



Best practices for migrating from Oracle to Amazon Aurora

Database Modernization Week

Mark Mulligan and Nelly Susanto

Table of contents

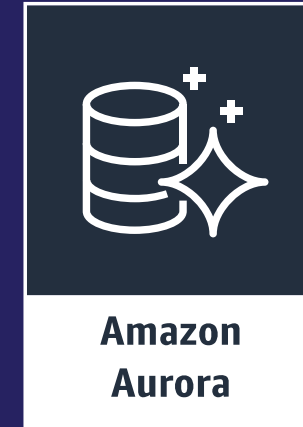
- Oracle to Aurora migration process overview
- Schema Conversion Tool
 - SCT Overview
 - Demonstration of Schema Conversion Tool (SCT)
 - SCT best practices
- Database Migration Service
 - DMS Overview
 - Demonstration of Database Migration Service (DMS)
 - DMS best practices
- How AWS can help
- Navisite, AWS Premier Partner migration case study
- Q&A



Oracle to Amazon Aurora Migration



Why customers are migrating?



Expensive



**Restrictive
licensing
terms**



**Reduce
cost**



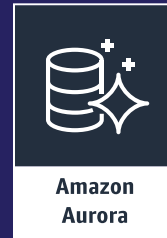
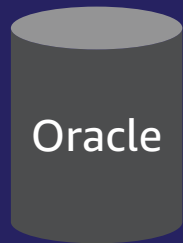
**Increase
agility**



**Innovate
faster**

Migrating Oracle to Aurora PostgreSQL

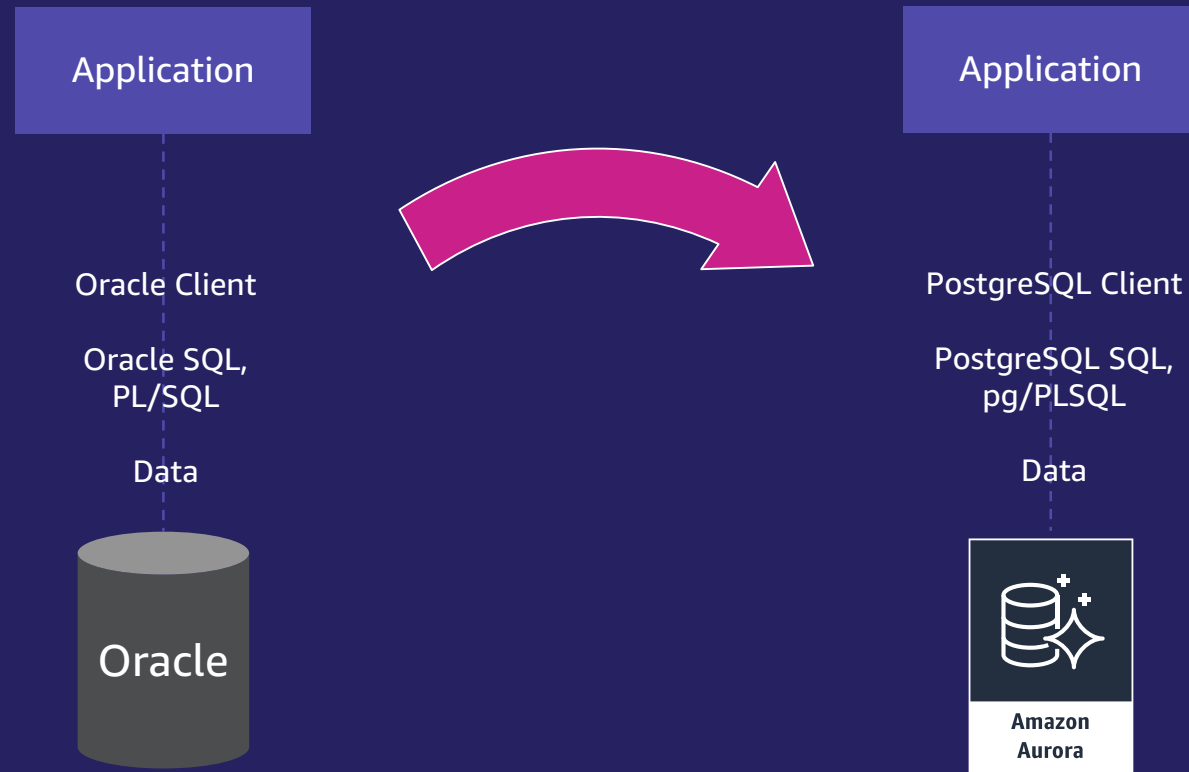
What does Oracle to *Aurora* PostgreSQL look like?



- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service

Migrating Oracle to Aurora PostgreSQL

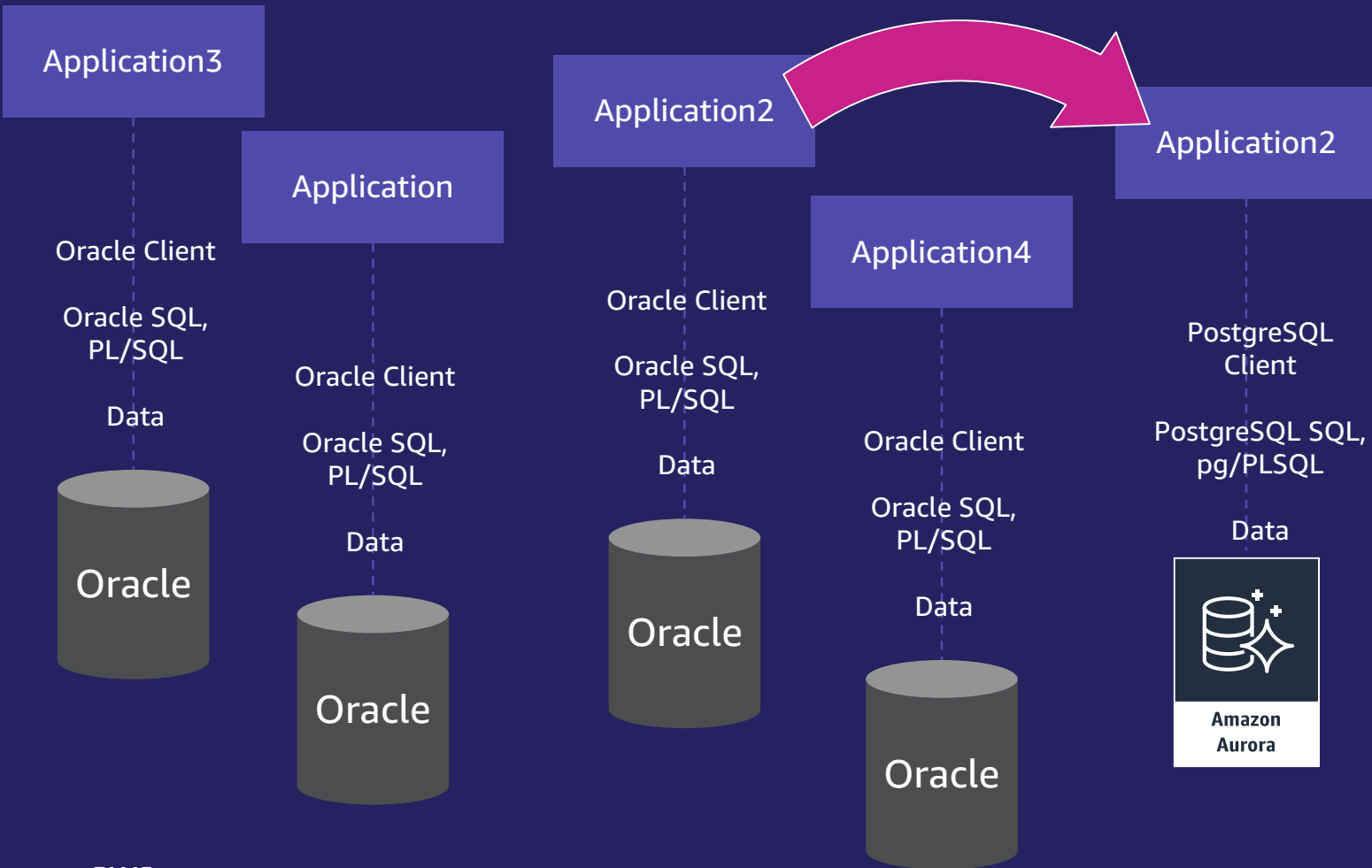
What does Oracle to Aurora PostgreSQL really look like?



- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc.) from Oracle to PostgreSQL
- Convert Database code-objects (functions, triggers, etc.) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to PostgreSQL (ANSI) SQL
- Migrate Data from Oracle to PostgreSQL
- Replace Oracle Client with PostgreSQL Client
- Test Application
- Cut over Production

Migrating Oracle to Aurora PostgreSQL

What does Oracle to Aurora PostgreSQL really look like?



- Assess Application+Database pairs for migration complexity and classification
- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc.) from Oracle to PostgreSQL
- Convert Database code-objects (functions, triggers, etc.) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to PostgreSQL (ANSI) SQL
- Migrate Data from Oracle to PostgreSQL
- Replace Oracle Client with PostgreSQL Client
- Test Application
- Cut over Production



