CS-524 Intro to Cloud Computing

Final Project

**Migrating your Oracle database to Amazon Aurora**

Jichen Dai

10448922

**Contents**

|  |  |  |
| --- | --- | --- |
| **Section** | **Heading** | **Page Number** |
| **1. Introduction** | 1.1 Motivation  1.2 How AWS DMS works | 3  3 |
| **2. Implement** | 2.1 Task overview  2.2 Use CloudFormation to initiate a RDS instance  2.3 Install SQL Tools and AWS SCT  2.4 Test Connectivity to the Oracle DB Instance  2.5 Test the Connectivity to the Aurora Instance  2.6 Convert the Oracle Schema to Aurora MySQL  2.7 Create DMS Replication Instance  2.8 Create DMS Source and Target Endpoints  2.9 Create and Run Your DMS Migration Task  2.10 Verify Migration Completed Successfully  2.11 Delete Resources | 5  5  9  13  15  16  22  24  27  10  32 |
| **3. Observation**  **and**  **Conclusion** | 3.1 Observation  3.2 Conclusion | 33  34 |
| **Reference** | Reference | 35 |

**1. Introduction**

**1.1 Motivation**

Nowadays there are numerous kinds of database system such as MongoDB, MySQL, Oracle, etc. Originally, if we want to migrate from one database to another base, we will have to perform capacity analysis, procure hardware and software, install and administer systems, and test and debug the installation. Additionally, if we want to migrate from one kind of database to another kind of database, we will have to rewrite our project. These work is lengthy and costs extra costs.

Therefore, Amazon provides us a service to accomplish these automatically. It is called AWS Database Migration Service(AWS DMS). It is a cloud service for you to migrate your database or data warehouse from your PC to Cloud, or from one kind of database to another kin of database. It supports many kinds of data stores such as relational databases, data warehouse, NoSQL databases, etc. With AWS DMS we can accomplish database migration within a few minutes.

**1.2 How AWS DMS works**.

AWS DMS is a server whose obligation is to migrate your source database to a target database. First, user should tell the AWS DMS where is the source database and where is the target database. Then the user set a task on AWS DMS Server to let this server finish migration automatically.

The workflow can be seperated into four step:

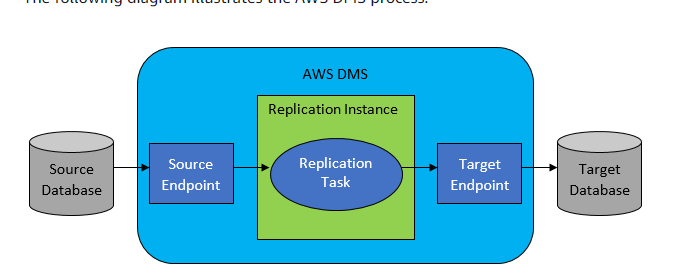
a. Identify source database and target database. They can be stored at either Cloud or PC or other machine allowed.

b. Inside the AWS DMS server, configure endpoint for both source and target database. This allows DMS Server to communicate with two machines.

c. Configure a EC2 instance to finish replicate. This instance will be configured automatically.

d. Create a replication task which specifies tables to be migrated and rules that should be obeyed. This task will be managed by AWS DMS, and information about migration process will be provided to users.

*Below is a diagram cited from AWS:*



**2. Implement Migration**

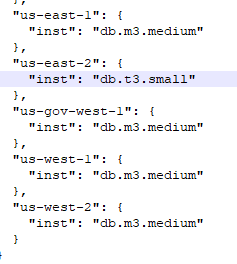
**2.1 Task Overview**

In this chapter, I will migrate a Oracle database to Amazon Aurora. Both the Oracle database and Aurora is resided on the cloud. The Oracle database is created on the RDS instance and I will use CloudFormation to help me create this database instance. When translate from Oracle to Aurora MySQL, I will use SCT for translation. After the whole experiment is finished, all instance will be deleted to prevent additional cost.

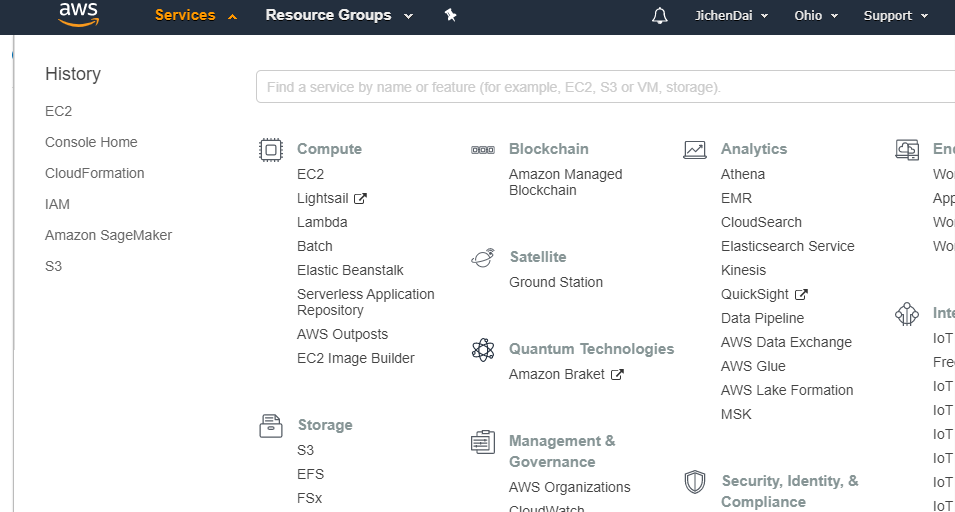
**2.2 Use CloudFront to initiate a RDS isntance**

1. Download the CloudFormation template on: <http://docs.aws.amazon.com/dms/latest/sbs/samples/dms-sbs-RDSOracle2Aurora.zip>

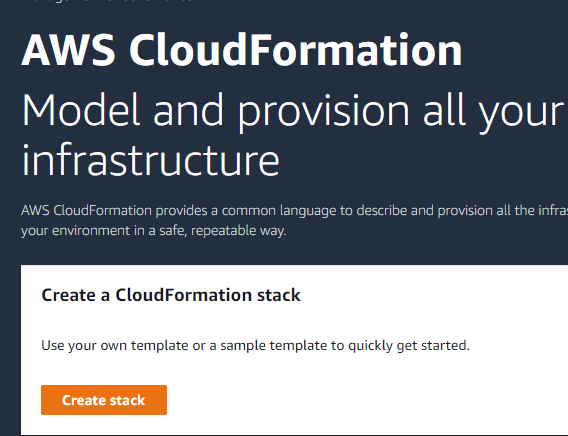
Change Configure in the template to satisfy our needs: add item about us-east-2, and change the instance type from db.m3.medium to db.t3.small.



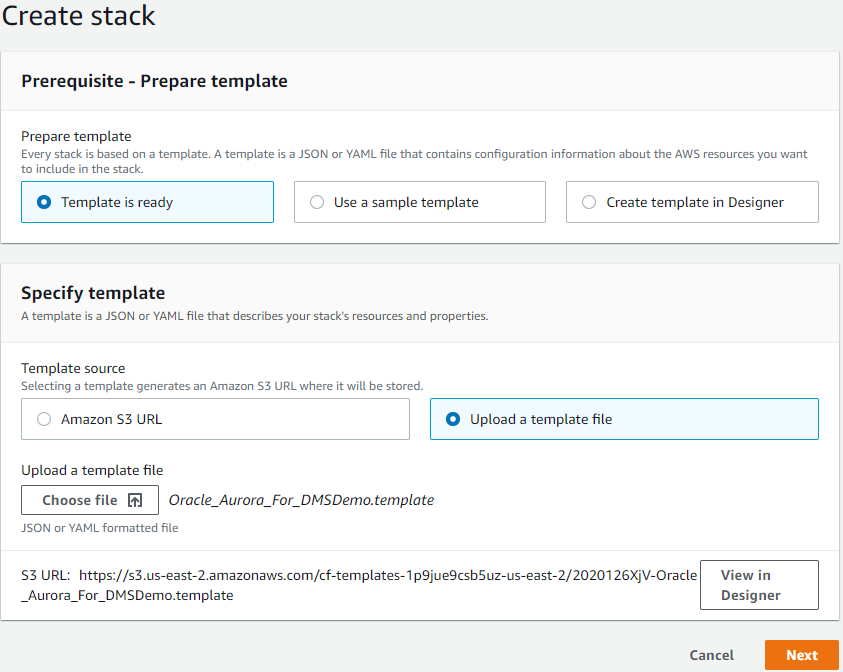
2. Sign in to AWS Management Console:



