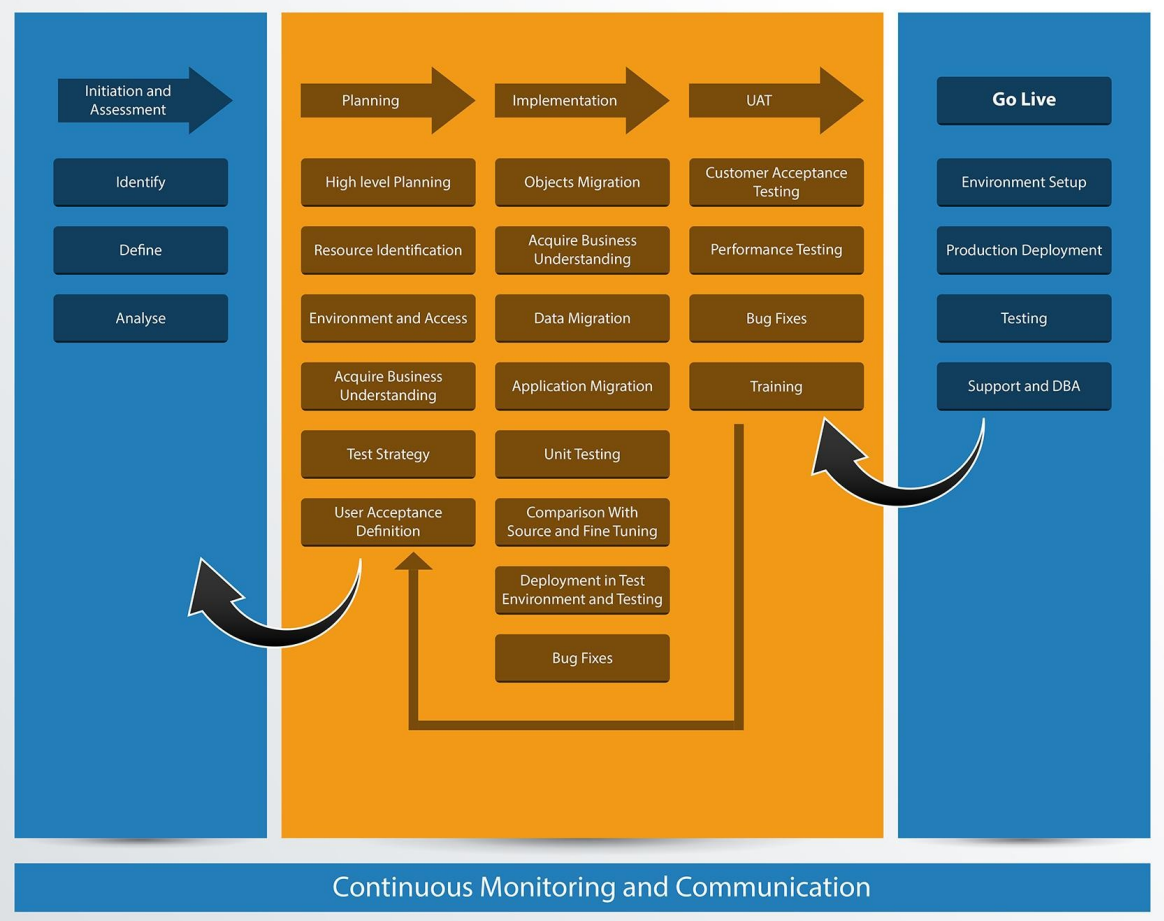
After the application is deployed to production, you will need database administrations who are experienced with running PostgreSQL in production. You may want to supplement your existing team with short-term help while your team is properly trained on PostgreSQL. Also, with Oracle, you had to ability to call for support when issues came up. You will need that ability with PostgreSQL as well. All of those pieces are available at OpenSCG to help make your migration off of Oracle successful.

Migration Service

Switching out your application's main database can be complex, but breaking the project into smaller well defined phases takes much of the risk out of the migration. The OpenSCG migration methodology has been developed over more than a decade of migrations to PostgreSQL and proven across more than 100 types of applications.

