

<<Database name>> DATABASE MIGRATION SUMMARY

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Change Reference** |
|  |  |  |  |

Review Summary

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Version Approved** | **Position** | **Date** |
|  |  |  |  |

Table of Contents

[Overview – Database Name 3](#_Toc71999483)

[DB Details 3](#_Toc71999484)

[Current State 3](#_Toc71999485)

[Migrated State 3](#_Toc71999486)

[Database Migration Details 4](#_Toc71999487)

[Datatypes 4](#_Toc71999488)

[New DB objects added 4](#_Toc71999489)

[DB Objects dropped 4](#_Toc71999490)

[ SQL Jobs migration 5](#_Toc71999491)

[ Linked Servers migration 5](#_Toc71999492)

[ Mapping rules 5](#_Toc71999493)

[ Database Migration steps 5](#_Toc71999494)

[ Data load steps using DMS 5](#_Toc71999495)

[ Manual conversion 6](#_Toc71999496)

[ Cross DB/Script Dependencies 6](#_Toc71999497)

[ Validation Failed 6](#_Toc71999498)

[Issues Found 6](#_Toc71999499)

[Known Issues 7](#_Toc71999500)

[Inline Queries 7](#_Toc71999501)

[Recommendations/Conclusion 7](#_Toc71999502)

[References 7](#_Toc71999503)

## Overview – Database Name

Backend of the <<name of application>> is Microsoft SQL Server <<version>> database. The scope of engagement is to migrate <<Customer>> <<Database name>> from MS SQL Server to Aurora PostgreSQL RDS in AWS.

## DB Details

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Source Database Details** | | | | | | |
| Source DB Type | Source Server Name | Database Name | Username | Database Owner | DB Version | Source Code Branch |
| SQL Server |  |  | xxx\_ReadOnly | xxxx |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Target Database Details** | | | | | | |
| Target DB Type | Target Server Name | Database Name | Username | Migration Lead | DB Version | Migrated Code Branch |
| Amazon Aurora Postgres | [<<Server](https://console.aws.amazon.com/rds/home?region=us-east-1#database:id=ma-staging-1;is-cluster=false;tab=connectivity) Name>> |  | xx\_admin\_user | xxxx |  |  |

**DMS Instance Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AWS Micro Account | Source Endpoint | Replication Instance | Target Endpoint | Migration Task Name |
| xxxxx |  |  |  |  |

## Current State

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Database Objects** | | | | | | | | | | | | |
| Tables | Database Size | schema | Procedures | Primary Key constraint | Foreign Key Constraint | Unique Constraints | Default constraint | Functions | Index | Views | SQL Jobs | Linked Server |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

## Migrated State

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Database Objects** | | | | | | | | | | | | |
| Tables | Database Size | schema | Procedures | Primary Key constraint | Foreign Key Constraint | Unique Constraints | Default constraint | Functions | Index | Views | SQL Jobs | Linked Server |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

**Note:** We could identify Object count mismatches occurring due to frequent database refresh.

Provide any data you want to highlight.

## Database Migration Details

The following table shows the SQL Server source data types and target equivalent data types in Aurora Postgres which are being used in <<list number>> of database tables. Sample data has been provided below. Please note this might change according to your Database.

### Datatypes

|  |  |  |
| --- | --- | --- |
| **SLNo** | **SQL Server Type** | **Aurora Postgres Type** |
| 1 | BIGINT | BIGINT |
| 2 | BIT | BOOLEAN |
| 3 | CHAR | CHARACTER |
| 4 | DATE | DATE |
| 5 | DATETIME | TIMESTAMP WITHOUT TIME ZONE(3) |
| 6 | DATETIME2 | TIMESTAMP WITHOUT TIME ZONE(6) |
| 7 | DECIMAL | NUMERIC |
| 8 | FLOAT | DOUBLE PRECISION |
| 9 | INT | INTEGER |
| 10 | NCHAR | CHAR |
| 11 | NTEXT | TEXT |
| 12 | NVARCHAR | CHARACTER VARYING |
| 13 | NVARCHAR(MAX) | TEXT |
| 14 | TEXT | TEXT |
| 15 | UNIQUEIDENTIFIER | UUID |
| 16 | VARBINARY | BYTEA |
| 17 | VARBINARY(MAX) | BYTEA |
| 18 | VARCHAR | CHARACTER VARYING |
| 19 | VARCHAR(MAX) | TEXT |
| 20 | XML | XML |

### New DB objects added

|  |  |  |  |
| --- | --- | --- | --- |
| **SLNo** | **Object Type** | **Object Name** | **Reason** |
| 1 | schema | utility |  |
| 2 | Function | utility.char() |  |
| 3 | Table | <<schema>>.<<tablename>> | <<reason>> |
| 4 | Function | <<schema>>.<<functionname>> | Inline query conversion |

### DB Objects dropped

|  |  |  |  |
| --- | --- | --- | --- |
| **SLNo** | **Object Type** | **Object Name** | **Reason** |
| 1 | Schema | <<objectname>> | Out of scope |
| 2 | Table | <<objectname>> | Out of scope |
| 3 | Table | <<objectname>> | Out of scope |

### SQL Jobs migration

* + <List out details of any SQL Jobs if existing in the DB currently being migrated>

