

JOINING HETEROGENEOUS
DATABASES IS A REALITY, NOT A MYTH



PERCONA

TECH DAYS

- Ibrar Ahmed
- Senior Database Architect - Percona LLC
- August 13 –2020

WHY? ACCESSING DATA FROM MULTIPLE SOURCES

SELECT * from multiple “Database Engines” and generate results?



APPLICATION ARCHITECTURE 1/2



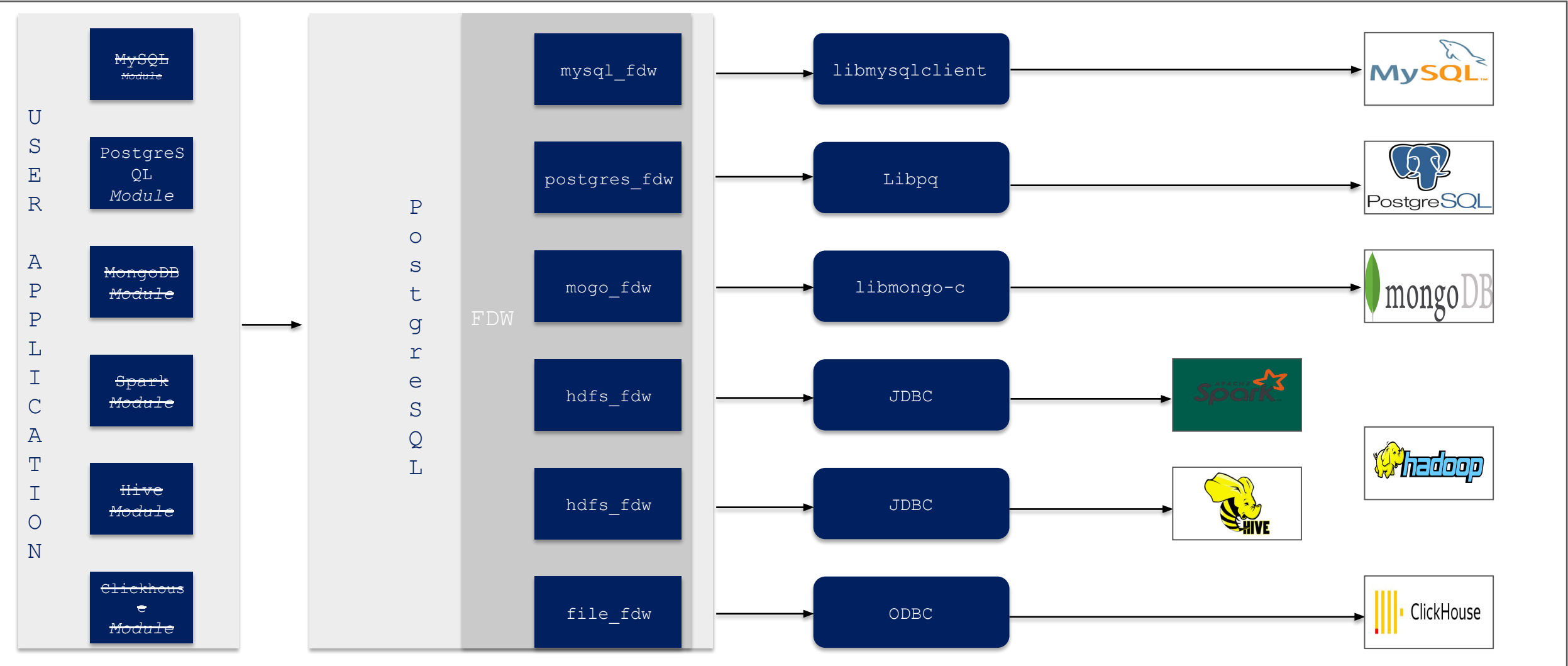
SQL-MED - MANAGEMENT OF EXTERNAL DATA

- 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 2/2



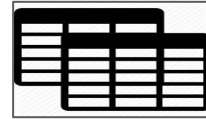
EXAMPLE

US States / Cities



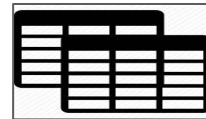
`pg_tbl_states`

Countries / Country



`mysql_tbl_continents`
`mysql_tbl_countries`

Flight Information



`clickhouse_tbl_ontime`

SETUP: MYSQLDB_FDW (MYSQL)

```
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');
```

