Name: Anish Mathur

E-Mail: [anishmathur82@yahoo.com](mailto:anishmathur82@yahoo.com)

Contact#: 7665176967

Assessment Test for Migration Specialist

**A. Migration of sample database**

1. Configure the Northwind sample database from this link **(https://github.com/Microsoft/sql-server-samples/tree/master/samples/databases/northwind-pubs)** and configure that locally.

2. Use any method or toolkit to migrate this to PostGRE. Create a runbook to explain the steps involved in configuration, mentioning the special aspects to be considered based on your own experience.

3. Any incompatibility needs to be noted and an approach identified for fixing that out.

**Solution:**

**Northwind Database Setup in SQL Server**

The Northwind sample database can be downloaded from the above-mentioned link.

The link includes the script to restore the database. I have restored the database on Microsoft SQL Server. Below mentioned are the list of objects created from the script.

1. Tables
   1. CustomerCustomerDemo
   2. CustomerDemographics
   3. Region
   4. Territories
   5. EmployeeTerritories
   6. Employees
   7. Categories
   8. Customers
   9. Shippers
   10. Suppliers
   11. Orders
   12. Products
   13. Order Details
2. Views
   1. Customer and Suppliers by City
   2. Alphabetical list of products
   3. Current Product List
   4. Orders Qry
   5. Products Above Average Price
   6. Products by Category
   7. Quarterly Orders
   8. Invoices
   9. Order Details Extended
   10. Order Subtotals
   11. Product Sales for 1997
   12. Category Sales for 1997
   13. Sales by Category
   14. Sales Totals by Amount
   15. Summary of Sales by Quarter
   16. Summary of Sales by Year
3. Stored Procedures
   1. Sales by Year
   2. CustOrdersDetail
   3. CustOrdersOrders
   4. CustOrderHist
   5. SalesByCategory
   6. Ten Most Expensive Products
   7. Employee Sales by Country

**Northwind Database Setup in Postgres**

I have created a database in the postgres as Northwind with default configuration.

**Migration using TALEND Open Studio**

1. DB connections were configured in the repository both for SQL Server/Northwind and Postgres/Northwind and retrieve the schema.
2. Downloaded and installed the missing jar as per the tool like MSSQL Server jdbc connector jar.
3. Created separate jobs for better understanding for each table available in the source database.
4. Below-mentioned are the screenshots of the TALEND jobs for each table migration.
   1. CustomerCustomerDemo

A close-up of a line

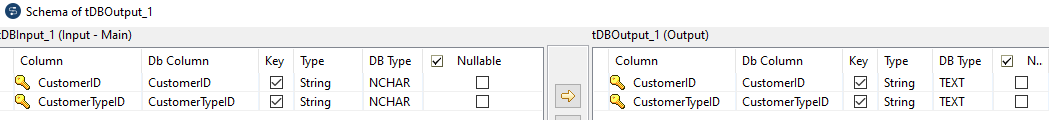
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



* 1. CustomerDemographics

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

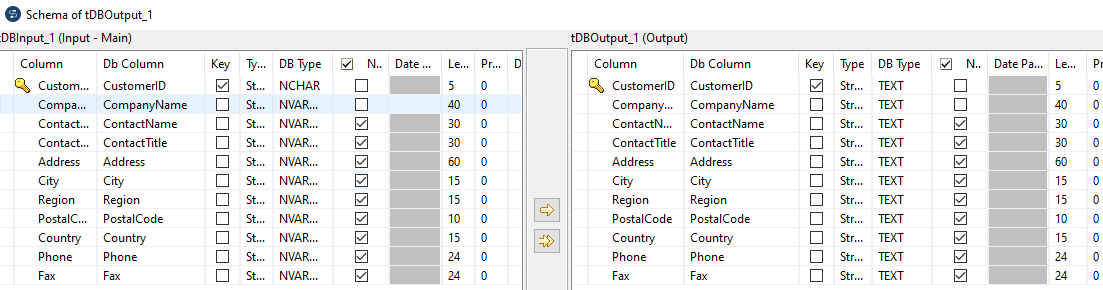
* 1. Customers

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



d.Employees

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

E. EmployeeTerritories

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

F. Order Details

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Orders

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Products

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

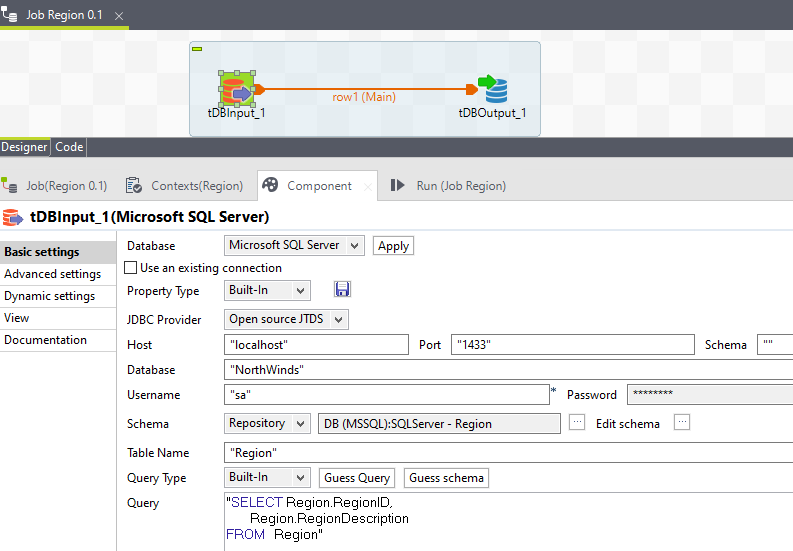
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Region



