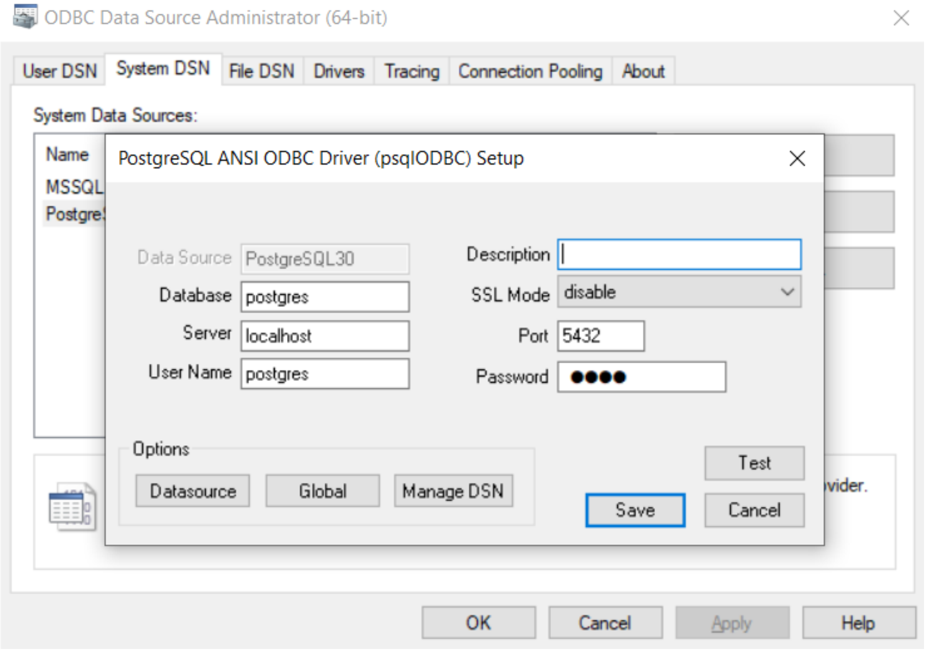
SETTING UP THE LINKED SERVER:

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-



Data Source PostgreSQL30

Database postgres

Server localhost

User Name postgres

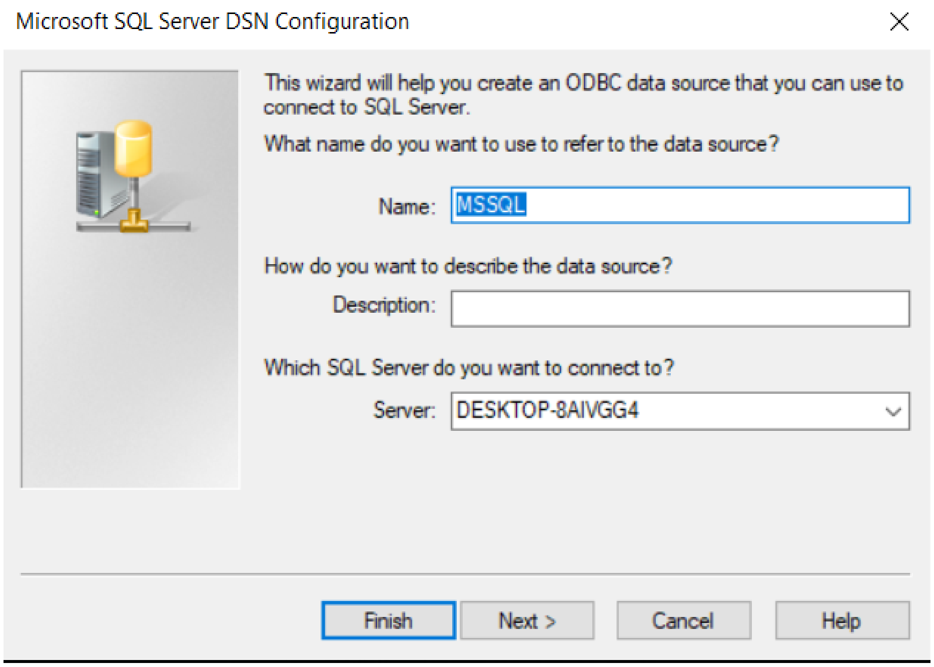
Description

SSL Mode disable

Port 5432

Password ●●●●

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: MSSQL

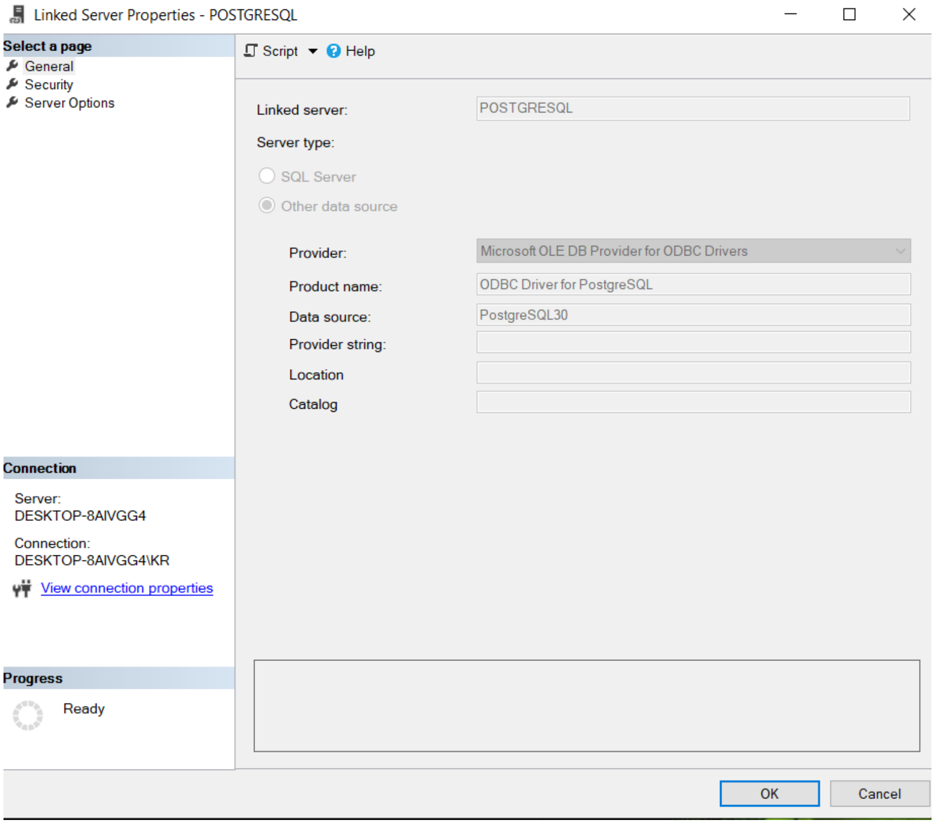
How do you want to describe the data source?

Description:

Which SQL Server do you want to connect to?

Server: DESKTOP-8AIVGG4

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

Linked server: POSTGRESQL

Server type:

○ SQL Server

● Other data source

Provider: Microsoft OLE DB Provider for ODBC Drivers

Product name: ODBC Driver for PostgreSQL

Data source: PostgreSQL30

Provider string:

Location

Catalog

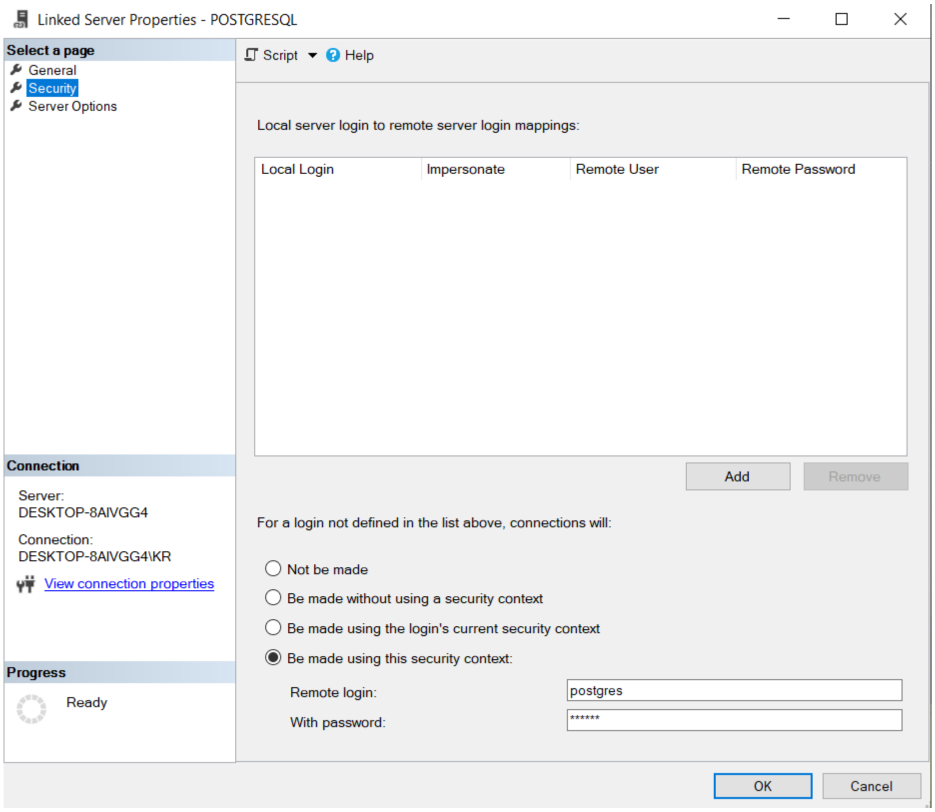
Connection

Server:
DESKTOP-8AIVGG4

Connection:
DESKTOP-8AIVGG4\KR

Progress

Ready



Local server login to remote server login mappings:

Local Login	Impersonate	Remote User	Remote Password

Connection

Server:
DESKTOP-8AIVGG4

Connection:
DESKTOP-8AIVGG4\KR

View connection properties

Progress

Ready

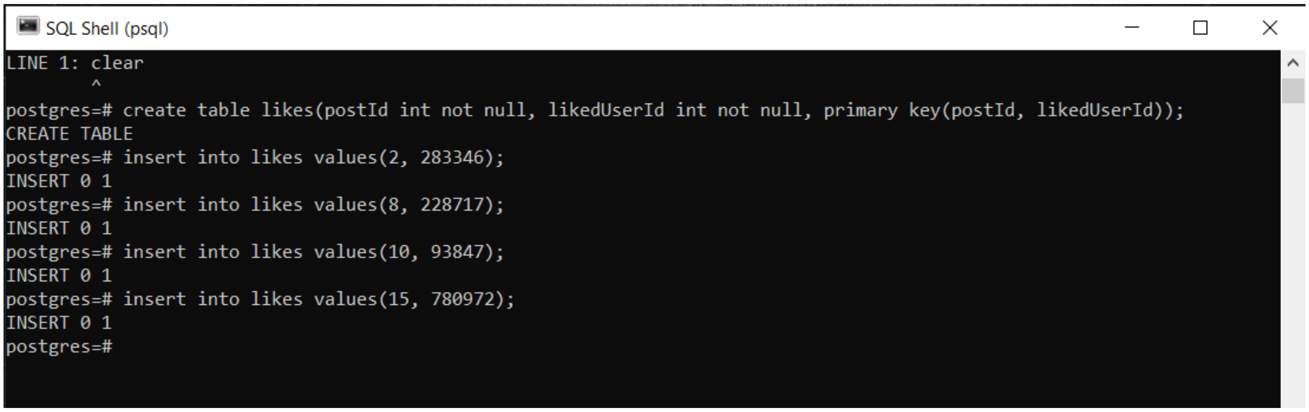
For a login not defined in the list above, connections will:

○ 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: postgres
With password: ******