AWS Tools to help Oracle to Aurora Migration



AWS Schema
Conversion Tool



AWS Database
Migration
Service

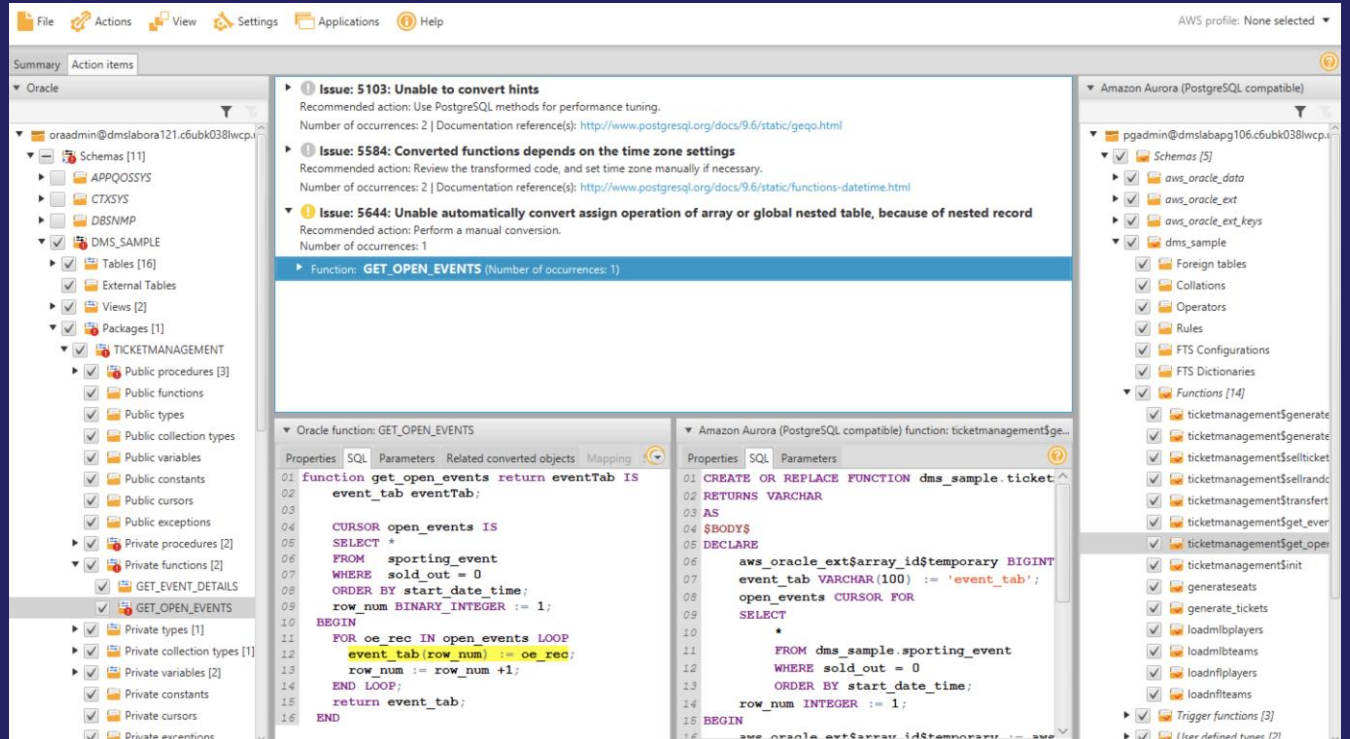
- Assess Application+Database pairs for migration complexity and classification
- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc.) from Oracle to PostgreSQL
- Convert Database code-objects (functions, triggers, etc.) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to PostgreSQL (ANSI) SQL
- Migrate Data from Oracle to PostgreSQL
- Replace Oracle Client with PostgreSQL Client
- Test Application
- Cut over Production

AWS Schema Conversion Tool



Schema
Conversion
Tool

Makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database



Features

- Database Migration Assessment report for choosing the right target engine
- Automatic conversion for eligible database objects and code
- Code browser to highlight places where manual edits are required

AWS Schema Conversion Tool (SCT) Demo

- AWS SCT
- Assessment
- Database object conversion
- Application code conversion