SETUP: CLICKHOUSEDB_FDW (CLICKHOUSE)

```
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_ontime(  
  Year      INTEGER,  
  Quarter  INTEGER,  
  Month     INTEGER,  
  ...  
) SERVER clickhouse_svr OPTIONS (table_name 'ontime');
```

SELECT DATA FROM MYSQL USING MYSQLDB_FDW 1/2

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

Same table name exists in MySQL

code	name
AF	Africa
AN	Antarctica
AS	Asia
EU	Europe
NA	North America
OC	Oceania
SA	South America

(7 rows)

Data comes from MySQL Database

```
postgres=# SELECT code, name, continent_code
```

```
FROM mysql_tbl_countries LIMIT 7;
```

code	name	continent_code
AD	Andorra	EU
AE	United Arab Emirates	AS
AF	Afghanistan	AS
AG	Antigua and Barbuda	NA
AI	Anguilla	NA
AL	Albania	EU
AM	Armenia	AS

(7 rows)

SELECT DATA FROM MYSQL USING MYSQLDB_FDW 2/2

```
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;
```

code	name	name
AO	Angola	Africa
BF	Burkina Faso	Africa
BI	Burundi	Africa

(3 rows)

Country name comes from mysql_tbl_countries table

SELECT DATA FROM CLICKHOUSE USING CLICKHOUSEDB_FDW

```
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;
```

Year	 	value
1987		199
1988		654182000

(2 rows)

JOIN CLICKHOUSE, MYSQL AND POSTGRESQL USING FDW

```
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;
```

Year	code	OriginStateName	country_code	name
2011	MO	Missouri	US	United States of America
2011	MO	Missouri	US	United States of America
2011	MO	Missouri	US	United States of America

(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
```

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)

POSTGRESQL FOREIGN DATA WRAPPER - AGGREGATE PUSH DOWN

```
postgres=# EXPLAIN VERBOSE SELECT count(*) FROM postgres_tbl_name;
```

QUERY PLAN

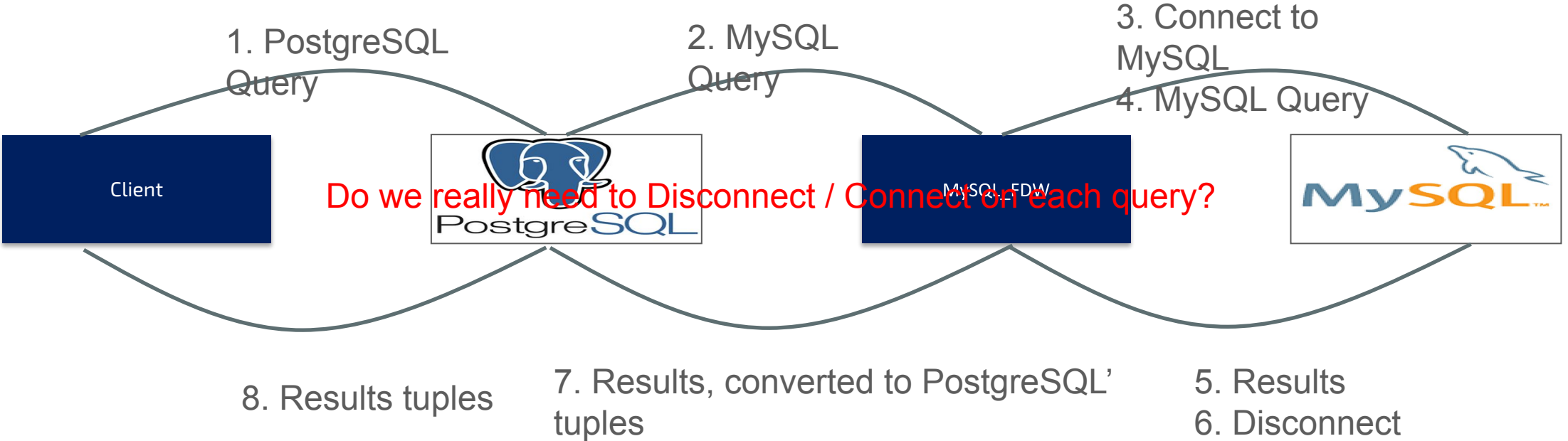
```
Foreign Scan (cost=108.53..152.69 rows=1 width=8)
  Output: (count(*))
  Relations: Aggregate on (public.postgres_tbl_name)
  Remote SQL: SELECT count(*) FROM public.postgres_tbl_name
(4 rows)
```