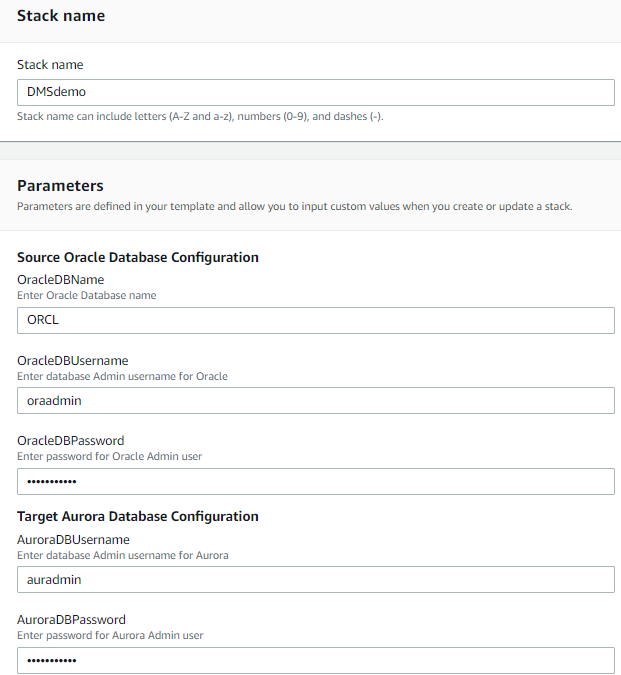
3. Go to CloudFormation and choose Create Stack.



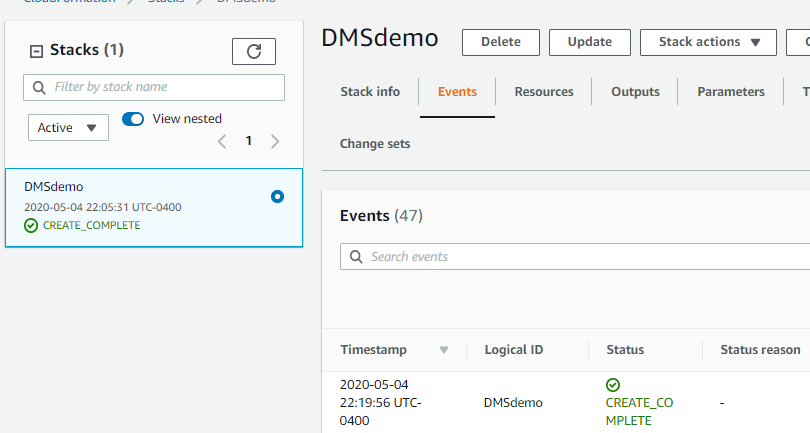
4. Upload the file you just downloaded,



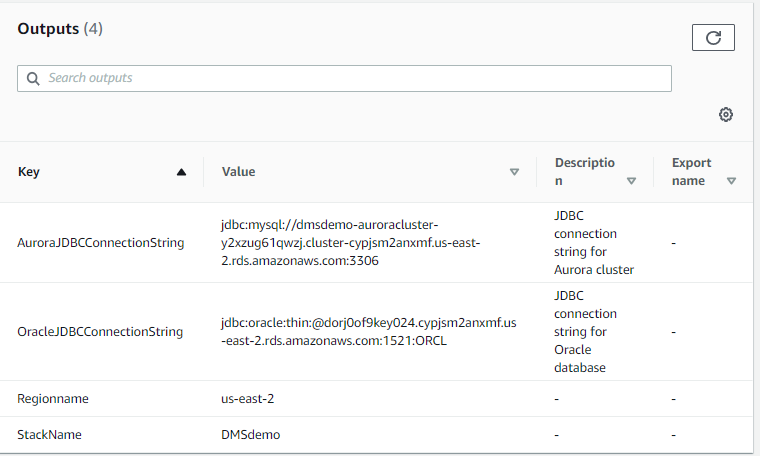
5. Choose next, configure the stack as shown below:



6. Choose next twice,click Create Stack:Now we got a new stack

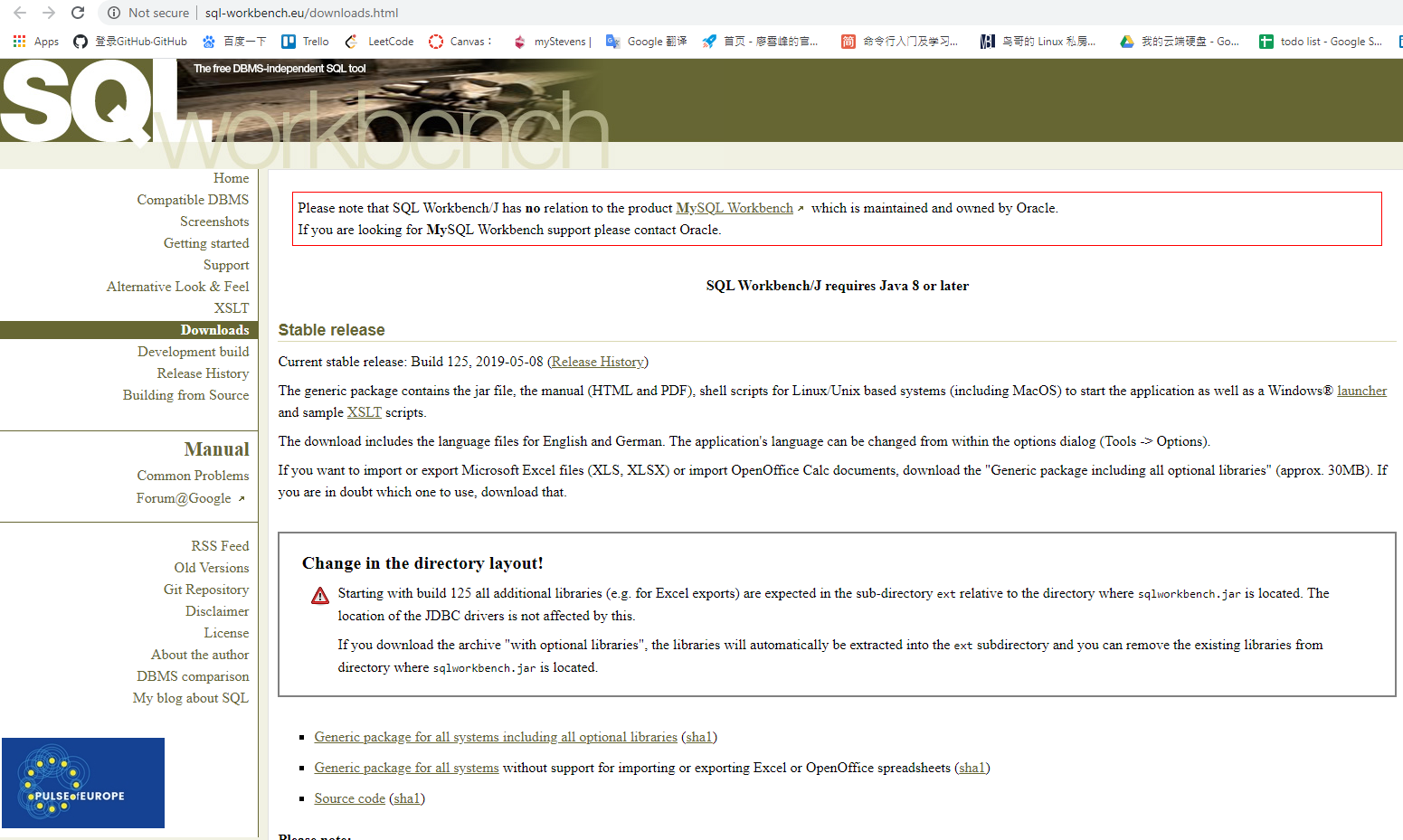


7. go to Output of this stack and record information we need later.