A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Shippers

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Suppliers

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Categories

A screenshot of a computer

Description automatically generated

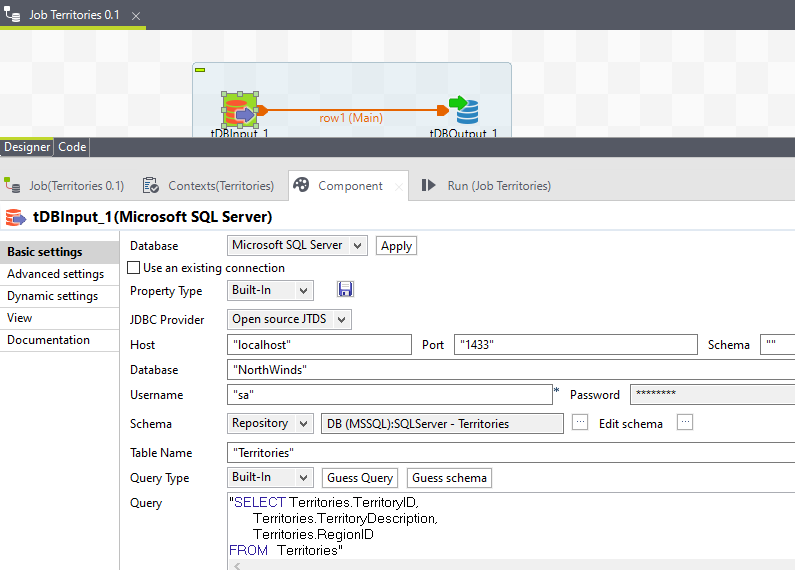
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Territories



A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Code Level Migration: All the codes including constraints, views and stored procedures are converted to postgres format. The online tool which is used to convert this code is **SQLines**.

Also any type of casting or convertions are handled in the code.



**B) Migration Strategy**

1. A client has an SQL Server database that has SSIS jobs, and a Service Broker configured on it. The database is approximately 10TB in size and grows about 10GB monthly. It currently uses 24-core VCPU and 256GB of RAM under SQL Enterprise Edition 2016. This is a transactional database that has a max downtime limit of 4 hours on special update events.

2. What will be the strategy to migrate such a database to PostGRE considering the size and transactional volume? Mention any tooling (open-source or proprietary) that can ease out this process.

3. What can be the issues being faced and possible mitigation plan?

4. What will be the roadmap for the transition and what factors will determine the timelines of such a migration?

Solutions:

1. **Migration Strategy:**

a. **Assessment and Planning:**

* Thoroughly analyze database schema, data types, dependencies, SSIS jobs, and Service Broker usage.
* Evaluate PostgreSQL compatibility with existing features and potential workarounds for unsupported ones.
* Estimate data transfer time and downtime requirements, considering network bandwidth and database size.

b. **Schema Conversion:** Use tools like:

* AWS Schema Conversion Tool (SCT): Free, supports SQL Server to PostgreSQL conversion, generates migration reports.
* pgloader: Open-source, handles data transfer with validation and transformation.
* sqlserver2pgsql: Open-source, converts schema and data using Pentaho Data Integration (Kettle).
* Manually review and adjust converted schema for compatibility.

c. **Data Migration:**

* Choose a method based on downtime constraints and data volume:
* Offline (full downtime): Faster, but requires application downtime.
* Online (minimal downtime): Uses replication or tools for continuous migration, but can be more complex.
* Optimize data transfer with compression, parallelization, and efficient tools.

d. **Application Reconfiguration:**

* Modify application code to connect to PostgreSQL using its JDBC driver.
* Adapt SSIS jobs to PostgreSQL-compatible tools like Talend Open Studio for Data Integration or Pentaho Data Integration.
* Reimplement Service Broker functionality using PostgreSQL triggers, queues, or external messaging systems.

e. **Testing and Validation:**

* Conduct thorough testing in a staging environment to ensure data integrity,

functionality, and performance.

* Address any issues before production migration.

**2. Tooling:**

a. **Assessment and Planning:**

* AWS SCT
* Idera SQL Migration (commercial)
* Quest Data Transporter (commercial)

b. **Schema Conversion:**

* AWS SCT
* pgloader
* sqlserver2pgsql

c. **Data Migration:**

* pgloader
* AWS Database Migration Service (DMS)
* Idera SQL Migration
* Custom scripts (for complex scenarios)

**3. Potential Issues and Mitigation:**

a. **Data Type Mismatch:**

* Identify and address incompatible data types during schema conversion.

b. **Unsupported Features:**

* Find workarounds or alternative implementations for unsupported features.

c. **Performance Impact:**

* Optimize PostgreSQL configuration and queries for performance.

d. **Data Integrity:**

* Implement robust validation and error handling during migration.

e. **Downtime Constraints:**

* Choose appropriate migration methods and carefully plan downtime windows.

f. **Application Refactoring:**

* Thoroughly test application changes in a staging environment.

**4. Roadmap and Timelines:**

* Assessment and Planning: 1-2 weeks
* Schema Conversion and Testing: 2-4 weeks
* Data Migration (offline or online): 1-4 weeks, depending on method and data volume
* Application Reconfiguration and Testing: 2-4 weeks
* Production Cutover and Monitoring: 1-2 weeks

**Timeline factors:**

* Database size and complexity
* Data transfer speed
* Downtime constraints
* Available resources and expertise
* Testing and validation thoroughness

**Recommendations:**

* Conduct thorough testing and validation.
* Minimize downtime through careful planning and execution.
* Optimize PostgreSQL performance post-migration.
* Continuously monitor and adjust the new environment.