Nelly Susanto – AWS Database Migration Specialist



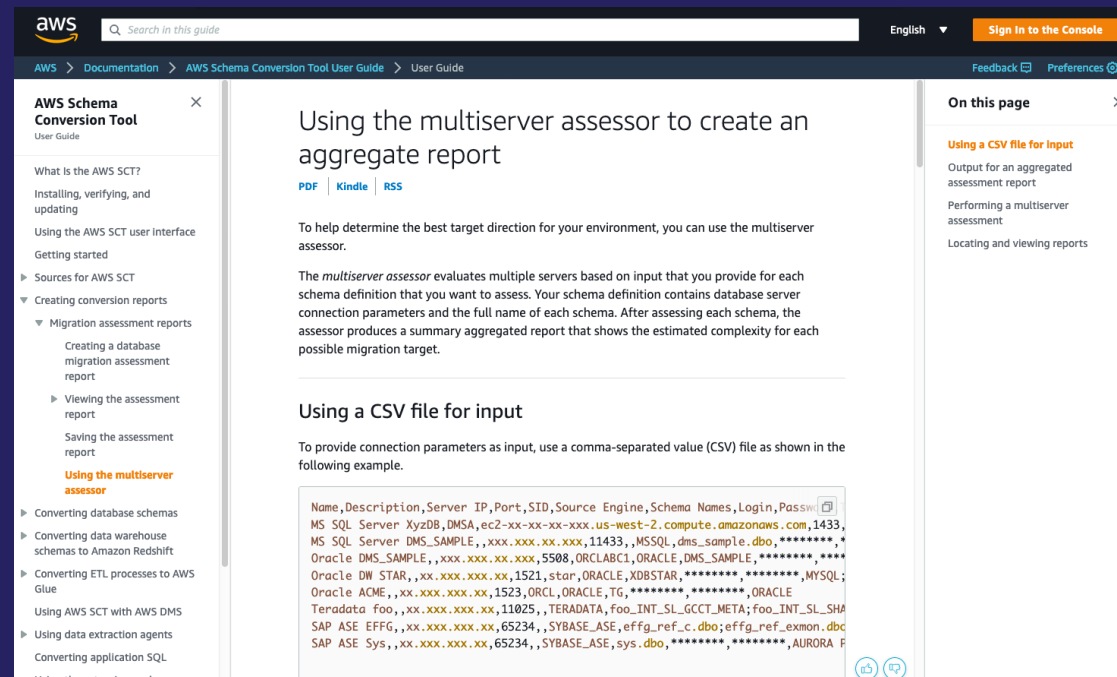
SCT best practices

- Assessment Phase



SCT best practices - Assessment

- Use the SCT Multi-server Assessment feature
- This makes it easier to run assessments against multiple databases and schemas
- https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_AssessmentReport.Multiserver.html



SCT best practices - Assessment

- Be sure to save the CSV files when you run an assessment
- The CSV data can be used to create custom reports and used with your classification algorithms
- https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_AssessmentReport.Save.html
- Useful blog about classifying database workloads:
- <https://aws.amazon.com/blogs/database/categorizing-and-prioritizing-a-large-scale-move-to-an-open-source-database/>



SCT best practices

- Conversion Phase



SCT best practices - Conversion

- Don't treat the target like the source. Understand your differences.
- Some basic examples:

Hint: Just getting started with PostgreSQL? Check out the "Introduction to PostgreSQL" chapter in the AWS PostgreSQL Immersion Day: <https://rdspg.workshop.aws/>

```
Log | 1: pg_tables [1] x
+-----+-----+-----+
| schemaname | tablename | tableowner |
+-----+-----+-----+
| 1 public    | imdbname_basic | master |
+-----+-----+-----+
```

PostgreSQL is a lowercase data dictionary

```
test=# SELECT CURRENT_DATE;

 current_date
-----
2017-05-09
(1 row)
```

No DUAL table needed

```
Examples

Set the schema search path:

SET search_path TO my_sc
```

search_path replaces PUBLIC SYNONYM

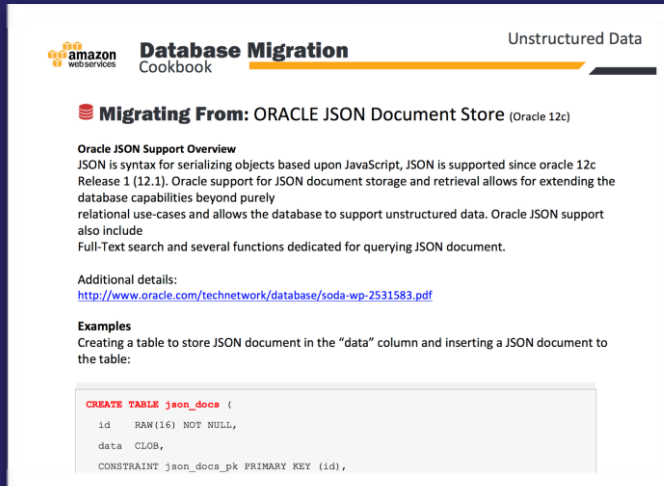
```
test=# SELECT COALESCE(fname, '') || ' ' ||
COALESCE(mname, '') || ' ' || COALESCE(lname, '') FROM users;
?column?
-----
George Washington
John Adams
Thomas Jefferson
James Madison
James Monroe
Andrew Jackson
Martin Van Buren
John Tyler
John Quincy Adams
William Henry Harrison
(10 rows)
```

Oracle concatenates nulls strings differently than PostgreSQL

The screenshot shows the AWS documentation page for "INTRODUCTION POSTGRESQL: DATABASES AND SCHEMAS". The left sidebar lists prerequisites and topics. The main content area includes a warning message and a step to explore databases. Below the text is a screenshot of the pgAdmin interface, showing the "Servers" tree with "rds-pg-labs" selected, and a "Server sessions" chart.