The core of the methodology is testing, testing, and more testing. By focusing the project on quality assurance, you can be assured that at each step along the way the migration is moving in the right direction and there are no major surprises at the end. The OpenSCG migration process requires unit tests on all database objects and then builds a full test framework from the ground up to make sure there are no lingering issues as the application is moved through the development cycle. Individual stored procedures are tested for functionality and performance

early in the cycle so that issues surface long before end users start using your migrated application.



Scope of Service

The OpenSCG Migration Service follows a mature and vetted process leveraging hands-on experience and a series of tools. A migration project can be accomplished using a combination of on-site and off-site resources depending on customer requirements.

Identifying Migration Candidates

A database migration starts with an assessment to identify and define which databases are candidates to move to PostgreSQL. The cost advantages of moving

to PostgreSQL are obvious, but some applications just cannot be moved. The reasons for these trapped applications falls into two categories: business and technical.

Trapped Applications (Business)

The most common reason an application is stuck on Oracle is product certification. Many third party applications certify on specific databases and versions and PostgreSQL may not be part of their blessed list. Some bet their product on Oracle while others give you options like SQL Server or MySQL. Technically, it may be possible to migrate these applications to PostgreSQL, but the first time you call your vendor for support and they find out you migrated your database to an uncertified solution, they'll push back. It is like swapping out the hard drive on your TiVo, you've just voided your warranty.

Another common business reason an application is trapped on Oracle is that the Return on Investment (ROI) for a migration project just does not work out. Oracle is expensive, but once the initial licenses are purchased, the yearly maintenance fees are a fraction of the original purchase price. If your application is happily using Standard Edition 2 on a server with only a couple of cores, it just does not make financial sense to move to PostgreSQL.

Trapped Applications (Technical)

Let's face it, Oracle is a multi-billion dollar company and they invest heavily in their flagship product. How can a rag tag bunch of open source developers create something that can do everything Oracle can do. The truth is, we can't. Most applications don't use those advanced features that are not available in PostgreSQL, but some do, and those applications are trapped.

The most common feature in this category is Oracle Real Application Clusters (RAC). PostgreSQL does not have a feature like Oracle RAC, but it does give you some of its functionality. If you are using Oracle RAC for simple High Availability and a few seconds of downtime is acceptable during a failure, PostgreSQL may be great. If you are really using RAC for its scaling features and transparent fail over, PostgreSQL may not be the right fit.

However, in reality, there are very few trapped applications because of technical reasons. If the business value justifies the migration, there is a technical solution to just about any feature in Oracle.

The solution may just involve more open source projects than PostgreSQL. Take for example, Advanced Queuing. PostgreSQL does not have message queues built

into the core database, but there are a number of open source message queues like the Apache projects ActiveMQ and Kafka. PostgreSQL even allows you to interact with those queues directly from the database using Perl or Python stored procedures. Even the RAC example that we just used has alternatives if we think outside of the database box. There are many open source in-memory caches like the Apache projects Ignite and Geode that can front-end a database and provide the high availability and scalability of Oracle RAC.

Analyzing Migration Candidates

A database migration has three main factors when determining the size of a project: the number and complexity of database objects, the SQL embedded in the application code and the size of the overall data. Combining the estimates of these three factors will result in an accurate estimate of the project scope and time-line.

Database Objects

When analyzing an Oracle database for a migration, the number of database objects is usually the dominating factor in a project. Some migrations where the only database objects are Tables and Indexes are very simple, while others may have hundreds or thousands of stored procedures. The objects you need to consider for the migration include:

- Schemas
- Tables
- Indexes
- Constraints
- Stored Procedures
- User Defined Functions
- Custom Types
- Views
- Triggers
- Packages

When analyzing the database objects, it is important to remember that the development time to translate an object from Oracle to PostgreSQL syntax is a minor percentage of the overall effort. Over the years, and many migrations, time and time again, the effort needed to translate objects from Oracle to PostgreSQL account for only 20%-30% of the project time. Take for example a view in Oracle that uses ANSI SQL. It can be created in PostgreSQL with no changes at all, but there is significantly more effort for a migration. It runs in PostgreSQL, but how do you know it is the same as in Oracle? Each object needs a set of unit test scripts to make sure the results out of the database are the same in both Oracle and PostgreSQL and if they are not, that everyone understands why. Believe it or not, people have bugs in their database code which end up being caught in the migration. Catching these bugs are only possible by using a comprehensive unit test suite across all database objects which is an added benefit of a migration project. Doing a migration without a deep testing strategy at the object level will more often than not lead to a delayed or failed project.

