JOINING HETEROGENEOUS DATABASES IS A REALITY, NOT A MYTH

POSTGRESQL-FDW

Ibrar Ahmed
Who am I?

IBRAR AHMED
Senior Software Architect
Percona LLC

Software Career
• Software industries since 1998.

PostgreSQL Career
• Working on PostgreSQL Since 2006.
• EnterpriseDB (Associate Software Architect core Database Engine) 2006-2009
• EnterpriseDB (Software Architect core Database Engine) 2011 - 2016
• EnterpriseDB (Senior Software Architect core Database Engine) 2016 – 2018
• Percona (Senior Software Architect core Database Engine) 2018 – Present

PostgreSQL Books
• PostgreSQL Developer's Guide
• PostgreSQL 9.6 High Performance

@ibrar_ahmad
https://www.facebook.com/ibrar.ahmed
https://www.linkedin.com/in/ibrarahmed74/
https://pgelephant.com/
01 Application Architecture
02 SQL-MED
03 FDW Example
04 Push Down
05 Connection Pooling
06 Questions and Answers
Why? Accessing Data From Multiple Sources

SELECT * from multiple “Database Engines” and generate results?
1. Application Architecture
Application Architecture

MySQL Module
PostgreSQL Module
MongoDB Module
JDBC Module
ODBC Module

libmysqlclient
Libpq
libmongo-c
JDBC
JDBC
JDBC

MySQL
PostgreSQL
MongoDB
JDBC
Hive
ClickHouse
SQL-MED - Management of External Data

- SQL standard, it is defined by ISO/IEC 9075-9:2008
- SQL/MED provides extensions to SQL that define FDW (Foreign Data Wrapper)
- PostgreSQL start implementing in its core since PostgreSQL Version 9.1
- PostgreSQL community builds PostgreSQL FDW called postgresql_fdw

Now there are many FDWs implemented by other people

https://wiki.postgresql.org/wiki/Foreign_data_wrappers
Application Architecture

USER APPLICATION

MySQL Module
PostgreSQL Module
MongoDB Module
Spark Module
Hive Module
ClickHouse Module

POSTGRES SQL

POSTGRES SQL

FDW

mysql_fdw
postgres_fdw
mongo_fdw
hdfs_fdw
file_fdw

libmysqlclient
Libpq
libmongo-c
JDBC
JDBC
JDBC
ODBC

MySQL
PostgreSQL
mongoDB
Spark
Hive
ClickHouse
2. FDW-Example
CREATE EXTENSION mysqldb_fdw;

CREATE SERVER mysql_svr
    FOREIGN DATA WRAPPER mysqldb_fdw
    OPTIONS (host '127.0.0.1',
              port '3306')

CREATE USER MAPPING FOR postgres
    SERVER mysql_svr
    OPTIONS (username 'mysql_user', password 'mysql_pass');

CREATE FOREIGN TABLE mysql_tbl_continents
(  code VARCHAR(2),
  name VARCHAR(255)
) SERVER mysql_svr OPTIONS(dbname 'db');

CREATE FOREIGN TABLE mysql_tbl_countries
(  code VARCHAR(2),
  name VARCHAR(255),
  full_name VARCHAR(255),
  iso3 CHAR(3),
  number INTEGER,
  continent_code VARCHAR(2)
) SERVER mysql_svr OPTIONS (dbname 'db');
CREATE EXTENSION clickhousedb_fdw;

CREATE SERVER clickhouse_svr
    FOREIGN DATA WRAPPER clickhousedb_fdw
    OPTIONS (dbname 'test_database',
               driver '/use/lib/libclickhouseodbc.so');

CREATE USER MAPPING FOR postgres
    SERVER clickhouse_svr
    OPTIONS (username 'clickhouse_user', password 'clickhouse_pass');

CREATE FOREIGN TABLE clickhouse_tbl_onetime(
    Year INTEGER,
    Quarter INTEGER,
    Month INTEGER,
    ...
) SERVER clickhouse_svr OPTIONS (table_name 'ontime');
## SELECT Data From MySQL Using `mysqldb_fdw` 1/2