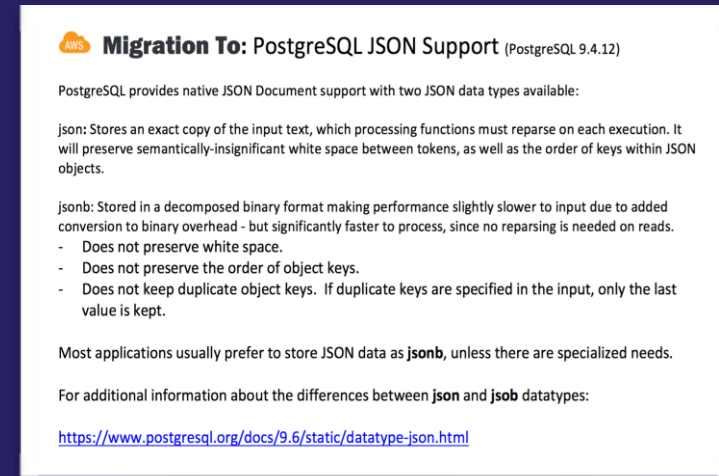
SCT best practices - Conversion

- Consult the [Oracle to Aurora PostgreSQL Migration Playbook](#) (400+ pages)
- <https://aws.amazon.com/dms/resources/>



The screenshot shows the 'Database Migration Cookbook' page for 'Unstructured Data'. The main heading is 'Migrating From: ORACLE JSON Document Store (Oracle 12c)'. It includes an 'Oracle JSON Support Overview' section, a link to 'Additional details' (http://www.oracle.com/technetwork/database/soda-wp-2531583.pdf), and an 'Examples' section with a SQL code block for creating a table named 'json_docs'.

```
CREATE TABLE json_docs (  
  id RAW(16) NOT NULL,  
  data CLOB,  
  CONSTRAINT json_docs_pk PRIMARY KEY (id),
```



The screenshot shows the 'Migration To: PostgreSQL JSON Support (PostgreSQL 9.4.12)' blog post. It explains that PostgreSQL provides native JSON Document support with two JSON data types: 'json' and 'jsonb'. The 'json' type stores an exact copy of the input text, while 'jsonb' is stored in a decomposed binary format. A list of characteristics for 'jsonb' is provided, and a link to the PostgreSQL documentation is included.

- Does not preserve white space.
- Does not preserve the order of object keys.
- Does not keep duplicate object keys. If duplicate keys are specified in the input, only the last value is kept.

Most applications usually prefer to store JSON data as **jsonb**, unless there are specialized needs.

For additional information about the differences between **json** and **jsonb** datatypes:

<https://www.postgresql.org/docs/9.6/static/datatype-json.html>

- Check the AWS Database Blog <https://aws.amazon.com/blogs/database/> for additional topics.
- You can narrow down with the [Amazon Aurora](#) and [RDS For PostgreSQL](#) tags

SCT best practices - General

- Use the latest version of SCT
 - It is updated often and each version enhances conversion capabilities
 - https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_ReleaseNotes.html
- If needed, adjust the SCT memory settings higher
 - SCT builds an in-memory model of the database objects. More memory equals better performance
 - https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_BestPractices.html
- If needed, leverage the SCT log file
 - If you have any issues with SCT (such as seeming to hang), check the log file to see the source object that it is trying to convert
 - <https://aws.amazon.com/blogs/database/configuring-the-aws-schema-conversion-tool/>





Best practices for migrating from Oracle to Amazon Aurora

Database Modernization Week

Part 2 - DMS

Mark Mulligan and Nelly Susanto

Table of contents

- Oracle to Aurora migration process overview
- Schema Conversion Tool
 - SCT Overview
 - Demonstration of Schema Conversion Tool (SCT)
 - SCT best practices
- Database Migration Service
 - DMS Overview
 - Demonstration of Database Migration Service (DMS)
 - DMS best practices
- How AWS can help
- Navisite, AWS Premier Partner migration case study
- Q&A



AWS Database Migration Service (DMS)



AWS Database Migration Service



AWS Database
Migration
Service

- Start your first migration in *10 minutes or less*
- Keep your *apps running* during the migration
- *Replicate* from within, to, or from AWS
- Move data to the same or *different database engine*

Sources*
Oracle
SQL Server
Azure SQL Server
PostgreSQL
MySQL
SAP ASE
MongoDB
Amazon S3
IBM DB2 (LUW)
Amazon DocumentDB

Targets*
Oracle
SQL Server
PostgreSQL
MySQL
SAP ASE
Amazon Redshift
Amazon S3
Amazon DynamoDB
Amazon Kinesis
Amazon OpenSearch
Amazon DocumentDB
Amazon Neptune
Apache Kafka
Redis

