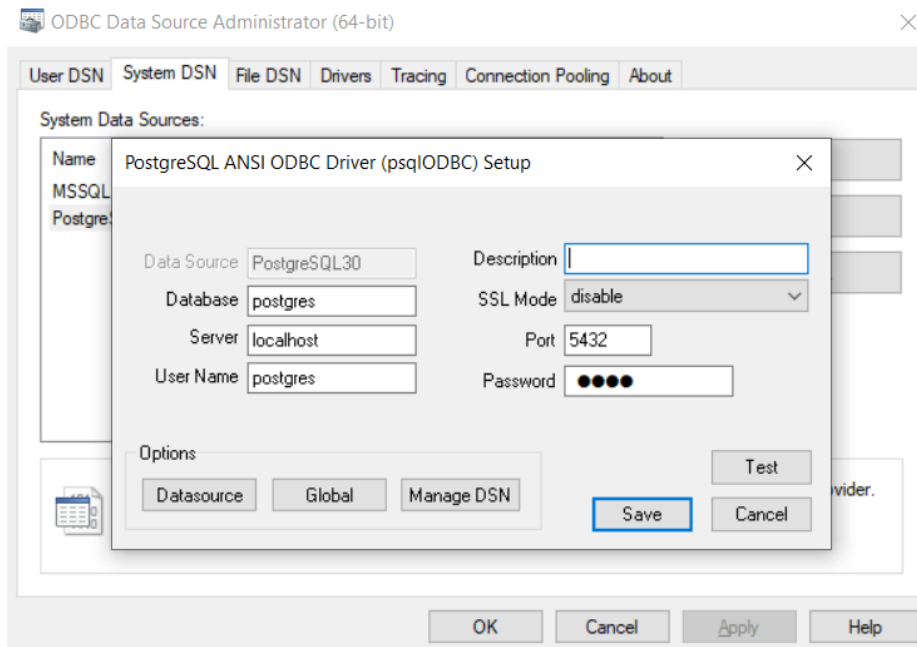


SETTING UP THE LINKED SERVER:

I am using a heterogeneous distributed database system of one instance of MSSQL and one instance of PostgreSQL.

We begin by adding the drivers to the System DSN in the ODBC gateway.

PostgreSQL-



MSSQL-

Microsoft SQL Server DSN Configuration

This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name:

How do you want to describe the data source?

Description:

Which SQL Server do you want to connect to?

Server:

Now in MSSQL under 'Linked Servers', we create POSTGRESQL. We specify the Data Source as PostgreSQL30 (from ODBC). The security settings is set to the fourth option to login with password.

Linked Server Properties - POSTGRESQL

Select a page: General, Security, Server Options

Script Help

Linked server:

Server type:

☐ SQL Server

☒ Other data source

Provider:

Product name:

Data source:

Provider string:

Location:

Catalog:

Connection

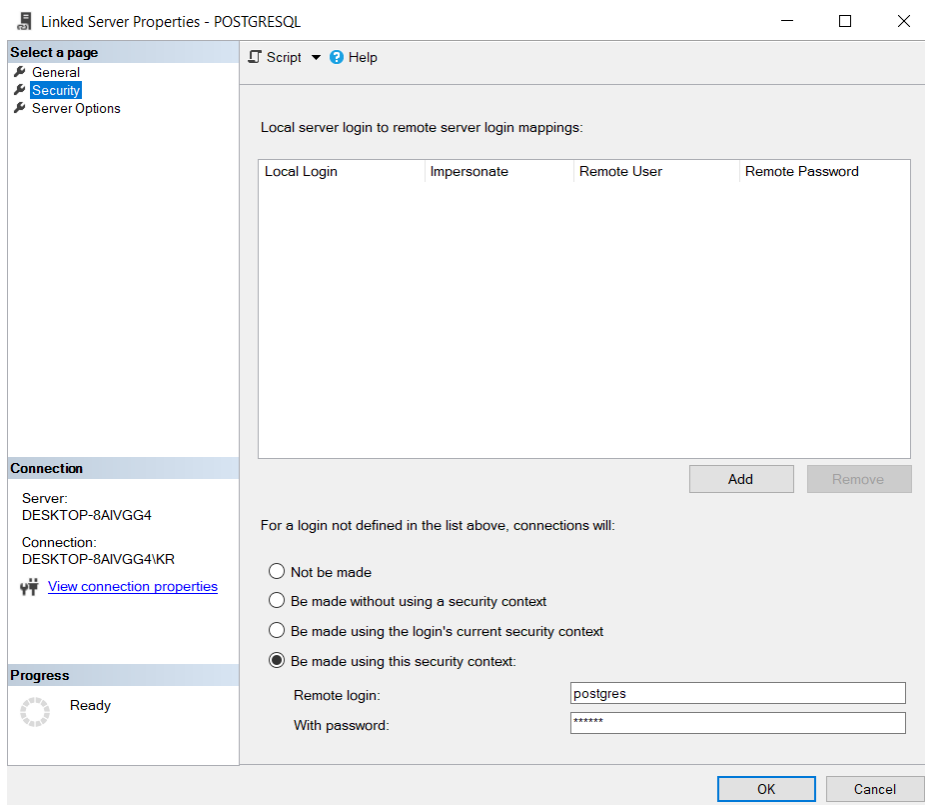
Server: DESKTOP-8AIVGG4

Connection: DESKTOP-8AIVGG4\KR

[View connection properties](#)

Progress

Ready



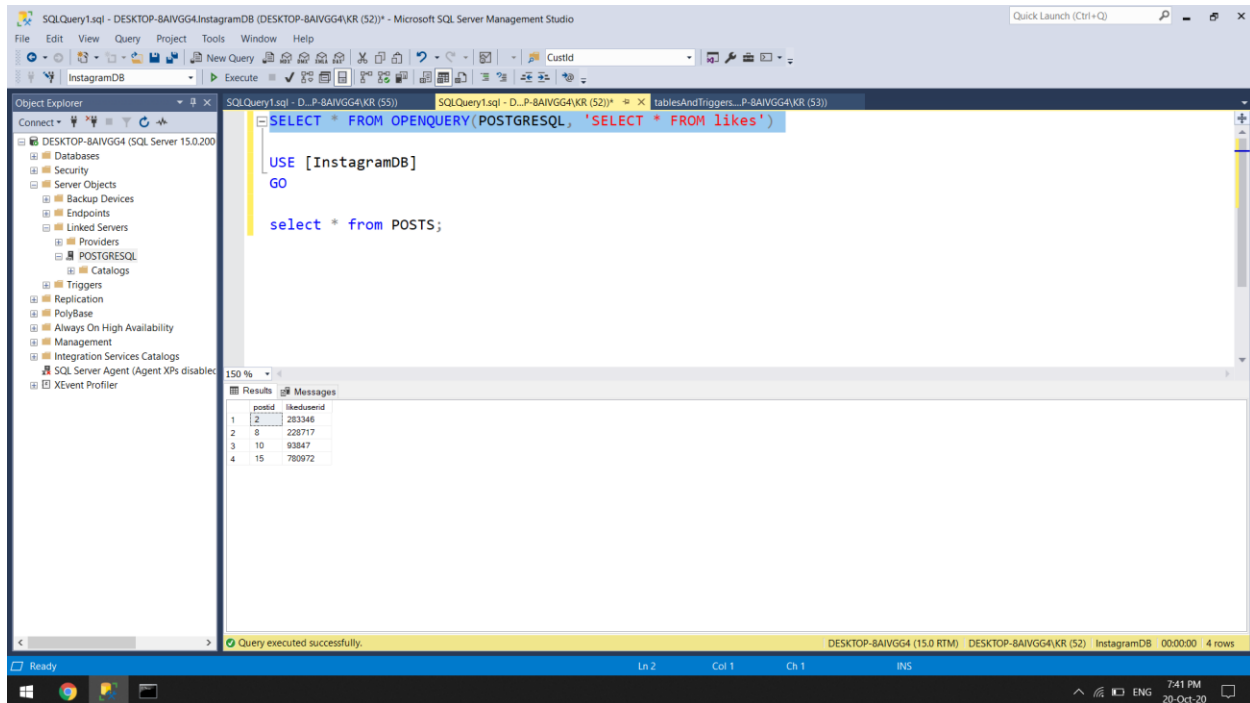
We create a table called LIKES in PostgreSQL and insert 4 rows into the table.