### Postgres Query Example

```sql
postgres=# SELECT * FROM mysql_tbl_continents;
```

<table>
<thead>
<tr>
<th>code</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>Africa</td>
</tr>
<tr>
<td>AN</td>
<td>Antarctica</td>
</tr>
<tr>
<td>AS</td>
<td>Asia</td>
</tr>
<tr>
<td>EU</td>
<td>Europe</td>
</tr>
<tr>
<td>NA</td>
<td>North America</td>
</tr>
<tr>
<td>OC</td>
<td>Oceania</td>
</tr>
<tr>
<td>SA</td>
<td>South America</td>
</tr>
</tbody>
</table>

(7 rows)

### Postgres Query Example

```sql
postgres=# SELECT code, name, continent_code FROM mysql_tbl_countries LIMIT 7;
```

<table>
<thead>
<tr>
<th>code</th>
<th>name</th>
<th>continent_code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>Andorra</td>
<td>EU</td>
</tr>
<tr>
<td>AE</td>
<td>United Arab Emirates</td>
<td>AS</td>
</tr>
<tr>
<td>AF</td>
<td>Afghanistan</td>
<td>AS</td>
</tr>
<tr>
<td>AG</td>
<td>Antigua and Barbuda</td>
<td>NA</td>
</tr>
<tr>
<td>AI</td>
<td>Anguilla</td>
<td>NA</td>
</tr>
<tr>
<td>AL</td>
<td>Albania</td>
<td>EU</td>
</tr>
<tr>
<td>AM</td>
<td>Armenia</td>
<td>AS</td>
</tr>
</tbody>
</table>

(7 rows)

Data comes from MySQL Database

---

Same table name exists in MySQL

---
### SELECT Data From MySQL Using mysqldb_fdw 2/2

```sql
postgres=# SELECT country.code, country.name, continent.name  
     FROM mysql_tbl_continents continent, mysql_tbl_countries country  
WHERE continent.code = country.continent_code LIMIT 3;
```

<table>
<thead>
<tr>
<th>code</th>
<th>name</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO</td>
<td>Angola</td>
<td>Africa</td>
</tr>
<tr>
<td>BF</td>
<td>Burkina Faso</td>
<td>Africa</td>
</tr>
<tr>
<td>BI</td>
<td>Burundi</td>
<td>Africa</td>
</tr>
</tbody>
</table>

(3 rows)  
Country name comes from mysql_tbl_countries table
```
postgres=# SELECT a."Year", c1/c2 as value 
   FROM 
   (SELECT "Year", count(*)*1000 as c1 
    FROM clickhouse_tbl_ontime 
    WHERE "DepDelay">10 GROUP BY "Year") a 
   INNER JOIN 
   (SELECT "Year", count(*) as c2 
    FROM clickhouse_tbl_ontime GROUP BY "Year" ) b 
   ON a."Year"=b."Year" LIMIT 3;

<table>
<thead>
<tr>
<th>Year</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>199</td>
</tr>
<tr>
<td>1988</td>
<td>654182000</td>
</tr>
</tbody>
</table>
(2 rows)
```
### Join ClickHouse, MySQL and PostgreSQL Using FDW

```sql
postgres=# SELECT "Year", pg.code, "OriginStateName", pg.country_code, my.name
FROM clickhouse_tbl_ontime ch
LEFT JOIN pg_tbl_states pg
ON pg.name = ch."OriginStateName"
LEFT JOIN mysql_tbl_countries my
ON pg.country_code = my.code
LIMIT 3;
```

<table>
<thead>
<tr>
<th>Year</th>
<th>code</th>
<th>OriginStateName</th>
<th>country_code</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>MO</td>
<td>Missouri</td>
<td>US</td>
<td>United States of America</td>
</tr>
<tr>
<td>2011</td>
<td>MO</td>
<td>Missouri</td>
<td>US</td>
<td>United States of America</td>
</tr>
<tr>
<td>2011</td>
<td>MO</td>
<td>Missouri</td>
<td>US</td>
<td>United States of America</td>
</tr>
</tbody>
</table>