Consult DMS Documentation for latest DMS sources and targets

AWS Database Migration Service (DMS) Demo

- AWS Database Migration Service
- Data migration
- Data validation

Nelly Susanto – AWS Database Migration Specialist



DMS best practices



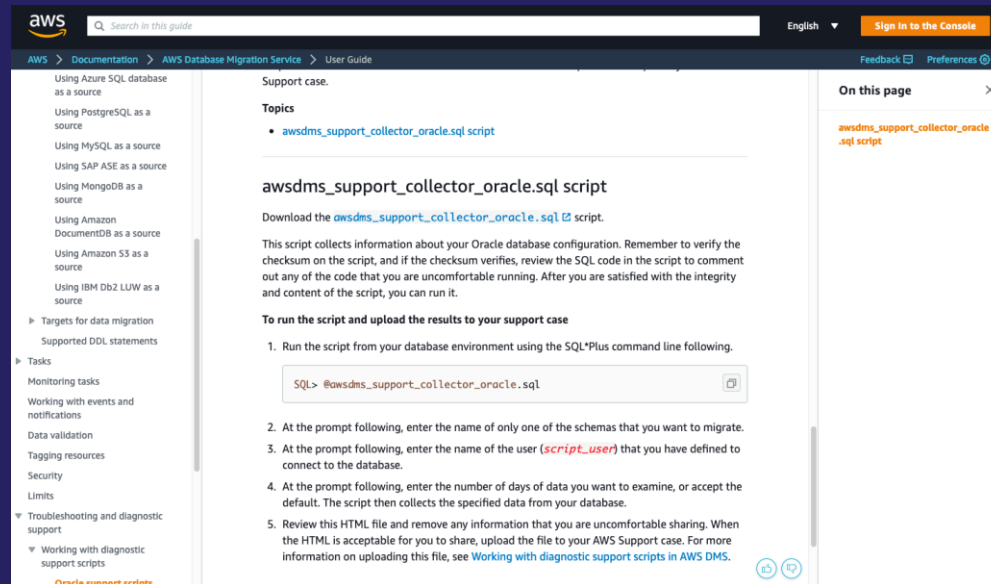
DMS best practices – Oracle as a Source

Start with the Oracle as a Source chapter in the DMS documentation. It is important. It is updated regularly.

https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html

There is a very good support SQL*Plus script you can use as a pre-check:

https://docs.aws.amazon.com/dms/latest/userguide/CHAP_SupportScripts.Oracle.html



The screenshot shows the AWS documentation page for the Oracle support script. The page title is "awsdms_support_collector_oracle.sql script". The main content area contains the following text:

Download the [awsdms_support_collector_oracle.sql](#) script.

This script collects information about your Oracle database configuration. Remember to verify the checksum on the script, and if the checksum verifies, review the SQL code in the script to comment out any of the code that you are uncomfortable running. After you are satisfied with the integrity and content of the script, you can run it.

To run the script and upload the results to your support case

1. Run the script from your database environment using the SQL*Plus command line following.

```
SQL> @awsdms_support_collector_oracle.sql
```

2. At the prompt following, enter the name of only one of the schemas that you want to migrate.
3. At the prompt following, enter the name of the user (*script_user*) that you have defined to connect to the database.
4. At the prompt following, enter the number of days of data you want to examine, or accept the default. The script then collects the specified data from your database.
5. Review this HTML file and remove any information that you are uncomfortable sharing. When the HTML is acceptable for you to share, upload the file to your AWS Support case. For more information on uploading this file, see [Working with diagnostic support scripts in AWS DMS](#).

The page also features a left-hand navigation menu with various topics, a search bar at the top, and a right-hand sidebar with a "On this page" section.

DMS best practices – Handling Oracle LOBS

- What LOB columns do you have?
- What is the biggest LOB size for each LOB column?
- Do any of the tables with LOBs not have PKeys?
- Consider using [per table LOB settings](#) in DMS task

✓ Need to plan migrations for **tables that have no PKs and contain LOBs**. Here is a query to identify those tables:

```
SELECT owner,table_name FROM dba_tables where owner='schema_name' and table_name NOT IN (SELECT table_name FROM dba_constraints WHERE constraint_type='P' and owner='schema_name ') and table_name in (select DISTINCT table_name from dba_tab_cols where data_Type IN ('CLOB', 'LOB', 'BLOB') and owner ='schema_name ');
```

✓ Find **the max LOB size** using Oracle system tables:

```
select 'select (max(length(' || COLUMN_NAME || '))/(1024)) as "Size in KB" from ' || owner || '.' || TABLE_NAME ||';' "maxlobsizeqry" from dba_tab_cols where owner= 'schema_name' and data_type in ('CLOB','BLOB','LOB');
```



DMS best practices – Table Mappings JSON

- Understand the richness of the Table Mappings JSON
- https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.CustomizingTasks.TableMapping.html
- You can specify filters
- You can adjust for the UPPERCASE (Oracle) vs lowercase (PostgreSQL) data dictionary differences
- You can use different settings (LOB, parallel) by table
- You can replicate big tables in parallel chunks
- You can order big tables to load first

DMS best practices – Plan time for setup and dry runs

- When migrating databases to the cloud, almost every customer's pre-cloud database configuration is unique
- Allow time in your schedule to work through any site-specific source configuration setup
- Also, allow time for dry runs to make sure you don't have surprises for the production cutover
- Reach out to AWS Support if you run into technical issues



DMS best practices – Understand the scope

Understand the scope of DMS. For instance,

- It doesn't replicate stored procedures
- It doesn't replicate sequence values
- It doesn't enable/disable foreign keys or triggers for you
- Don't migrate objects and data not being used

Helpful blog showing a realistic workflow:

<https://aws.amazon.com/blogs/database/how-to-migrate-your-oracle-database-to-postgresql/>

1. Create your schema in the target database.
2. Drop foreign keys and secondary indexes on the target database, and disable triggers.
3. Set up a DMS task to replicate your data – full load and change data capture (CDC).
4. Stop the task when the full load phase is complete, and recreate foreign keys and secondary indexes.
5. Enable the DMS task.
6. Migrate tools and software, and enable triggers.