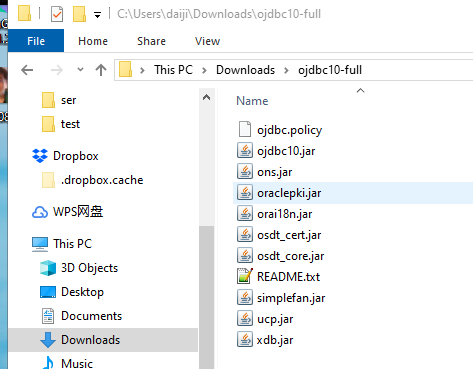


# **2.3 Install the SQL Tools and AWS Schema Conversion Tool on Local Computer**

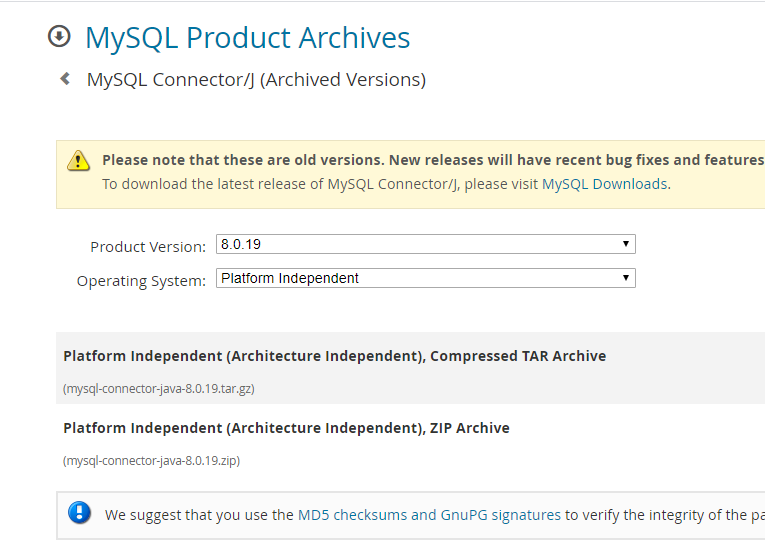
1. Download SQL Workbench/J



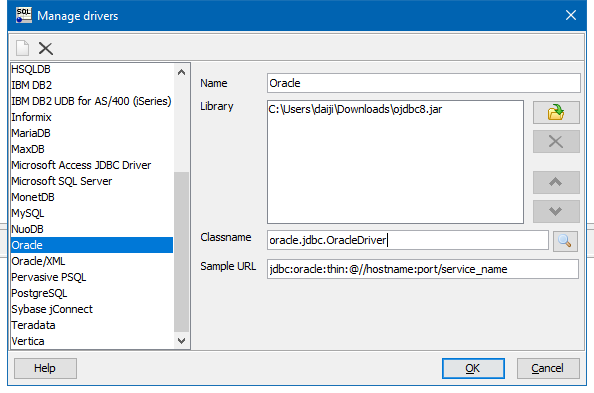
2. Download JDSC Driver and find the path of it in your computer.



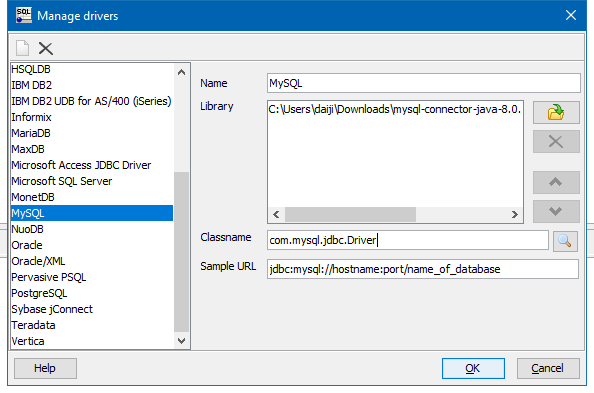
3. Download the MySQL JDBC driver (.jar file). For more information



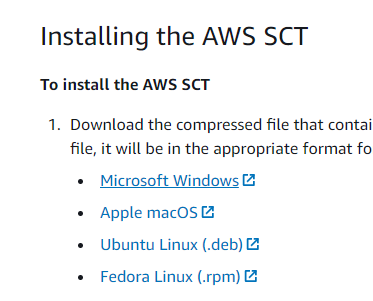
4. In SQL Workbench/J, choose File, then choose Manage Drivers.From the list of drivers, choose Oracle. Choose the Open icon, then choose the .jar file for the Oracle JDBC driver that you downloaded in the previous step. Choose OK.



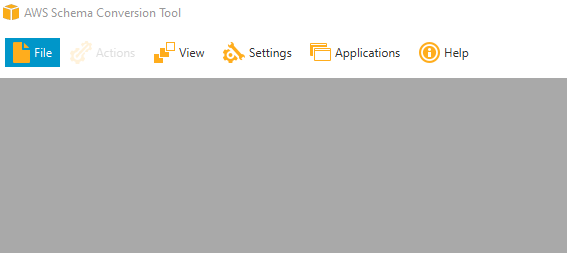
5. From the list of drivers, choose MySQL.Choose the Open icon, then choose the MySQL JDBC driver that you downloaded in the previous step. Choose OK.