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

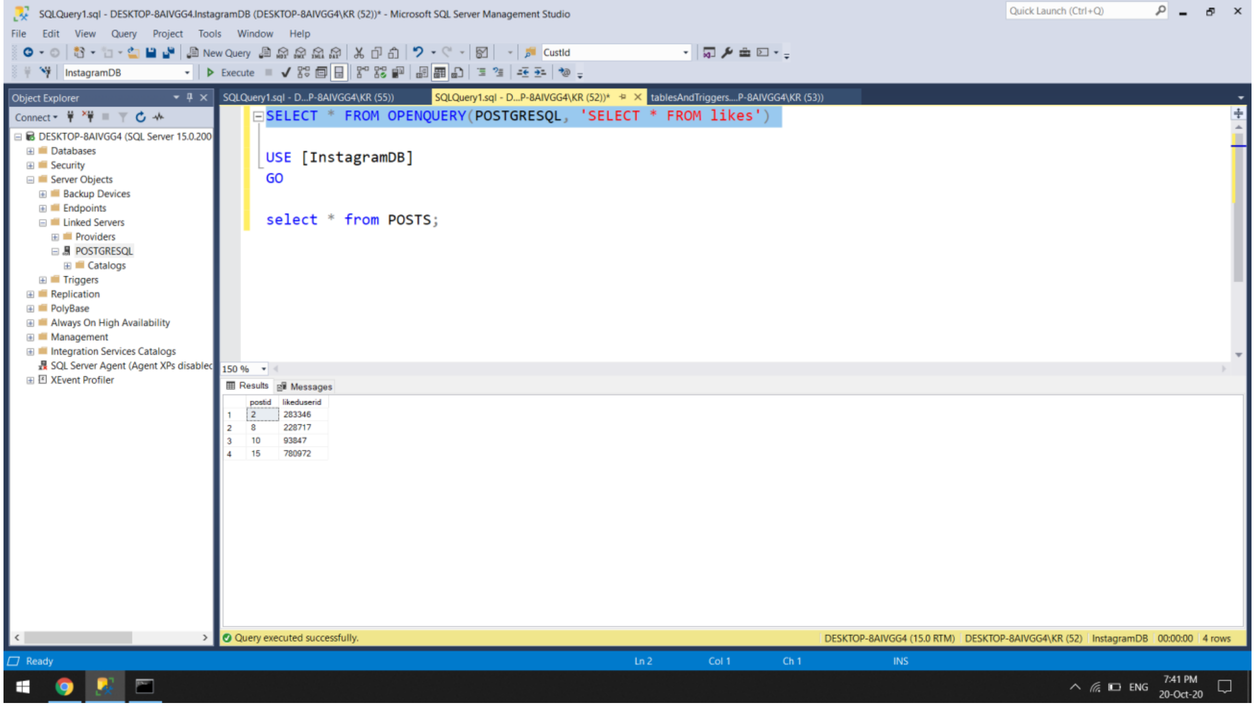
CREATE TABLE LIKES (postId INT NOT NULL, likedUserId INT NOT NULL, PRIMARY KEY(postId, likedUserId);



```
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.


SQLQuery1.sql - DESKTOP-8AIVGG4.InstagramDB (DESKTOP-8AIVGG4\KR (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
InstagramDB
CustId
Object Explorer
Connect
DESKTOP-8AIVGG4 (SQL Server 15.0.200
Databases
Security
Server Objects
Backup Devices
Endpoints
Linked Servers
Providers
POSTGRESQL
Catalogs
Triggers
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled
XEvent Profiler
SQLQuery1.sql - D...P-8AIVGG4\KR (55))
SQLQuery1.sql - D...P-8AIVGG4\KR (52))*
tablesAndTriggers....P-8AIVGG4\KR (53))
150 %
Results
Messages
Ready
Ln 2
Col 1
Ch 1
INS

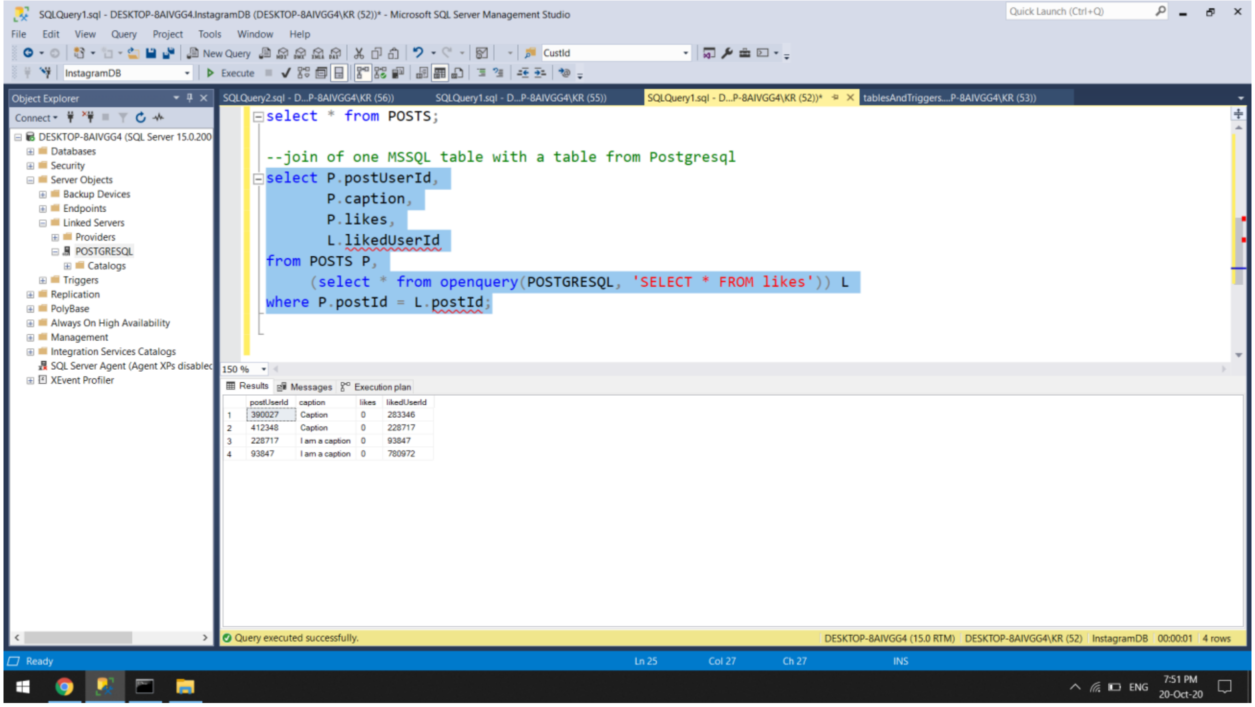
```
SELECT * FROM OPENQUERY(POSTGRESQL, 'SELECT * FROM likes')