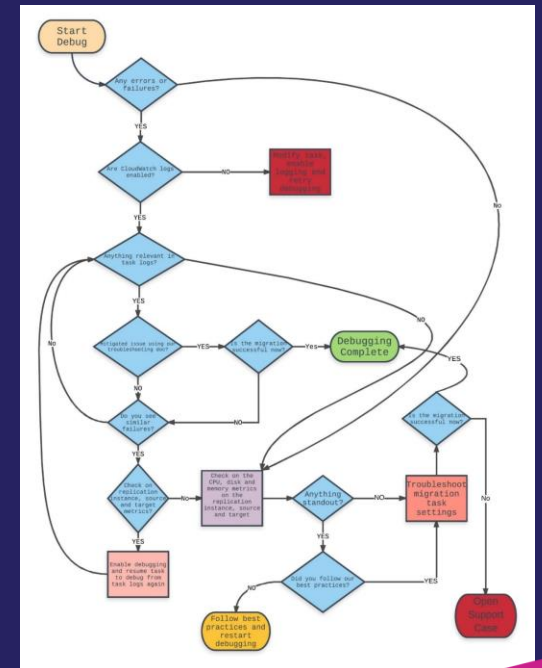
DMS best practice - General

Good blog for dealing with troubleshooting:

<https://aws.amazon.com/blogs/database/debugging-your-aws-dms-migrations-what-to-do-when-things-go-wrong-part-1/>

As we traverse the flow chart through all the decision boxes, we explore what the importance of each step is in this troubleshooting process. We talk about a few tips and tricks that we have picked up along the way while helping debug thousands of AWS DMS migrations. As stated earlier, migrations are complex and require some configuration tuning and testing based on a number of factors to be successful. As you determine the best possible configuration parameters for your migration using AWS DMS, here are a few factors to consider:

1. Infrastructural issues on the AWS DMS replication instance or source database or target database instances
2. Network issues between the source and replication instance or between the replication instance and the target
3. Data-related issues on the sources
4. AWS DMS limitations (you can find specific limitations for each of our [sources](#) and [targets](#))



How AWS can help



AWS Database Freedom

Programs



Database Freedom **reduces the risk and cost** of migrations via technical workshops, POCs, Pilots, and trained Partners

Experts



Extend your talent with AWS Solutions Architects, Professional Services, System Integrators, and Training & Certification for your teams

Speed your migration by leveraging **proven practices and guidance**

Innovation



Innovative migration tools such as AWS Database Migration Service (**DMS**) and Schema Conversion Tool (**SCT**), with high automation to **reduce manual effort**

Amazon Database Migration Accelerator

Fixed-price, risk-mitigated way to convert legacy databases



AWS Conversion experts

Rely on AWS experts who have experience migrating countless workloads



Fixed, competitive price

Know exactly how much it will cost to migrate



Re-factored database & application

Leverage modern feature-rich databases



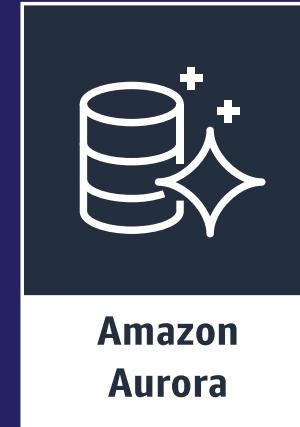
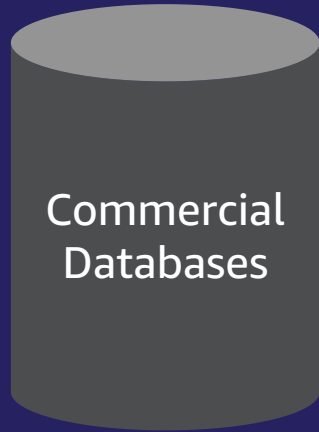
Speed

Reduce conversion time and time to value

Recap



Why customers are migrating?