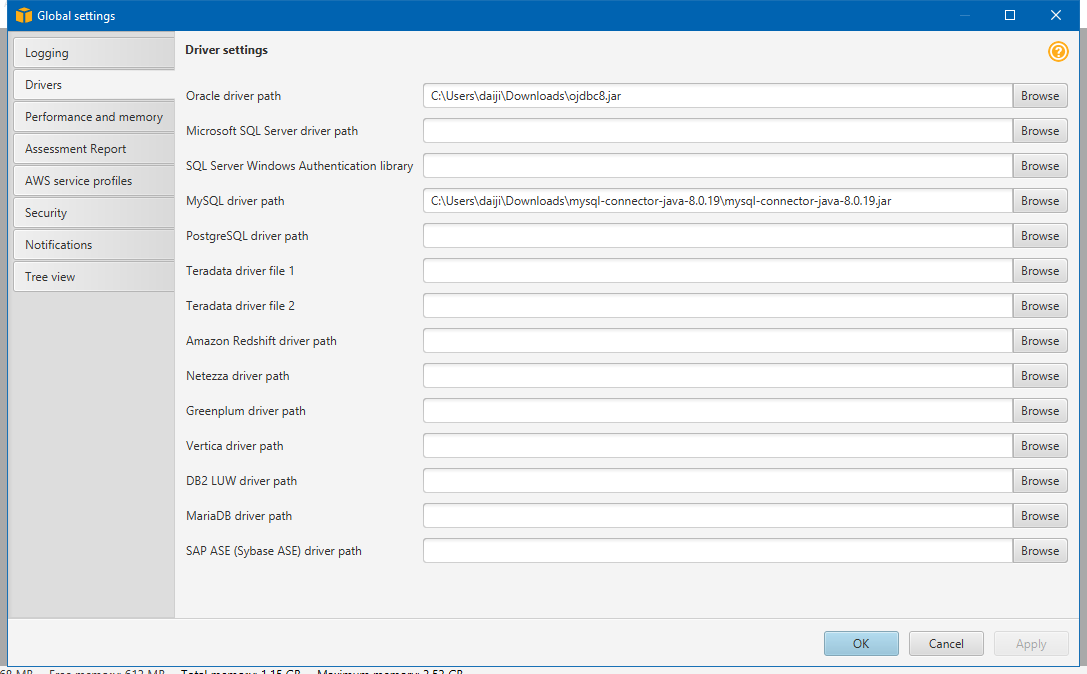
6. Download AWS Schema Conversion Tool



7. Launch AWS Schema Conversion Tool

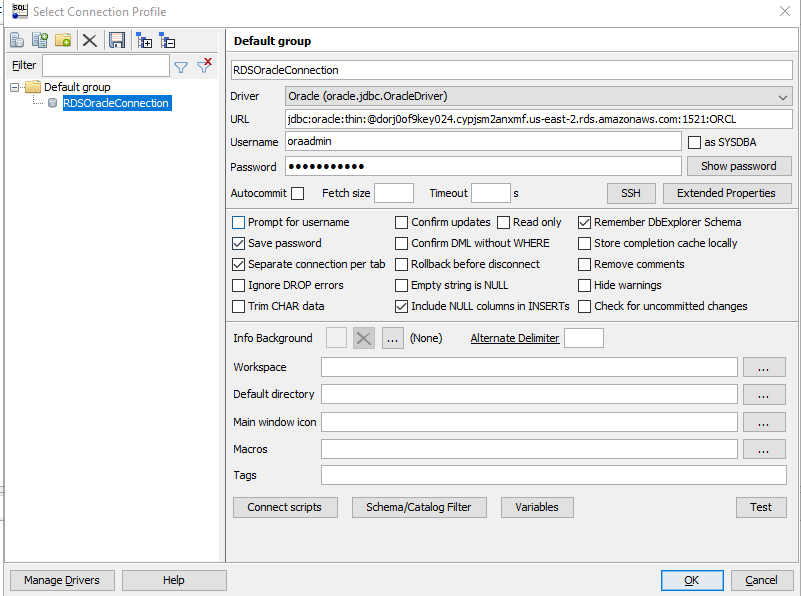


8. In the AWS Schema Conversion Tool, choose Global Settings from Settings.

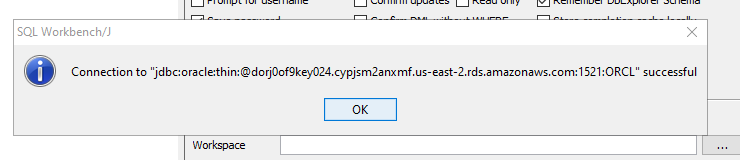


# **2.4 Test Connectivity to the Oracle DB Instance and Create the Sample Schema**

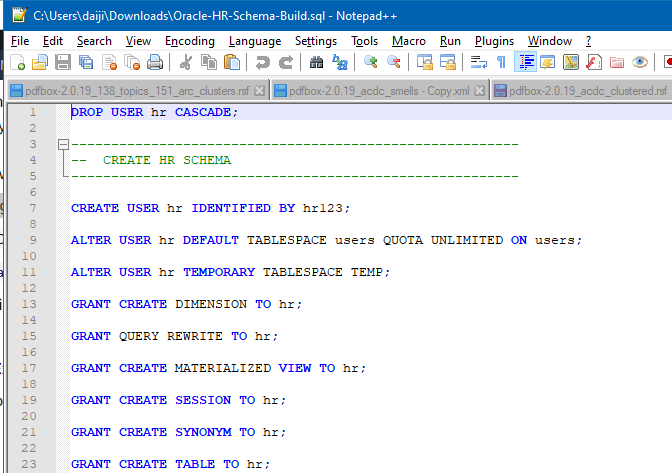
1. In SQL Workbench/J, choose File, then choose Connect window. Create a new connection profile using the following information as shown following.



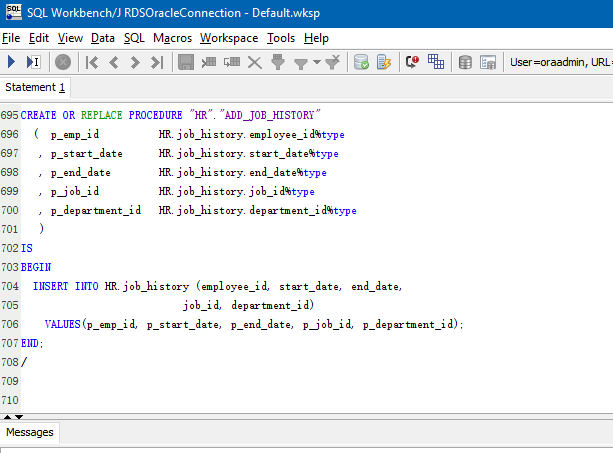
2. Test the connection by choosing Test. Choose OK to close the dialog box, then choose OK to create the connection profile.



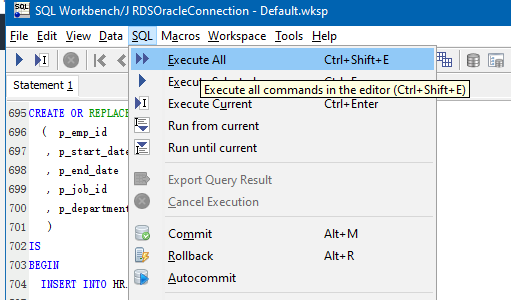
3. Open the SQL script we downloaded before in a text editor.



4. In SQL Workbench/J, paste the SQL script in the Default.wksp window showing Statement 1.

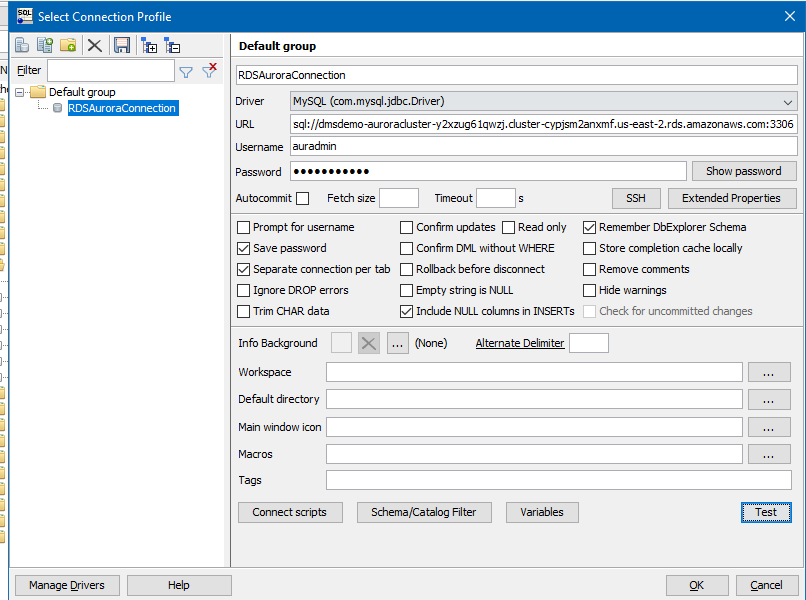


5. Choose SQL, then choose Execute all.

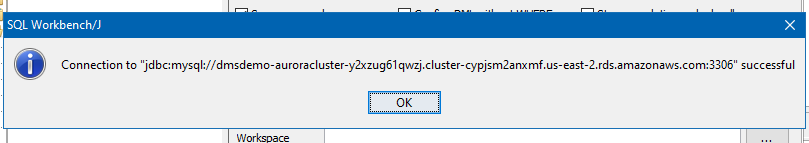


# **2.5 Test the Connectivity to the Aurora MySQL DB Instance**

1. In SQL Workbench/J, choose **File**, then choose **Connect window**. Choose the Create a new connection profile icon. using the following information: Connect to the Aurora MySQL DB instance in SQL Workbench/J by using the information as shown following

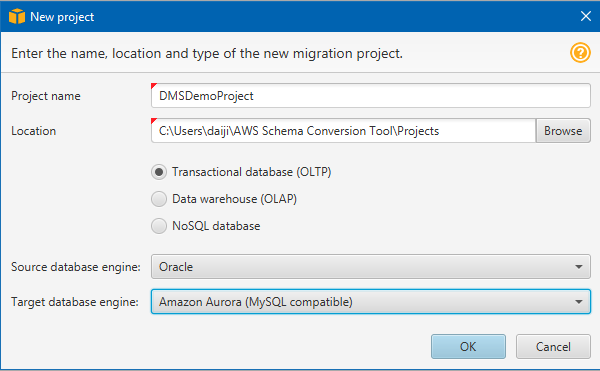


2. Test the connection by choosing Test. Choose OK to close the dialog box, then choose OK to create the connection profile.

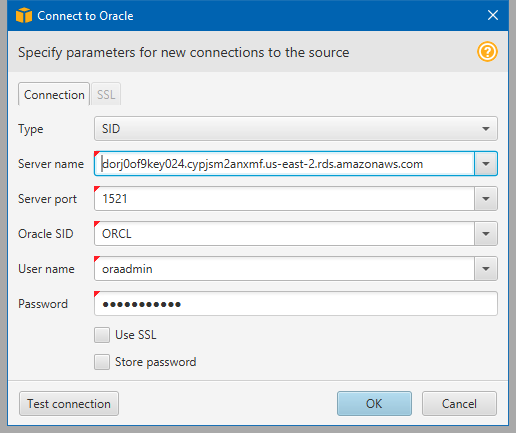


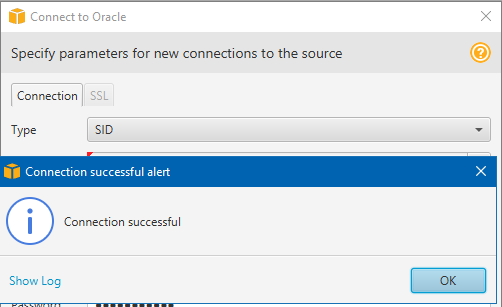
# **2.6 Use the AWS Schema Conversion Tool (AWS SCT) to Convert the Oracle Schema to Aurora MySQL**

# 1. Launch the AWS Schema Conversion Tool (AWS SCT). In the AWS SCT, choose **File**, then choose **New Project**. Create a new project called **DMSDemoProject**. Enter the following information in the New Project window and then choose **OK**.