CREATE TABLE LIKES (postId INT NOT NULL, likedUserId INT NOT NULL, PRIMARY KEY(postId, likedUserId);

```
SQL Shell (psql)
LINE 1: clear
      ^
postgres=# create table likes(postId int not null, likedUserId int not null, primary key(postId, likedUserId));
CREATE TABLE
postgres=# insert into likes values(2, 283346);
INSERT 0 1
postgres=# insert into likes values(8, 228717);
INSERT 0 1
postgres=# insert into likes values(10, 93847);
INSERT 0 1
postgres=# insert into likes values(15, 780972);
INSERT 0 1
postgres=#
```

QUERYING DISTRIBUTED DATABASES:

We can perform a query on the table LIKES in PostgreSQL from MSSQL using the POSTGRESQL method.



We try to execute a join operation on attributes from table POSTS which is on MSSQL and LIKES which is on PostgreSQL. See the output below.

```
select P.postUserId,
       P.caption,
       P.likes,
       L.likedUserId
from POSTS P,
     (select * from openquery(POSTGRESQL, 'SELECT * FROM likes')) L
where P.postId = L.postId;
```

SQLQuery1.sql - DESKTOP-BAIVGG4\InstagramDB (DESKTOP-BAIVGG4\KR (52))* - Microsoft SQL Server Management Studio

```

select * from POSTS;

--join of one MSSQL table with a table from Postgresql
select P.postUserId,
       P.caption,
       P.likes,
       L.likedUserId
from POSTS P,
     (select * from openquery(PGSQL, 'SELECT * FROM likes')) L
where P.postId = L.postId;

```

postId	caption	likes	likedUserId
390027	Caption	0	283346
412348	Caption	0	228717
228717	I am a caption	0	93847
93847	I am a caption	0	780972

Query executed successfully.

See the execution plan of the query- a clustered index scan is being performed on POSTS whereas a Remote Scan is being performed on LIKES, following which the projected attributes of both are joined.

SQLQuery1.sql - DESKTOP-BAIVGG4\InstagramDB (DESKTOP-BAIVGG4\KR (52))* - Microsoft SQL Server Management Studio

```

select * from POSTS;

--join of one MSSQL table with a table from Postgresql
select P.postUserId,
       P.caption,
       P.likes,
       L.likedUserId
from POSTS P,
     (select * from openquery(PGSQL, 'SELECT * FROM likes')) L
where P.postId = L.postId;

```

Query 1: Query cost (relative to the batch): 100%

```

select P.postUserId, P.caption, P.likes, L.likedUserId from POSTS P, (select * from openquery(PGSQL, 'SELECT * FROM likes')) L where P.postId = ...

```

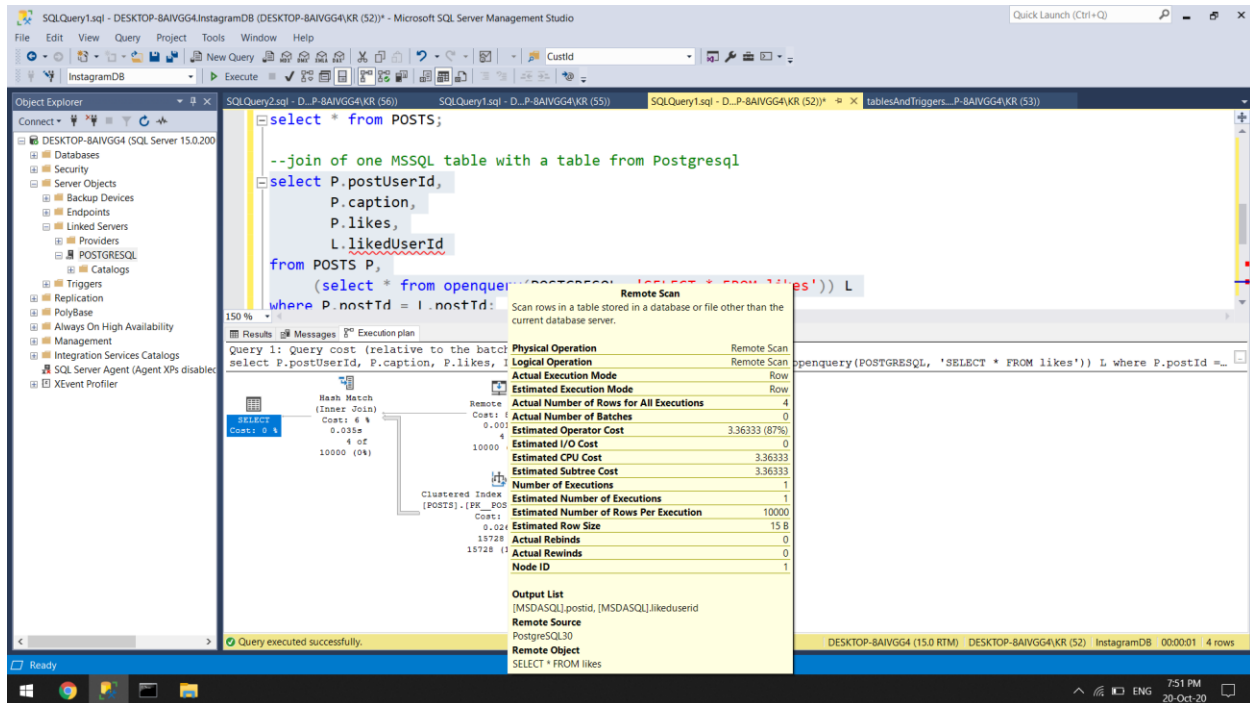
Execution Plan Details:

- Hash Match (Inner Join): Cost: 6.1, 0.035s, 4 of 10000 (0%)
- Remote Scan: Cost: 87.1, 0.001s, 4 of 10000 (0%)
- Clustered Index Scan (Clustered Index Scan on POSTS): Cost: 8.1, 0.026s, 15728 of 15728 (100%)

Query executed successfully.

In the remote scan, we can notice two things-

- The estimated I/O cost is 0 because there is no way for MSSQL to be able to estimate the cost of scan on a table in postgresql.
- Both attributes of LIKES table are seen in the output list, because we use postId in the where condition and likedUserId in the selection list.



Now we try another join operation with the only difference that the sellist and where condition, both, include postId from LIKES. So we do not need the likedUserId attribute from LIKES anymore. Now in the execution plan, in the remote scan on LIKES, the output list contains only postId.

```
select P.postUserId,
       P.caption,
       P.likes,
       L.postId
from POSTS P,
     (select * from openquery('POSTGRES', 'SELECT * FROM likes')) L
where P.postId = L.postId;
```

However, the query compiler is not reliable in such a scenario for being able to choose the most optimum execution plan. So the developer/user who is querying the distributed database must be aware of the schema and locations of the attributes involved in a distributed database query and try to write the most optimal form of the query.

SQLQuery1.sql - DESKTOP-8AIVGG4\InstagramDB (DESKTOP-8AIVGG4\KR (52)) - Microsoft SQL Server Management Studio

Object Explorer: Connect to DESKTOP-8AIVGG4 (SQL Server 15.0.2002) > Databases > InstagramDB

SQLQuery1.sql - D..P-8AIVGG4\KR (52):

```
write P.postId = L.postId;

select P.postUserId,
       P.caption,
       P.likes,
       L.postId
from POSTS P,
     (select * from openquery(POSTGRESQL, 'SELECT * FROM likes')) L
where P.postId = L.postId;
```

Execution Plan: Hash Match (Inner Join) Cost: 0.012s, 4 of 10000 (0%)

Query 1: Query cost (relative to the batch) select P.postUserId, P.caption, P.likes, L.postId

Physical Operation: Remote Scan

Logical Operation: Remote Scan

Actual Execution Mode: Row

Estimated Execution Mode: Row

Actual Number of Rows for All Executions: 0

Estimated Operator Cost: 3.36333 (87%)

Estimated I/O Cost: 0

Estimated CPU Cost: 3.36333

Estimated Subtree Cost: 3.36333

Number of Executions: 1

Estimated Number of Rows Per Execution: 10000

Estimated Row Size: 11 B

Actual Rebinds: 0

Actual Rewinds: 0

Node ID: 1

Output List: [MSDASQL].[postid]

Remote Source: PostgreSQL30

Remote Object: SELECT * FROM likes

Query executed successfully.