USE [InstagramDB]
GO

select * from POSTS;
```

	postid	likeduserid
1	2	283346
2	8	228717
3	10	93847
4	15	780972

```
Query executed successfully.    DESKTOP-8AIVGG4 (15.0 RTM)    DESKTOP-8AIVGG4\KR (52)    InstagramDB    00:00:00    4 rows
```

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;
```



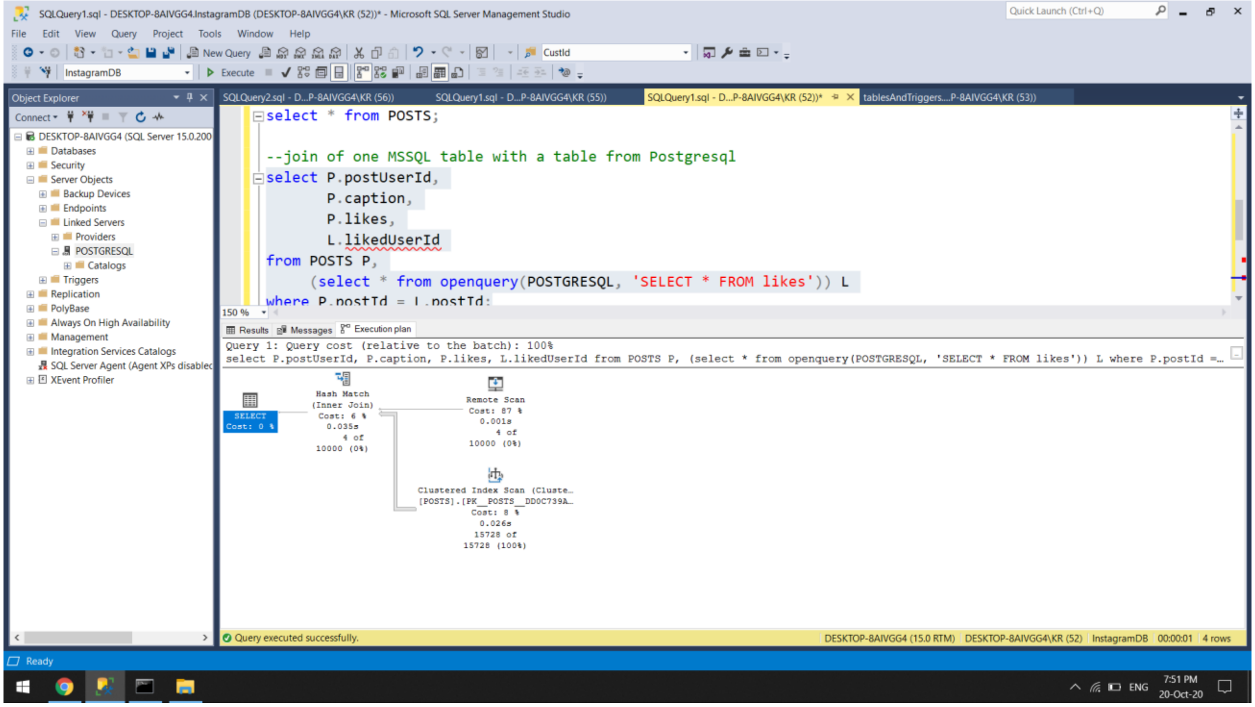
SQLQuery1.sql - DESKTOP-8AIVGG4.InstagramDB (DESKTOP-8AIVGG4\KR (52))* - Microsoft SQL Server Management Studio

```sql
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(POSTGRESQL, 'SELECT * FROM likes')) L
where P.postId = L.postId;
```

	postUserId	caption	likes	likedUserId
1	390027	Caption	0	283346
2	412348	Caption	0	228717
3	228717	I am a caption	0	93847
4	93847	I am a caption	0	780972