```
postgres=# EXPLAIN VERBOSE SELECT count(*) FROM mysql_tbl_continents;
```

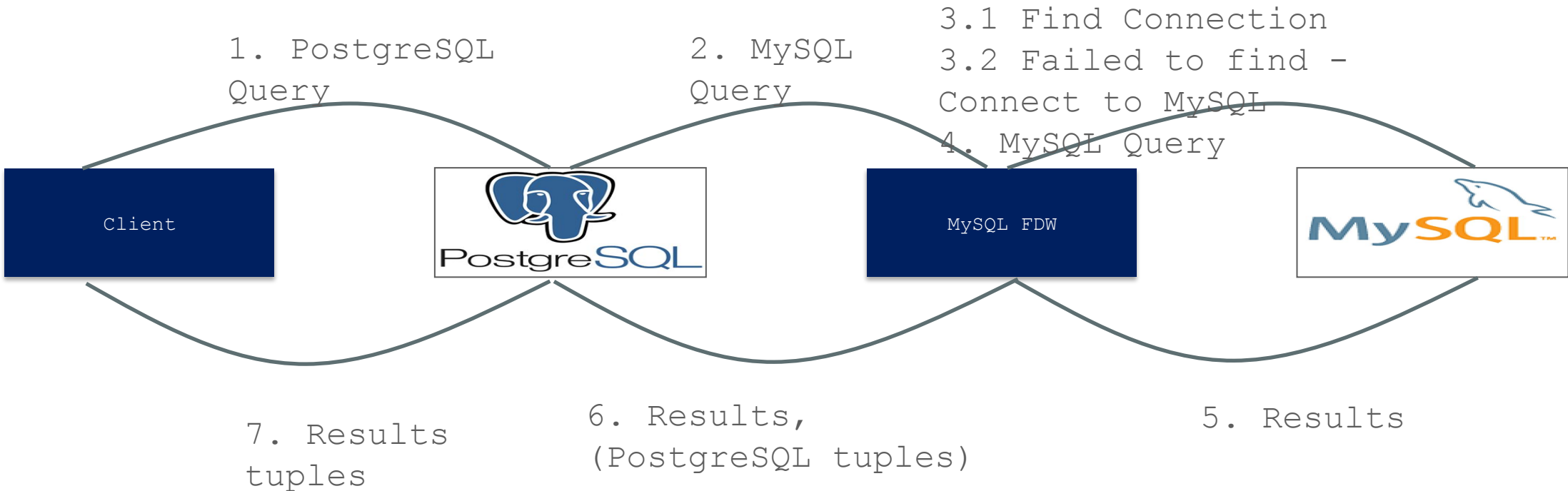
QUERY PLAN

```
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)
```

CONNECTIONS 1/2



CONNECTIONS 2/2



DML SUPPORT

- PostgreSQL has DML support
- There are several Foreign Data Wrappers that support DML such as:
 - postgres_fdw
 - mysql_fdw
 - oracle_fdw

QUESTIONS?

“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*





PERCONA

LIVE ONLINE

20-21 OCTOBER

2020