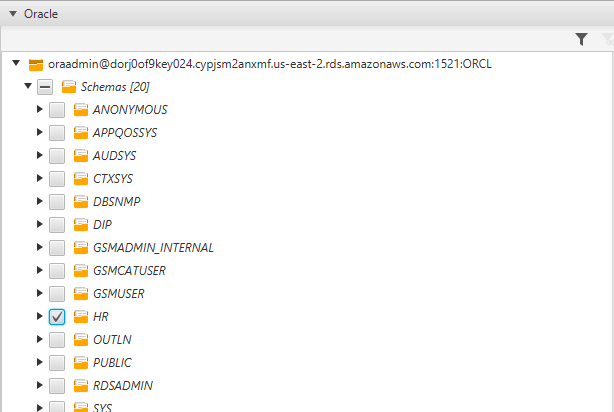


2. Choose Connect to Oracle. In the Connect to Oracle dialog box, enter the following information, and then choose Test Connection.

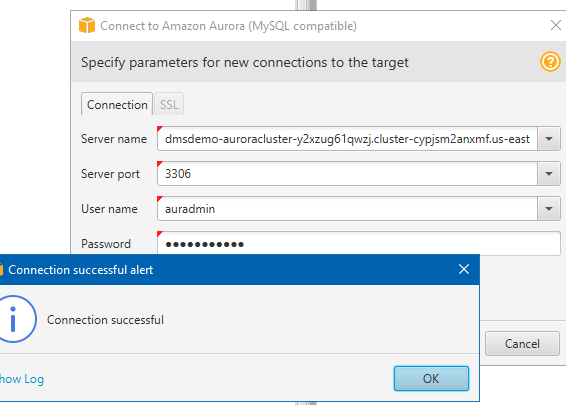




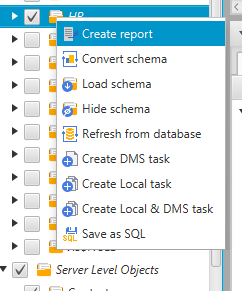
3. Choose **OK** to close the alert box, then choose OK to close the dialog box and to start the connection to the Oracle DB instance. The database structure of the Oracle DB instance is shown. Select only the HR schema.



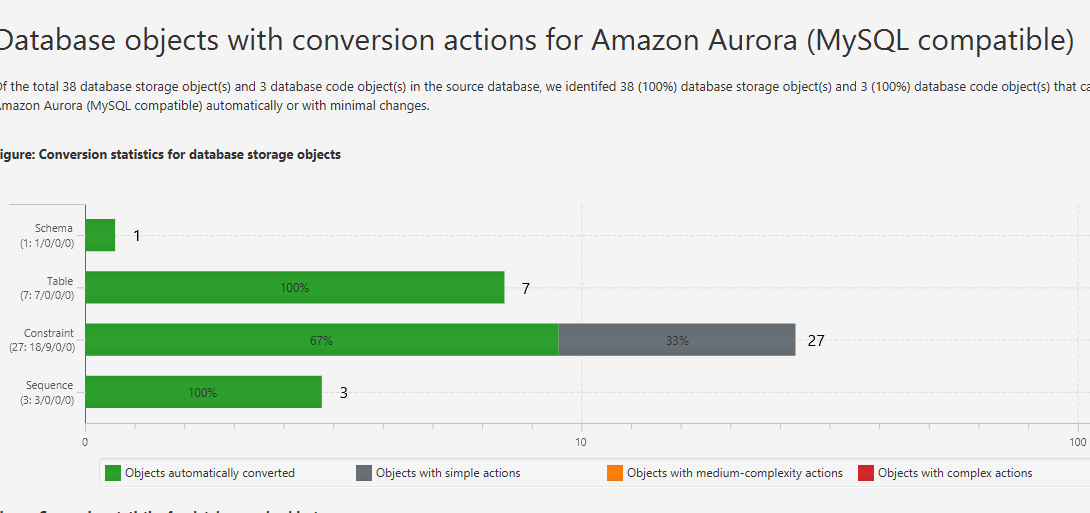
4. Choose Connect to Amazon Aurora. In the Connect to Amazon Aurora dialog box, enter the following information and then choose Test Connection.



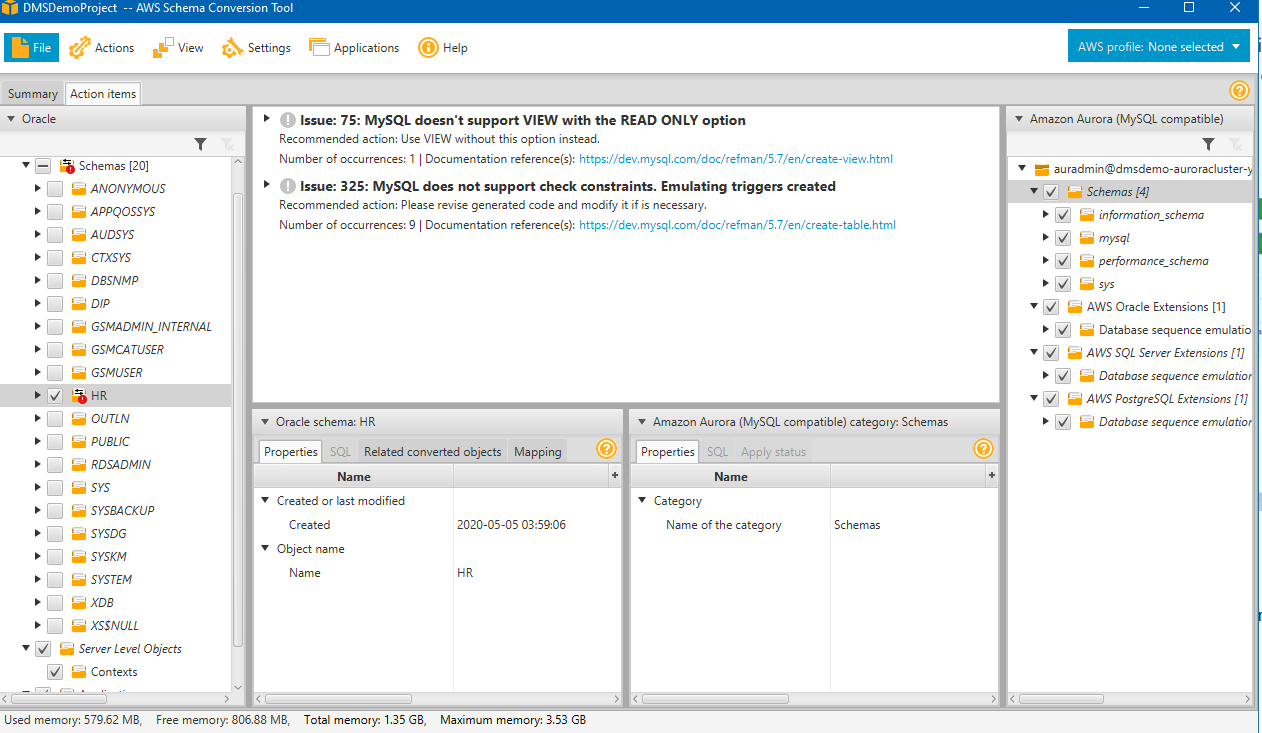
5. Right-click the HR schema and select Create Report.