DESKTOP-8AIVGG4 (15.0 RTM) DESKTOP-8AIVGG4\KR (52) InstagramDB 00:00:00 4 rows

SQLQuery1.sql - DESKTOP-8AIVGG4\InstagramDB (DESKTOP-8AIVGG4\KR (52)) - Microsoft SQL Server Management Studio

Object Explorer: Connect to DESKTOP-8AIVGG4 (SQL Server 15.0.2002) > Databases > InstagramDB

SQLQuery1.sql - D..P-8AIVGG4\KR (52):

```
write P.postId = L.postId;

select P.postUserId,
       P.caption,
       P.likes,
       L.postId
from POSTS P,
     (select * from openquery(POSTGRESQL, 'SELECT * FROM likes')) L
where P.postId = L.postId;
```

Execution Plan: Hash Match (Inner Join) Cost: 0.012s, 4 of 10000 (0%)

Query 1: Query cost (relative to the batch) select P.postUserId, P.caption, P.likes, L.postId

Physical Operation: Hash Match

Logical Operation: Inner Join

Actual Execution Mode: Row

Estimated Execution Mode: Row

Actual Number of Rows for All Executions: 0

Estimated Operator Cost: 0.194875 (5%)

Estimated I/O Cost: 0

Estimated CPU Cost: 0.194866

Estimated Subtree Cost: 3.85138

Number of Executions: 1

Estimated Number of Rows Per Execution: 10000

Estimated Row Size: 73 B

Actual Rebinds: 0

Actual Rewinds: 0

Node ID: 0

Output List: [InstagramDB].[dbo].[POSTS].[postUserId], [InstagramDB].[dbo].[POSTS].[caption], [InstagramDB].[dbo].[POSTS].[likes], [MSDASQL].[postid]

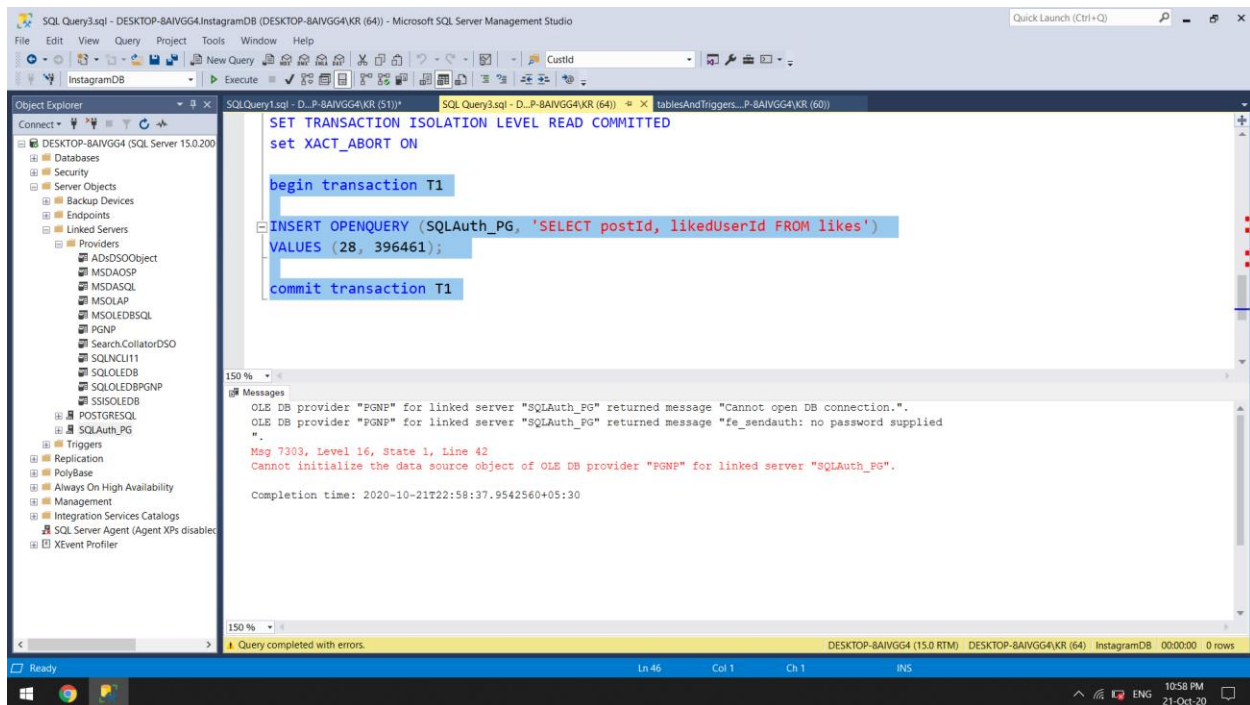
Hash Keys Probe: [InstagramDB].[dbo].[POSTS].[postid]

Query executed successfully.