Query executed successfully. DESKTOP-8AIVGG4 (15.0 RTM) DESKTOP-8AIVGG4\KR (52) InstagramDB 00:00:01 4 rows

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

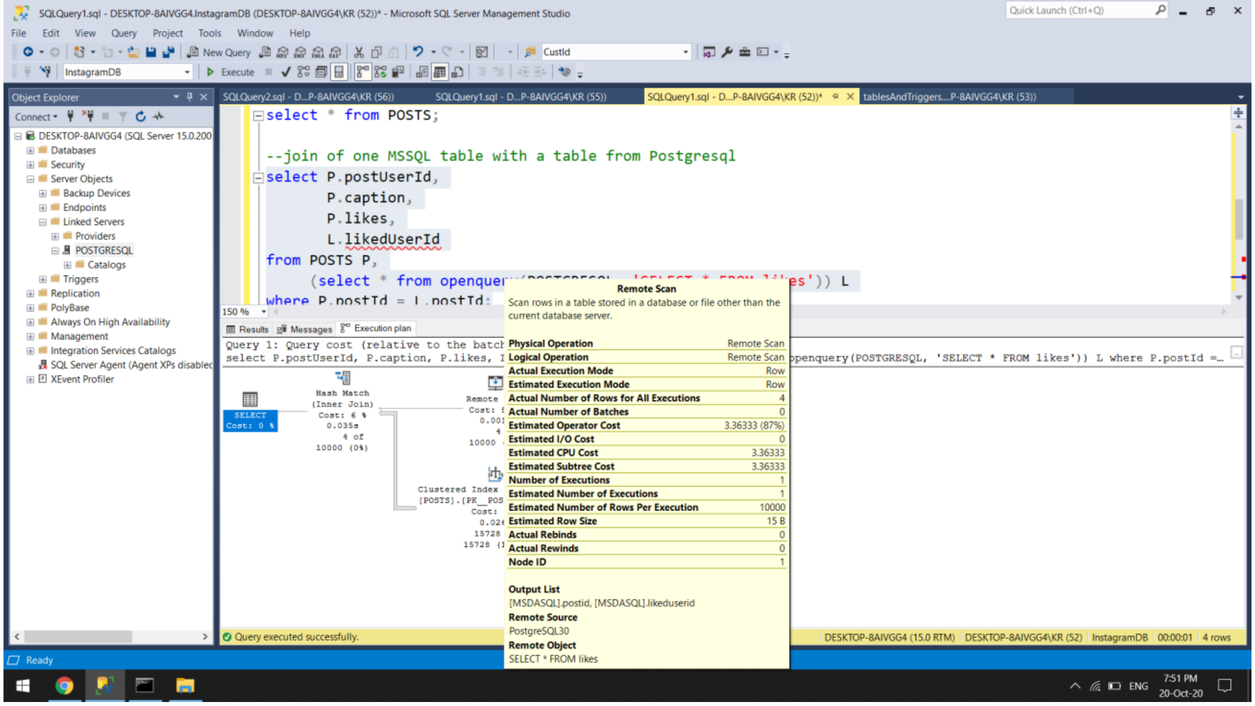
```sql
select * from POSTS;

--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(POSTGRESQL, '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(POSTGRESQL, 'SELECT * FROM likes')) L where P.postId =…
```

Query executed successfully. DESKTOP-8AIVGG4 (15.0 RTM) DESKTOP-8AIVGG4\KR (52) InstagramDB 00:00:01 4 rows

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.



SQLQuery1.sql - DESKTOP-8AIVGG4.InstagramDB (DESKTOP-8AIVGG4\KR (52))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
CustId
InstagramDB
Execute
Object Explorer
Connect
DESKTOP-8AIVGG4 (SQL Server 15.0.200
Databases
Security
Server Objects
Backup Devices
Endpoints
Linked Servers
Providers
POSTGRESQL
Catalogs
Triggers
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disablec
XEvent Profiler
SQLQuery2.sql - D...P-8AIVGG4\KR (56))
SQLQuery1.sql - D...P-8AIVGG4\KR (55))
SQLQuery1.sql - D...P-8AIVGG4\KR (52))*
tablesAndTriggers....P-8AIVGG4\KR (53))
150 %
Results
Messages
Execution plan
SELECT
Cost: 0 %
Hash Match
(Inner Join)
Cost: 6 %
0.035s
4 of
10000 (0%)
Remote
Cost:
0.001
4
10000
Clustered Index
[POSTS].[PK__POS
Cost:
0.02
15728
15728
Query executed successfully.
DESKTOP-8AIVGG4 (15.0 RTM)
DESKTOP-8AIVGG4\KR (52)
InstagramDB
00:00:01
4 rows
Ready
ENG
7:51 PM
20-Oct-20

```
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 openquer                      es')) L
where P.postId = L.postId;
```

```
Query 1: Query cost (relative to the batch
select P.postUserId, P.caption, P.likes, I                    openquery(POSTGRESQL, 'SELECT * FROM likes')) L where P.postId =…
```