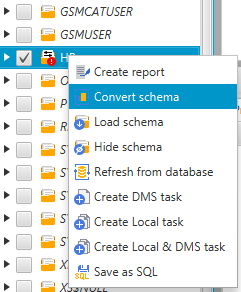
6. Check the report and the action items it suggests. The report discusses the type of objects that can be converted by using AWS SCT, along with potential migration issues and actions to resolve these issues. For this walkthrough, you should see something like the following:



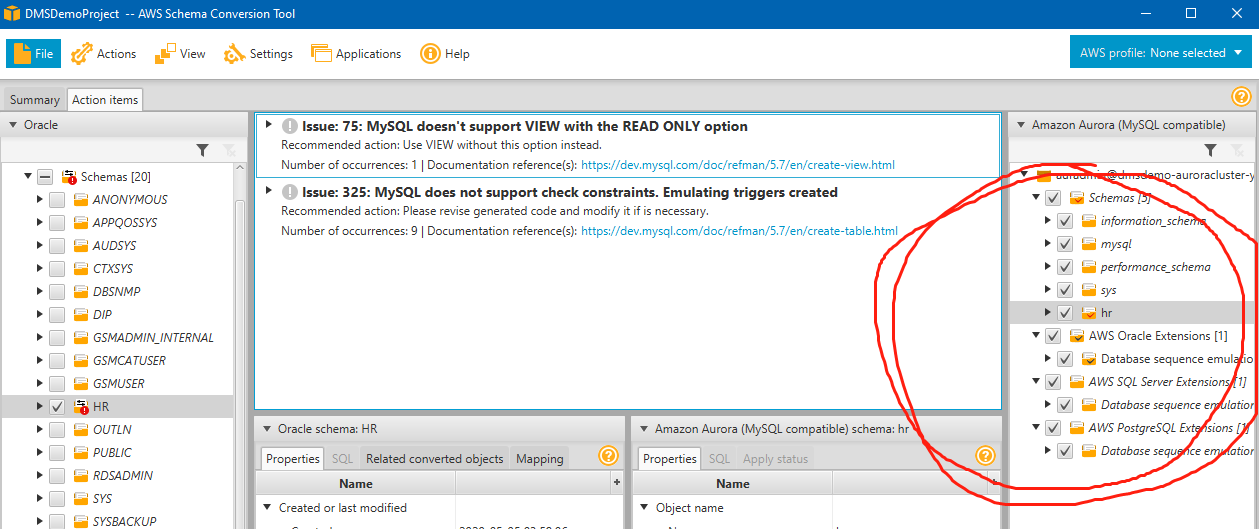
7. Choose the Action Items tab, and review any recommendations that you see.



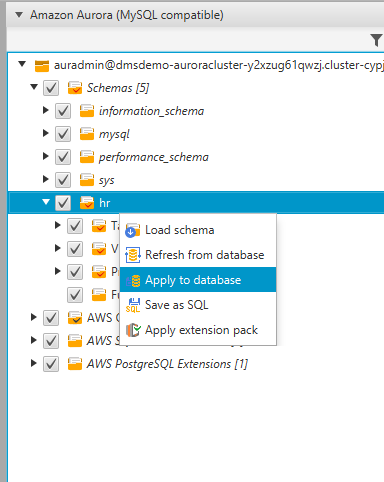
8. Right-click the HR schema, and then choose Convert schema.



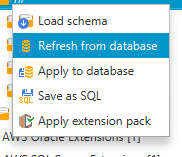
9. Choose Yes for the confirmation message. AWS SCT then converts your schema to the target database format.



10. Choose the HR schema, and then choose Apply to database to apply the schema scripts to the target Aurora MySQL instance, as shown following.



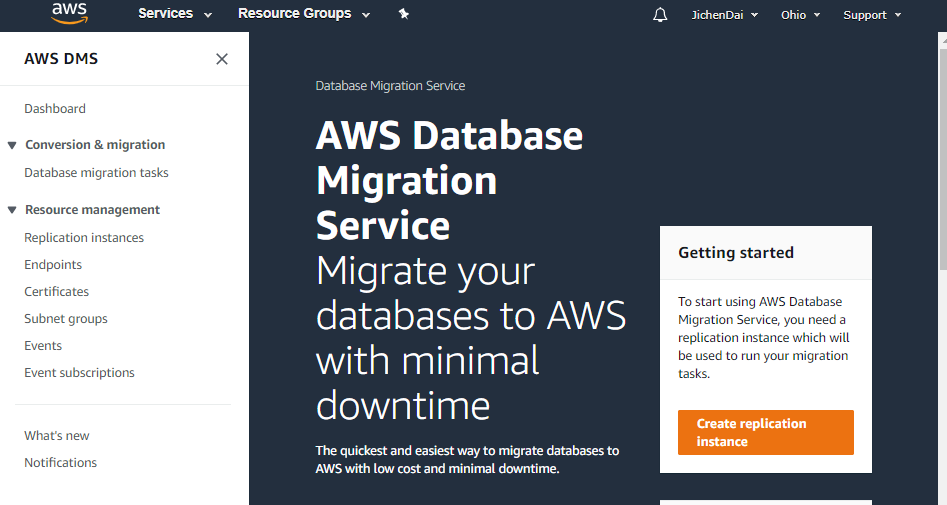
11. Choose the HR schema, and then choose Refresh from Database to refresh from the target database, as shown following.



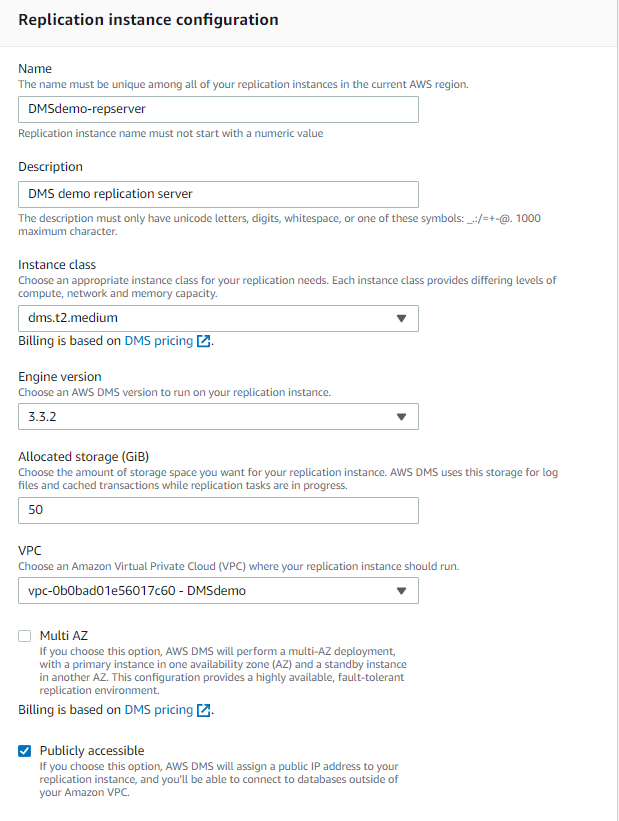
# **2.7 Create a AWS DMS Replication Instance**

# The replication instance has influences the overall time required for the migration

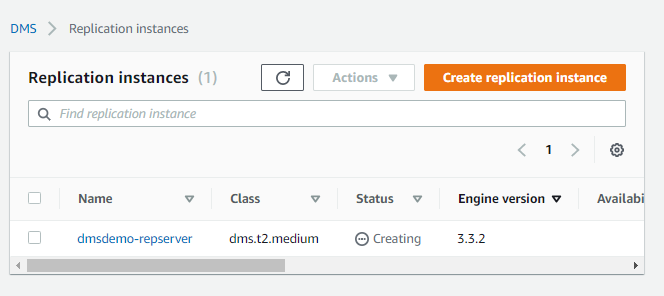
1. In AWS Management Console, select AWS DMS. click **create replication instance**.



2. specify your replication instance information as shown following.