Application Code

Analyzing the code in the application tier for places that interact with Oracle may be the most difficult and time consuming migration task. Some application developers fully embrace Object-Relation Mapping (ORM) tools to abstract their code from the database and the code change needed are simply change the ORM dialect to PostgreSQL instead of Oracle. This is the best case scenario and the one time your DBAs will stop complaining about the SQL generated by an ORM. At the other end of the spectrum is a C-based application that uses Oracle's native OCI library where all of the database calls need to be converted to use PostgreSQL's native libpq library. In reality most applications are somewhere in the middle. Many times they are a Java or .NET application where the developers isolated the database interaction to a few classes or functions. The tricky part is finding that piece of code that does not follow that pattern.

Data Migration

Moving the data from Oracle to PostgreSQL is technically pretty simple. You can do it in any number of ways from using ETL tools, exporting the data as files or using the Oracle Foreign Data Wrapper in PostgreSQL to pull the data over. The size of the database and the size of the outage window available for the migration will determine the effort required to move the data. If you are moving 10GB of data on a local network and you can take an outage for a few hours, things are pretty simple. However, if your database is 10TB and you are migrating from on-premise hardware to the cloud with a 1 hour maintenance window, things just got

interesting. In these cases, replication is the answer. There are many tools that replicate from Oracle to PostgreSQL, both commercial and open source.

Planning a Migration

Once a migration is identified and scoped, planning a migration project is a comprehensive effort involving many people in your organization. Switching your database is a major change. Many organizations went through similar changes 10 years ago as they switched from Unix to Linux. It involves your operations team, your development team and your testing team.

Resource Identification

The key first step of any migration is to identify the project team. Even if you are using a partner like OpenSCG to do the heavy lifting of the project, there are four main roles every organization needs to provide for a successful migration.

Champion	The main voice advocating for the migration internally.
Project Manager	This person is the initial point of contact for all migration related questions. The project manager's main roles are to clear all roadblocks for the migration team as well reporting status internally.
Technical Lead	Depending on the application being migrated, the technical lead may be in one of two roles. If the application is something the technical lead knows well, the role is to share that knowledge with the migration team and then learn the new PostgreSQL environment so it can be maintained going forward. For the more common case where a legacy application is being migrated and the team that built it has long since left the company, the technical lead's main role is to regain that institutional knowledge.
Testing Lead	The testing lead is the gatekeeper for the application. This person owns the user acceptance criteria and certifies that the unit test scripts are comprehensive.

Environment and Access

When migrating to Oracle, new development and test environments will be needed at least temporarily. During the start of the project there will be a period of time where the application will be very unstable. During this time the environment where the application is being deployed will be mostly unusable. If you are stopping all other development and maintenance on the application while the migration is ongoing, you can use your existing development and testing environments. However, most people need to continue moving their applications forward so a temporary environment is needed until the migration is far enough along that you are ready to point everything including your developers at PostgreSQL instead of Oracle.

In higher security organizations, the process to get the migration team access to the needed servers should start as soon as possible. The developers and testers need access to the existing Oracle databases and the new PostgreSQL databases. You also need to start thinking about how the data will move from Oracle to PostgreSQL. If the data will be moved by any method other than exporting as files, there needs to be a pipe between the Oracle and PostgreSQL databases. That may mean firewall rule changes which in some organizations is a long process.