Remote Scan

Scan rows in a table stored in a database or file other than the current database server.

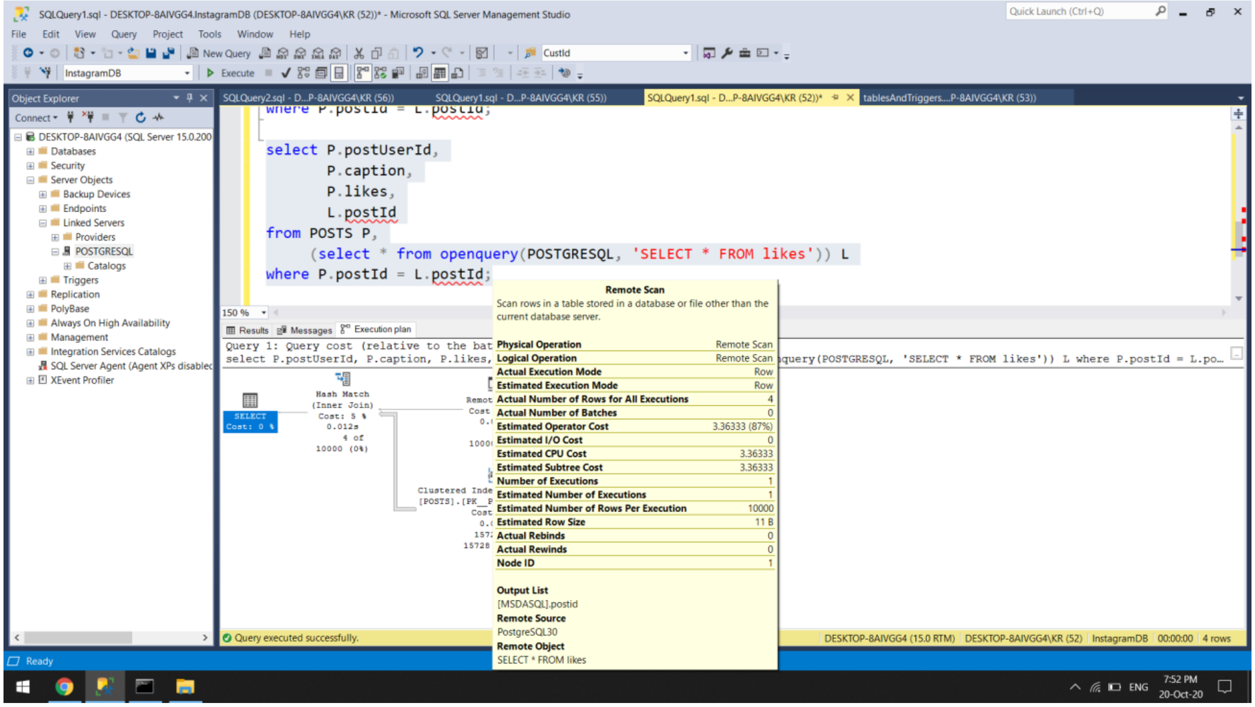
Physical Operation	Remote Scan
Logical Operation	Remote Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows for All Executions	4
Actual Number of Batches	0
Estimated Operator Cost	3.36333 (87%)
Estimated I/O Cost	0
Estimated CPU Cost	3.36333
Estimated Subtree Cost	3.36333
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows Per Execution	10000
Estimated Row Size	15 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	1

Output List
[MSDASQL].postid, [MSDASQL].likeduserid
Remote Source
PostgreSQL30
Remote Object
SELECT * FROM likes

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(POSTGRESQL, '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.



```
where 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;
```

```
Query 1: Query cost (relative to the bat
select P.postUserId, P.caption, P.likes,
```

```
query(POSTGRESQL, 'SELECT * FROM likes')) L where P.postId = L.po...
```