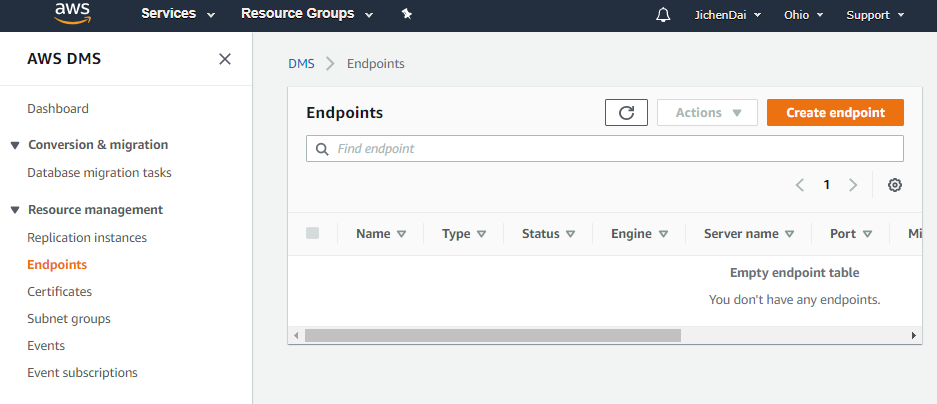


3. Click Create.

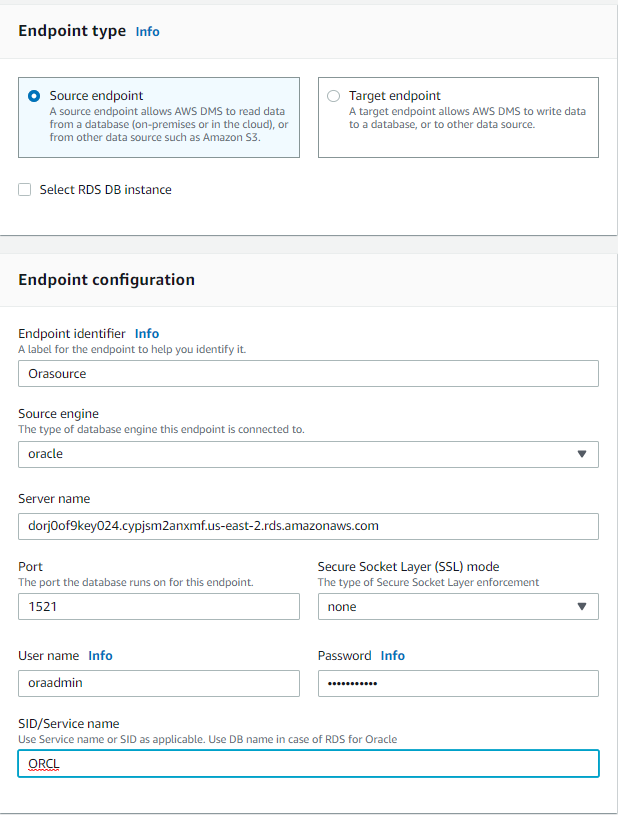


# **2.8 Create AWS DMS Source and Target Endpoints**

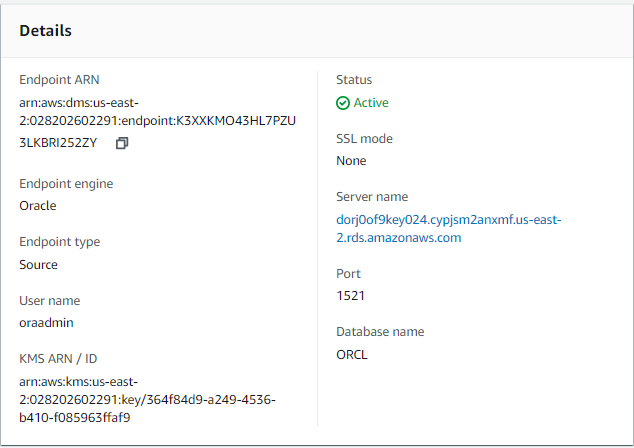
1. In Endpoints page, click **Create Endpoint**.



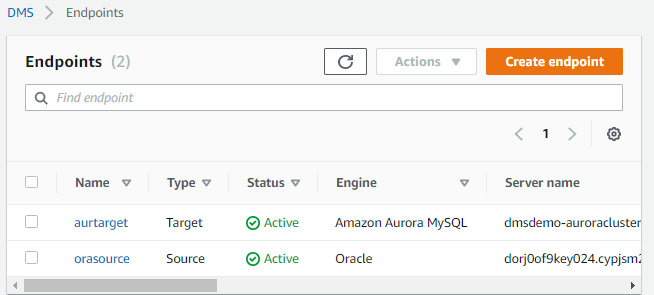
2. Specify your connection information for the source Oracle database and the target Amazon Aurora MySQL database. The following table describes the source settings.



2. The following table describes the target settings.

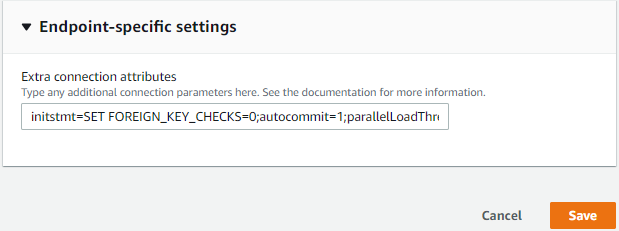


Now we have two endpoints inside our instance.



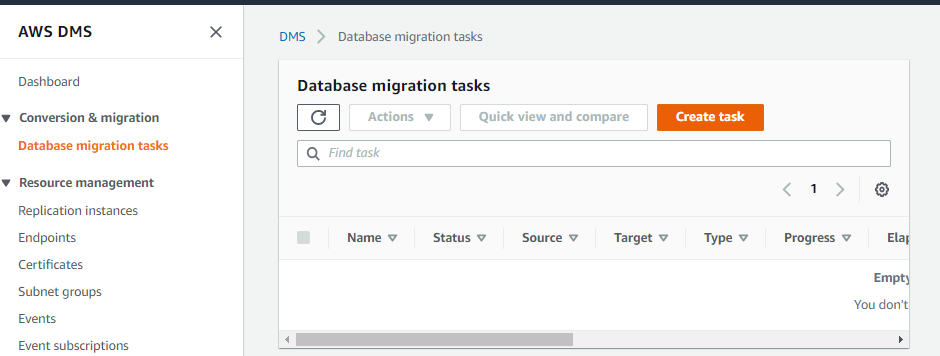
3. In order to disable foreign key checks during the initial data load, you must add the following commands to the target Aurora MySQL DB instance. In the **Advanced** section, shown following, type the following commands for **Extra connection attributes**: initstmt=SET FOREIGN\_KEY\_CHECKS=0;autocommit=1

The first command disables foreign key checks during a load, and the second command commits the transactions that DMS executes.

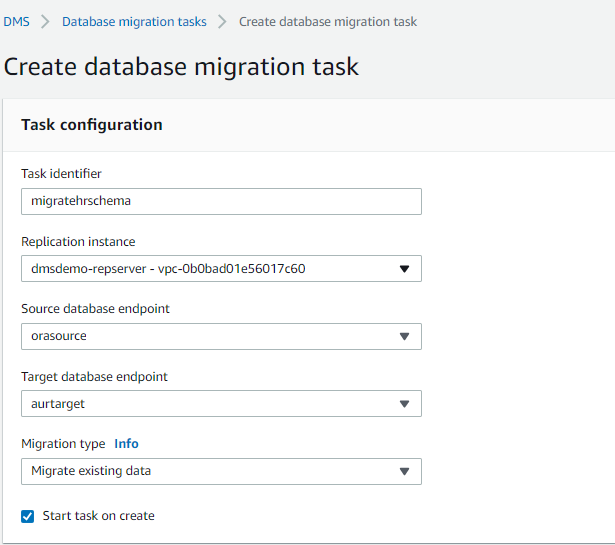


# **2.9: Create and Run Your AWS DMS Migration Task**

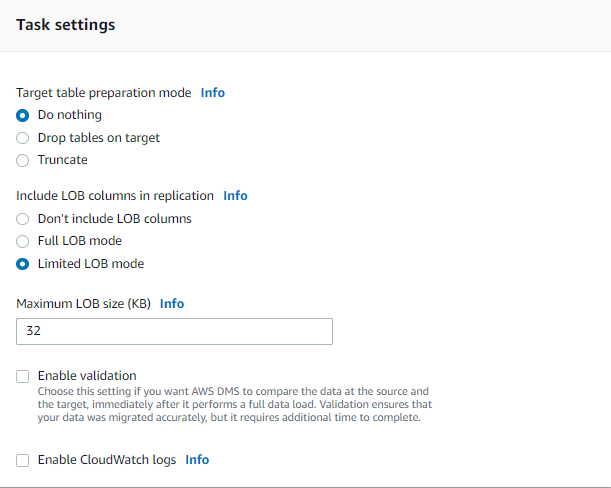
1. Go to Database migration task page, click **Create task**.



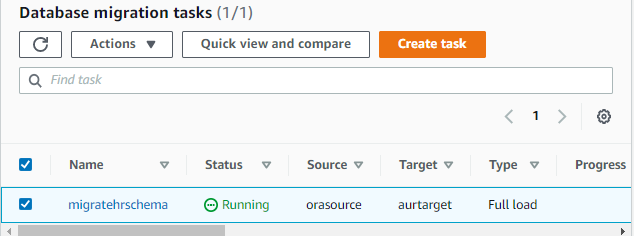
2. On the Create Task page, specify the task options. The following table describes the settings.