Knowledge Transfer

The migration team needs a good understanding of what they are migrating. In addition to a deep dive on the technology, a functional understanding is essential. This is a great opportunity for primary users of the application to demo for the team how they use the application to the team. This typically turns into a time for the end users to vent about some frustrations with some slow screens or overly difficult processes. Take notes and keep those areas in mind. A newly added index as part of the migrated project may make all the difference in the world to your end users and they will start telling everyone how great the migration to PostgreSQL is.

Test Strategy

The full testing strategy needs to be planned at the start of a migration project. With any database move, even if it is just a major upgrade of the same database, the application needs to be fully tested. Bugs can manifest in a number of places in a migration project so testing at the low level and working up the stack as parts of the application are verified allows issues to efficiently be found and fixed.

Data Testing	Simple tests such as verifying the tables definitions match and row counts are the same on Oracle and PostgreSQL are the minimum. A full data comparison really needs to do to ensure numeric precision or essential whitespace are not lost by the data loads.
Unit Testing	All database objects including procedures, packages, triggers, views, and functions need positive and negative unit test coverage.
Application Testing	The full application needs to be thoroughly tested and exercised.
Performance Testing	A performance benchmark needs to be established on the existing Oracle database to compare with the new PostgreSQL database. Note: it is common that the first time the performance test is run, PostgreSQL will be 2-5 times slower that Oracle. When the performance testing phase is over, PostgreSQL performance is typically the same as Oracle.
Operations Testing	The PostgreSQL configuration will be new so backups, high availability and other maintenance routines need to be tested.
User Acceptance	Business and other end users need to sign off on the migrated application to ensure the experience of the application meets or exceeds the old Oracle environment.

Migrating an Application

Once the planning phase is finished, the main phase of actually migrating the application to PostgreSQL starts. Most people think this is the hardest and longest part of a migration, but in reality, this phase is very mechanical and straightforward. PostgreSQL is very similar to Oracle so by following a few patterns, the actual technical migration moves quickly.

Object Migration

Some applications are fairly simple with the database objects just being some tables and indexes, while others may have thousands of stored procedures and functions. There are a number of tools that help with this particular task that

minimize the monotony of conversions from Oracle to PostgreSQL. In the open source world, there is a great tool called Ora2Pg that helps move the objects to PostgreSQL. Amazon is also getting into the database migration game with their Schema Migration Tool (SMT). The SMT is a free tool with a nice GUI that helps move the database objects from Oracle to PostgreSQL. No matter what tool you use to move your objects, the key is use a tool that gets you most of the way there. PostgreSQL is similar enough to Oracle, that this phase can be largely scripted.

Data Migration

Moving data from Oracle to PostgreSQL is a pretty simple process in a development and test environment, but the production data transfer can be much more complex. In a development environment where the data sizes are usually smaller and maintenance windows can be as large as you need to be, the data can be loaded to PostgreSQL from simple CSV files. You can even use the Oracle Foreign Data Wrapper to pull the data into PostgreSQL.

For a production data move, you may be in a situation where you need to migrate several terabytes of data with a maintenance window of one hour. The only way to accomplish this is by pre-staging the data in some way. Using a replication tool that allows Oracle to PostgreSQL migration like SymmetricDS, HVR or Amazon's Database Migration Service allows you to move the data long before the scheduled production cut-over. Thorough testing should be done on lower environments to understand the timings needed to fully synchronize the databases. Depending on the size of the database, data types in the tables and bandwidth between the servers, it may take days or weeks to do the initial load to PostgreSQL using replication. Some schedules can not accommodate that timeline, so you may need to combine several techniques to fit around business timetables and requirements.

Application Migration

In the migration phase, the code changes in the application are generally simple and straightforward. The heavy lifting of finding all of the embedded SQL was done in the previous analysis phase and changing the SQL to PostgreSQL syntax

is fairly mechanical. The most common code fix required is changing the references from SYSDATE to CURRENT_TIMESTAMP. There are several other common changes depending on the number of "Oracle'isms" used frequently in the code. The closer the developers stayed to the ANSI standard, the less code fixes are necessary.

Testing the Application

The testing phase really starts in parallel with the migration phase. As the database objects are converted, they should all have test cases associated with them. This allows the testing team to validate the individual objects are functioning properly in PostgreSQL before the full conversion is complete.