(3 rows)
EXPLAIN: Join ClickHouse, MySQL and PostgreSQL

postgres=# EXPLAIN VERBOSE
SELECT "Year", pg.code, "OriginStateName", pg.country_code, my.name
FROM clickhouse_tbl_ontime ch
LEFT JOIN pg_tbl_states pg ON pg.name = ch."OriginStateName"
LEFT JOIN mysql_tbl_countries my ON pg.country_code = my.code limit 3;

QUERY PLAN

-> Hash Right Join (cost=10.00..1900.21 rows=5000 width=558)
  Hash Cond: ((pg.name)::text = ch."OriginStateName")
  -> Nested Loop Left Join (cost=10.00..1899.09 rows=295 width=532)
    Join Filter: ((pg.country_code)::text = (my.code)::text)
    -> Seq Scan on public.pg_tbl_states pg (cost=0.00..1.59 rows=59 width=16)
    -> Materialize (cost=10.00..1015.00 rows=1000 width=528)
    -> Foreign Scan on public.mysql_tbl_countries my
       (cost=10.00..1010.00 rows=1000 width=528)
  Remote query: SELECT `code`, `name` FROM `db`.`mysql_tbl_countries`
-> Hash (cost=0.00..0.00 rows=0 width=36)
  -> Foreign Scan on public.clickhouse_tbl_ontime ch
     (cost=0.00..0.00 rows=0 width=36)
Output: ch."Year", ch."OriginStateName"

Remote SQL: SELECT "Year", "OriginStateName" FROM "default".ontime
3. Push Down
Push Down – A Performance Feature

- Operator and function push down
- Predicate push down
- Aggregate push down
- Join push down
PostgreSQL Foreign Data Wrapper - JOIN Push Down

postgres=# EXPLAIN (VERBOSE, COST off)
SELECT * FROM postgres_tbl_name n
RIGHT JOIN postgres_tbl_job j
ON (j.name_id > n.id);

QUERY PLAN

Foreign Scan
Output: n.id, n.name, j.id, j.job_title, j.name_id
Relations: (public.postgres_tbl_job j)
LEFT JOIN (public.postgres_tbl_name n)

Remote SQL: SELECT r2.id, r2.job_title, r2.name_id, r1.id, r1.name
FROM (public.postgres_tbl_job r2
LEFT JOIN public.postgres_tbl_name r1
ON     (((r2.name_id > r1.id))))
(4 rows)
```
postgres=# EXPLAIN VERBOSE SELECT count(*) FROM postgres_tbl_name;
QUERY PLAN
-------------------------------------------------------------
Foreign Scan  (cost=108.53..152.69 rows=1 width=8)
 Output: (count(*))
 Relations: Aggregate on (public.postgres_tbl_name)
 Remote SQL: SELECT count(*) FROM public.postgres_tbl_name
(4 rows)
```

```
postgres=# EXPLAIN VERBOSE SELECT count(*) FROM mysql_tbl_continents;
QUERY PLAN
--------------------------------------------------------------------------------------------
Aggregate  (cost=1012.50..1012.51 rows=1 width=8)
 Output: count(*)
->  Foreign Scan on public.mysql_tbl_continents  (cost=10.00..1010.00 rows=1000 width=0)
   Output: continent_id, continent_name
   Local server startup cost: 10
   Remote query: SELECT NULL FROM `db`.`mysql_tbl_continents`
(6 rows)
```
4. Connection Pooling
Connections 1/2

1. PostgreSQL Query
2. MySQL Query
3. Connect to MySQL
4. MySQL Query
5. Results
6. Disconnect
7. Results, converted to PostgreSQL' tuples
8. Results tuples

Do we really need to Disconnect / Connect on each query?
1. PostgreSQL Query

2. MySQL Query

3.1 Find Connection
3.2 Failed to find - Connect to MySQL
4. MySQL Query

5. Results

6. Results, (PostgreSQL tuples)

7. Results tuples
DML Support

- PostgreSQL has DML support
- There are several Foreign Data Wrappers that support DML such as:
  - postgres_fdw
  - mysql_fdw
  - oracle_fdw
“Poor leaders rarely ask questions of themselves or others. Good leaders, on the other hand, ask many questions. Great leaders ask the great questions.”

Michael Marquardt, author of Leading with Questions