3. Under Task Settings, choose Do nothing for Target table preparation mode



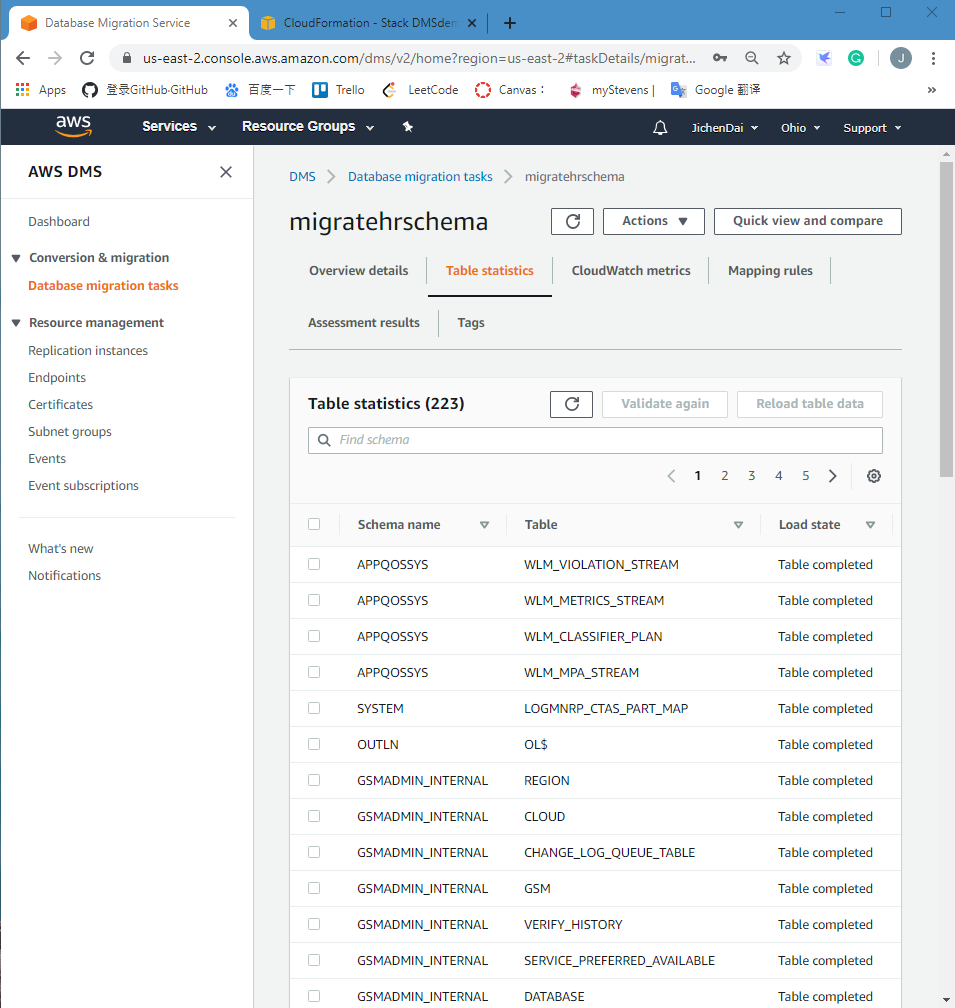
4. Choose Table mappings, choose Default for Mapping method, choose HR for Schema to migrate.Click **Create task**.



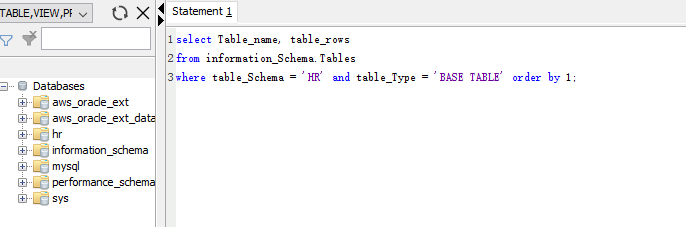
**2.10: Verify That Your Data Migration Completed Successfully**

1. In the migration tasks page. choose your task just created. Choose **Table statistics** tab.

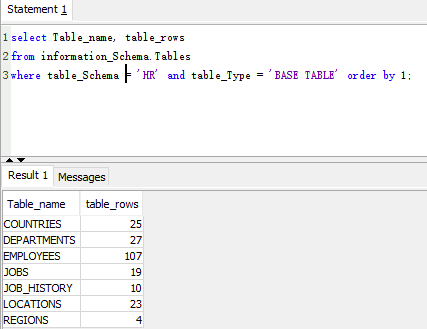
Now you can see information about tables.



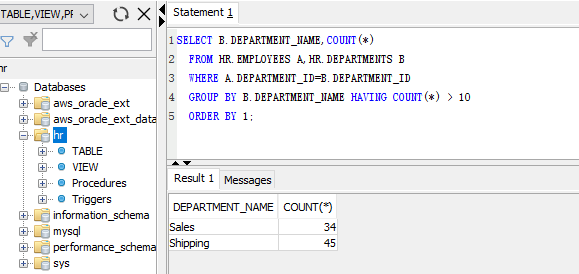
2. **Validation:** Connect to the Amazon Aurora MySQL instance by using SQL Workbench/J, and then check if the database tables were successfully migrated from Oracle to Aurora MySQL by running the SQL script shown following.



Result of our query is shown below



3. **Validation**: Run the f**ollowing** query to check the relationship in tables; this query checks the departments with employees greater than 10.



**Now we have successfully completed a database migration from an Amazon RDS Oracle DB instance to Amazon Aurora MySQL.**

# **2.11 Delete Resources**

For future inspection, I didn’t delete my resources immediately.

**3. Observation and Conclusion**

**3.1 Observation**

**Limitation:** Amazon DMS has a limitation on type of Instances that can be used for RDS. For example: all kinds of t2 instances doesn’t support MySQL and Oracle. Additionally, Certain version of database such as Amazon Aurora MySQL is not available in all regions. Amazon Aurora MySQL is currently available in US East (N. Virginia), US West (Oregon), EU (Ireland), Asia Pacific (Tokyo), Asia Pacific (Mumbai), Asia Pacific (Sydney), and Asia Pacific (Seoul).

**Advantage:** As we can see, using amazon DMS to migrate our database or data warehouse is very convenient. It not only save times but also save costs, since we don’t need to hire experts to do those bunch of things. Additionally, it support many kinds of databases and many kinds of machines. AWS database migration service can be used for migration of Oracle, SQL Server PostgreSQL, Amazon Aurora, MySQL or MariaDB. DMS supports migration between the same types, such as Oracle to Oracle, and migration between different database platforms, such as Oracle to Amazon Aurora or SQL Server to MySQL. The AWS database migration service can migrate the on-premises database to Amazon RDS or Amazon EC2, migrate the database running on EC2 to RDS, or reverse the migration, and also can migrate an RDS database to another RDS database.

**3.2 Conclusion**

The AWS database migration service needs to build and manage a replication instance on AWS. This example takes data from the source database and loads it into the target database. It can be continuously copied after a one-time migration, thereby minimizing downtime during the migration process. In this process, DMS will handle the complex details related to migration, including data type conversion from one database platform to another database platform (for example, from Oracle to Aurora). This service will also monitor the health status of the copy and the instance, if there is an error, it will give a prompt, and if necessary, it will automatically provide a replacement instance.

Although the DMS endpoint must always be located in AWS-running on the EC2 virtual machine or in the Relational Database Service (Relational Database Service)-but other endpoints can be located on any accessible host. Replica instances are located between endpoints and handle the secure transmission of data. DMS can be used to achieve a one-time migration, but also to achieve subsequent replication of the database. When configuring the migration task, the user can choose to "fully load existing data, fully load subsequent data changes, or simply copy subsequent data changes." DMS supports a wide range of relational databases, supports migration between similar database platforms, and also supports migration from one database platform to another database platform.

**Reference**

# **AWS Database Migration Service:** [https://docs.aws.amazon.com/zh\_cn/dms/latest/userguide/CHAP\_GettingStarted.html#CHAP\_GettingStarted.Dashboard](https://docs.aws.amazon.com/zh_cn/dms/latest/userguide/CHAP_GettingStarted.html" \l "CHAP_GettingStarted.Dashboard)

# **Migrating an Amazon RDS Oracle Database to Amazon Aurora MySQL**

<https://docs.aws.amazon.com/dms/latest/sbs/CHAP_RDSOracle2Aurora.Steps.DeleteResources.html>

**MySQL Document**

<https://dev.mysql.com/doc/workbench/en/wb-sql-editor-query-panel.html>