Unit Testing

Unit testing starts with the database developers creating the initial set of test cases for each object that is migrated. Depending on the object, there may be many test cases needed to fully test all lines of code. For example, a stored procedure with 3 parameters will have at least 9 test cases to account for different values of the parameters including "null" values. These test cases need to run against the existing Oracle database to produce baseline results which includes the results of exercising the object as well as the performance timing of running the individual test. Once an object is migrated to PostgreSQL, the unit tests need to be run to ensure the results are the same.

This test driven migration methodology focusing on unit tests has uncovered many subtle and difficult to find issues if application testing was the only way to test. Issues such as different row ordering or "null" handling may not be noticed in application testing except in certain critical corner cases where root cause analysis may be hard to pin down. By focusing on the testing of individual objects, issues are found quickly reducing the overall risk of migration project.

Application Testing

A database migration is major surgery on an application so a full testing cycle needs to be conducted on the application. Every test case that your testing team

runs through for a major release, just needs to be run. This should include any automated as well as manual testing.

Performance Testing

An initial performance test should be run pretty early in the testing cycle to start highlighting how your application is using the database and how PostgreSQL handles that type of workload. It is fairly common that the first time a performance test is run against PostgreSQL, it will be 2 to 10 times slower than Oracle, but do not worry. After a few rounds of performance testing and some tuning, PostgreSQL will perform about the same as Oracle and maybe a little faster.

Performance testing should start with timing the unit test case and progressively growing in complexity to match as close to a real application load as possible. After each round of testing, the bottlenecks need to be identified and passed back to either development or operations. Development will need to create indexes, rewrite queries, change data types or some other type of improvement. Operations may need to modify the PostgreSQL configuration in some way.

Production Deployment

As the migration project starts to wind down from a development and testing perspective, focus needs to shift toward rolling out to production. This may require a fair amount of coordination across teams, partners and even customers.

Environment Setup

The amount of effort needed to setup a production environment varies greatly depending on where the database is being deployed. On premise requires setting up hardware, operating systems and more. The cloud simplifies the setup and if you use a PostgreSQL cloud service like Amazon RDS, even high availability is handled.

Where possible, a new environment should be setup so the old Oracle environment can stay intact until the application has been fully vetted. In a cloud or virtualized environment this can be as simple as starting up new instances, but in a physical world, this can be more complex. If you decided to time your

migration with upgrading your hardware, you may have a duplicate environment, but if the only thing changing is the database, it needs to be coordinated carefully.

Data Migration

If you have planned and tested your data migration in the previous stages, the final data move will be a very predictable and straightforward process. In the testing phases, the data migration would be run multiple times so the running times will be know long before you try to move the data for the last time.

Production Cut-over

The final steps for going live with the new production application running on PostgreSQL may be as simple as pointing the DNS entries or load-balancer at the new production environment. It may be a bit more complicated if you are keeping most of your existing environment, but things like switching the connection strings would have been thoroughly tested in lower environments.

Training

Just before you move your newly migrated application to production, use idle time for the team to be trained on how to administer PostgreSQL in a production environment. Your operations team should at a minimum be skilled at:

- Backup and Recovery
- Performance Tuning
- Regular Maintenance
- Monitoring

Conclusion

This white paper covers the process of migrating from Oracle to PostgreSQL. Depending on your business needs, corporate culture and your migration strategy, you likely will use a combination of methods and processes to migrate your database. For a successful project, it is critical to plan appropriately and test extensively.

About OpenSCG

OpenSCG is the leading provider of Oracle to PostgreSQL migration services. We have been focused on heterogeneous database environments (PostgreSQL, SQL Server, Oracle, Sybase, Cassandra) since starting in 2010. Today, with over 120 employees around the world, we help our customers get the most out of the Open Source database.

We provide services covering all major steps of a typical migration project: assessment, schema conversion, data migration, application conversion, testing, integration, deployment, performance tuning, training, remote DBA, and support.

For more details, visit us at http://www.openscg.com, e-mail us at info@openscg.com, or call +1-732-339-3419.