Expensive



**Restrictive
licensing
terms**



**Reduce
cost**



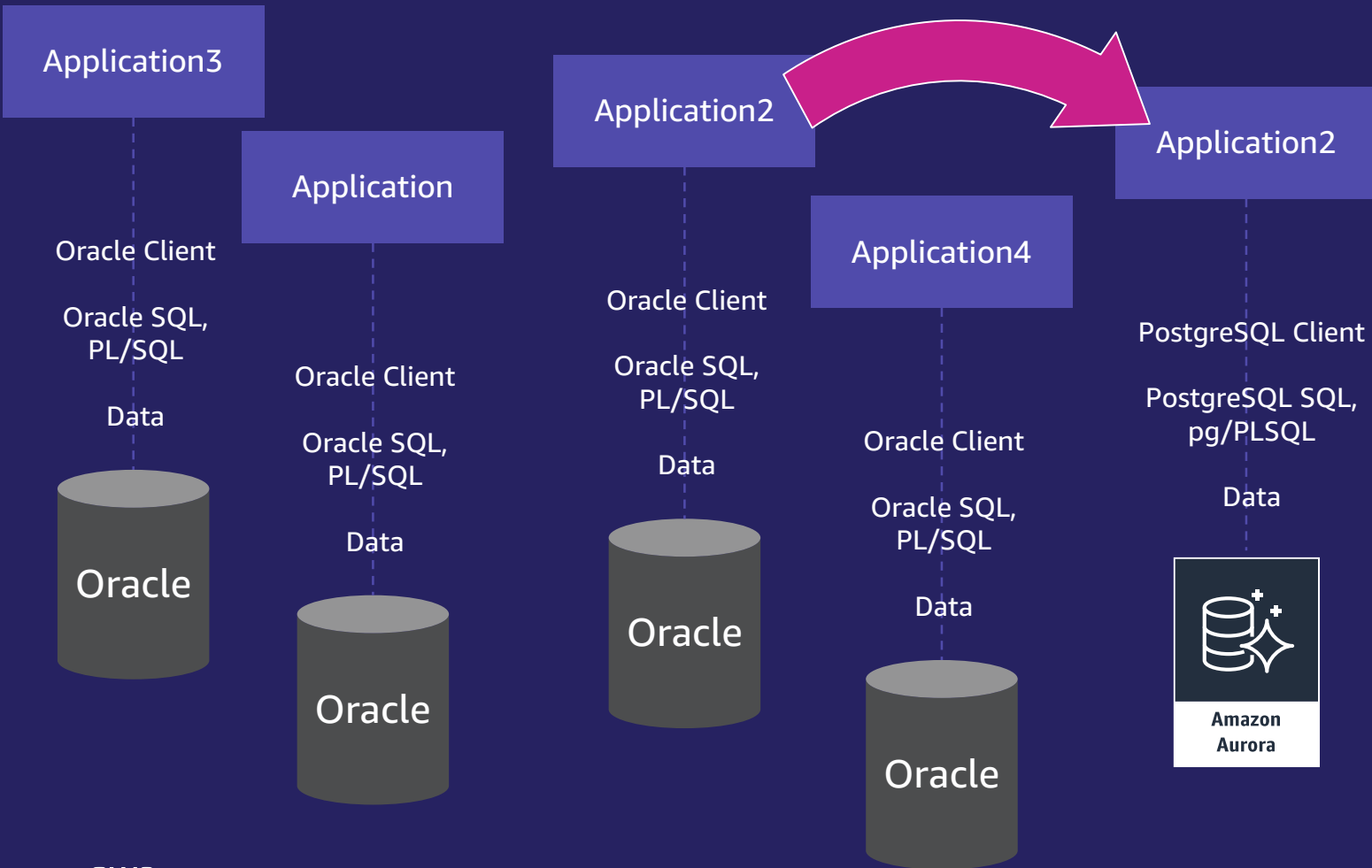
**Increase
agility**



**Innovate
faster**

Migrating Oracle to Aurora PostgreSQL

- What does Oracle to Aurora PostgreSQL really look like?



- Assess Application+Database pairs for migration complexity and classification
- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc.) from Oracle to PostgreSQL
- Convert Database code-objects (functions, triggers, etc.) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to PostgreSQL (ANSI) SQL
- Migrate Data from Oracle to PostgreSQL
- Replace Oracle Client with PostgreSQL Client
- Test Application
- Cut over Production



AWS Tools to help Oracle to Aurora Migration



Schema
Conversion Tool



AWS Database
Migration
Service

- Assess Application+Database pairs for migration complexity and classification
- Switch Database engine from Oracle to PostgreSQL
- Evolve from customer-managed to managed database service
- Convert Database Schema (tables, datatypes, etc.) from Oracle to PostgreSQL
- Convert Database code-objects (functions, triggers, etc.) from PL/SQL to pg/PLSQL
- Modify Application SQL from Oracle SQL to PostgreSQL (ANSI) SQL
- Migrate Data from Oracle to PostgreSQL
- Replace Oracle Client with PostgreSQL Client
- Test Application
- Cut over Production

AWS Database Freedom

Programs



Database Freedom **reduces the risk and cost** of migrations via technical workshops, POCs, Pilots, and trained Partners

Experts



Extend your talent with AWS Solutions Architects, Professional Services, System Integrators, and Training & Certification for your teams

Speed your migration by leveraging **proven practices and guidance**

Innovation



Innovative migration tools such as AWS Database Migration Service (**DMS**) and Schema Conversion Tool (**SCT**), with high automation to **reduce manual effort**



Thank you!