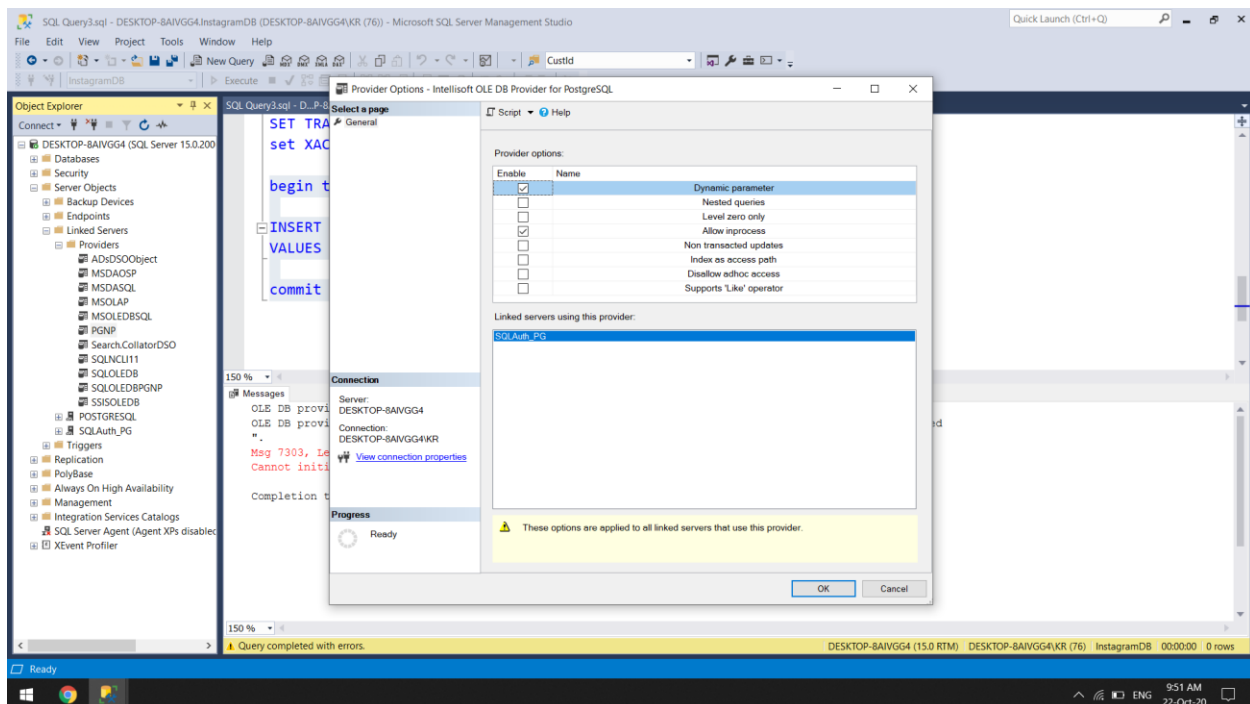
DESKTOP-8AIVGG4 (15.0 RTM) DESKTOP-8AIVGG4\KR (52) InstagramDB 00:00:00 4 rows

ERRORS IN SETTING UP DISTRIBUTED TRANSACTIONS:

Error message:



Provider settings:



Linked server settings:

Linked Server Properties - SQLAuth_PG

Select a page

General


Security

Server Options


Connection


Server:
DESKTOP-8AIVGG4

Connection:
DESKTOP-8AIVGG4\KR

 [View connection properties](#)

Progress

 Ready

Script  Help

Linked server:
SQLAUTH_PG

Server type:

SQL Server

Other data source

Provider:
Intellisoft OLE DB Provider for PostgreSQL

Product name:
PGNP

Data source:
localhost

Provider string:
PORT=5432;

Location

Catalog
sqlauthority

OK

Cancel

Linked Server Properties - SQLAuth_PG


Select a page

General


Security


Server Options

Connection

Server:
DESKTOP-8AIVGG4
Connection:
DESKTOP-8AIVGG4\KR
 [View connection properties](#)

Progress

 Ready

Script  Help

Local server login to remote server login mappings:

Local Login	Impersonate	Remote User	Remote Password
-------------	-------------	-------------	-----------------

Add

Remove

For a login not defined in the list above, connections will:

☐ Not be made

☐ Be made without using a security context

☐ Be made using the login's current security context

☒ Be made using this security context:

Remote login:

With password:

postgres

OK

Cancel