### Linked Servers migration

* + <List out details of any linked servers if existing>

### Mapping rules

<<Provide notes on Mapping rules, attach mapping rule JSON object >>

AWS SCT attempts to create an equivalent schema in Amazon Aurora RDS DB instance wherever possible. Not all data types can be converted automatically. You'll need to address these issues manually. If no direct conversion is possible, AWS SCT provides a scope to create mapping rules. We can import JSON script for mapping rules into SCT. Attached is the JSON file which can be used to create rules

### Database Migration steps

<<Sample steps as below>>

* + Execute 1.create-schema.sql file to create schema’s in the target <<DBName>> database.
  + Execute 2. create-type.sql to create user-defined type.
  + Execute 3. create-table.sql file to create tables in respective schemas.
  + Execute 4.create-constraint.sql file to create primary and unique key constraints.
  + Do the database full load using AWS DMS, steps are described later in the document.
  + Validate and verify the data on source and target.
  + Execute 5.create-foreign-key-constraint.sql file to create foreign keys.
  + Execute 6.create-indexes.sql file to create indexes.
  + Execute 7.create-functions.sql file to create functions on the target database.

### Data load steps using DMS

<<Sample steps as below>>

* + Create Replication instance with a storage of <<#>> GB within in the same VPC where Aurora Postgres RDS instance is hosted.
  + Create Source endpoint for SQL server and pass all the configuration parameters.
  + Create Target endpoint for Aurora Postgres RDS and pass all the configuration parameters.
  + Test the connectivity of source and target endpoint with replication instance using test connection Action.
  + Create Migrate existing data task and specify the source, target and replication instance along with migration settings.
  + Choose the Task Settings tab select Do nothing mode as the tables have been pre-created on the target.
  + Under Include LOB columns in replication section, select Limited LOB mode, Identify the chunk value by querying the source database and pass the value to Max LOB size (KB).
  + Enable CloudWatch logs as these are going to be used to debug any error’s or to monitor any task failures.
  + In the Table mappings section include schema’s which are participating in the data migration and exclude the tables which are not.
  + Save the task settings and start the task.
  + Validate the data on source and target, if validation is enabled during task creation. You can query the table *awsdms\_validation\_failures\_v1* to debug validation failures.
  + Execute 9.reset-sequences.sql file to reset the sequence values post data load.

### Manual conversion

* + Procedures and Functions were not converted automatically by AWS SCT. We have converted the procedure to be a function.

### Cross DB/Script Dependencies

* + NA

### Validation Failed

<<Provide all tables where there has been mismatch or validation failed along with reason>>

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Count Mismatch** | **Reason** |
| post\_acquisition\_merger\_text\_tracking | 4 | Special characters in text\_values for values id=1200,2661,2730,3052 |
| document\_decimal\_tracking | 1000 | created\_date\_utc and last\_updated\_date\_utc columns round off to 2 precision value for timestamp records |
| deal\_text\_tracking | 1000 | created\_date\_utc and last\_updated\_date\_utc columns round off to 2 precision value for timestamp records |
| legacy\_document\_text | 1000 | created\_date\_utc and last\_updated\_date\_utc columns round off to 2 precision value for timestamp records |

## Issues Found

<<List out all issues found and fixed. Sample data mentioned below>>

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl No** | **Issue** | **Cause** | **Solution** |
| 1 | BIT Datatype is converted to BOOLEAN type, but the default value is not getting converted | Not all tables definition can be converted automatically by AWS SCT. You'll need to address these issues manually | Modify the tables definition manually. BOOLEAN NOT NULL DEFAULT (0) to BOOLEAN NOT NULL DEFAULT ‘0’  BOOLEAN NOT NULL DEFAULT (1) to BOOLEAN NOT NULL DEFAULT ‘1’ |
| 2 | Identity columns of SQL Server tables are getting converted to BIGINT | Not all tables definition can be converted automatically by AWS SCT. You'll need to address these issues manually | Modify the tables definition manually. Replace BIGINT with INTEGER or SMALLINT type. |
| 3 | Integer type columns referring to the identity columns are getting converted to bigint because of the previous case | Not all tables definition can be converted automatically by AWS SCT. You'll need to address these issues manually | Modify the tables definition manually. Replaced BIGINT with INTEGER type by generating alter statements. |
| 4 | All the timestamp with 7 milliseconds precision values are rounded off to 6 precision. | In PostgreSQL time, timestamp, and interval accept an optional precision value range from 0 to 6 which specifies the number of fractional digits retained in the seconds field. | Validate with application team to check what suits better for the application |

## Known Issues

<<List of all known issues>>

* + Records with unique identifier column type are in UPPERCASE in SQL Server which gets converted to LOWERCASE post migrating into PostgreSQL.

## Inline Queries

<<Any reference to any inline queries converted>>

## Recommendations/Conclusion

The document captures the approach, infrastructure, code changes while migrating the <<DBName>> database from on-premises to AWS.

## References

* [AWS Schema conversion Tool](https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Welcome.html)
* [AWS Database Migration Service](https://docs.aws.amazon.com/dms/latest/userguide/Welcome.html)
* [Using a Microsoft SQL Server Database as a Source for AWS DMS](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.SQLServer.html)
* [Using a PostgreSQL Database as a Target for AWS Database Migration Service](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Target.PostgreSQL.html)