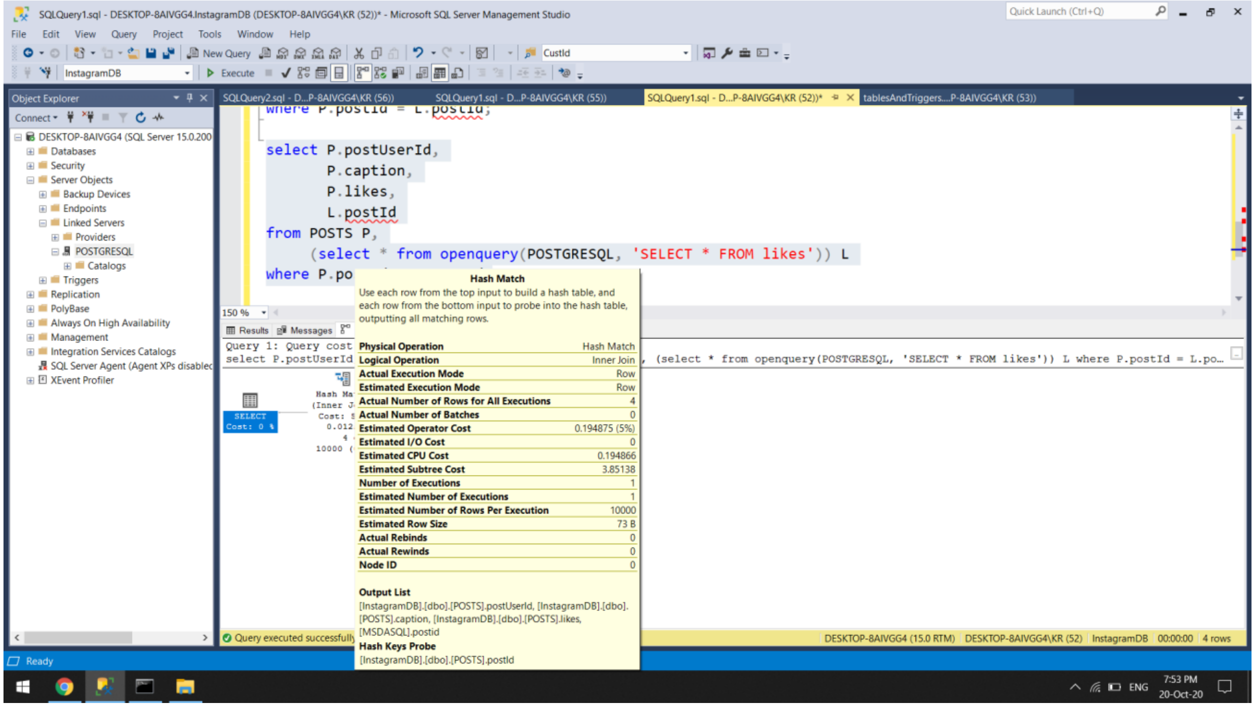
Remote Scan

Scan rows in a table stored in a database or file other than the current database server.

Physical Operation	Remote Scan
Logical Operation	Remote Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows for All Executions	4
Actual Number of Batches	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 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
Quick Launch (Ctrl+Q)
File
Edit
View
Query
Project
Tools
Window
Help
New Query
CustId
InstagramDB
Execute
Object Explorer
Connect
DESKTOP-8AIVGG4 (SQL Server 15.0.200
Databases
Security
Server Objects
Backup Devices
Endpoints
Linked Servers
Providers
POSTGRESQL
Catalogs
Triggers
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disablec
XEvent Profiler
SQLQuery2.sql - D...P-8AIVGG4\KR (56))
SQLQuery1.sql - D...P-8AIVGG4\KR (55))
SQLQuery1.sql - D...P-8AIVGG4\KR (52))*
tablesAndTriggers....P-8AIVGG4\KR (53))
150 %
Results
Messages
SELECT
Cost: 0 %
Hash Ma
(Inner J
Cost: 5
0.012
4
10000 (
Ready
ENG
7:53 PM
20-Oct-20

```
where 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.po
```

```
Query 1: Query cost
select P.postUserId
```

```
, (select * from openquery(POSTGRESQL, 'SELECT * FROM likes')) L where P.postId = L.po...
```

Hash Match

Use each row from the top input to build a hash table, and each row from the bottom input to probe into the hash table, outputting all matching rows.

Physical Operation	Hash Match
Logical Operation	Inner Join
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows for All Executions	4
Actual Number of Batches	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 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