2ndQuadrant®
PostgreSQL

Foreign Data Wrappers and their utilization in real world scenarios

Boriss Mejías
Consultant - 2ndQuadrant
Air Guitar Player

The Planet of Krikkit

Planet PostgreSQL

Planet PostgreSQL in the Real World

- You can’t always migrate to PostgreSQL

Planet PostgreSQL in the Real World

- You can’t always migrate to PostgreSQL
- Sometimes you don’t want to migrate

Planet PostgreSQL in the Real World

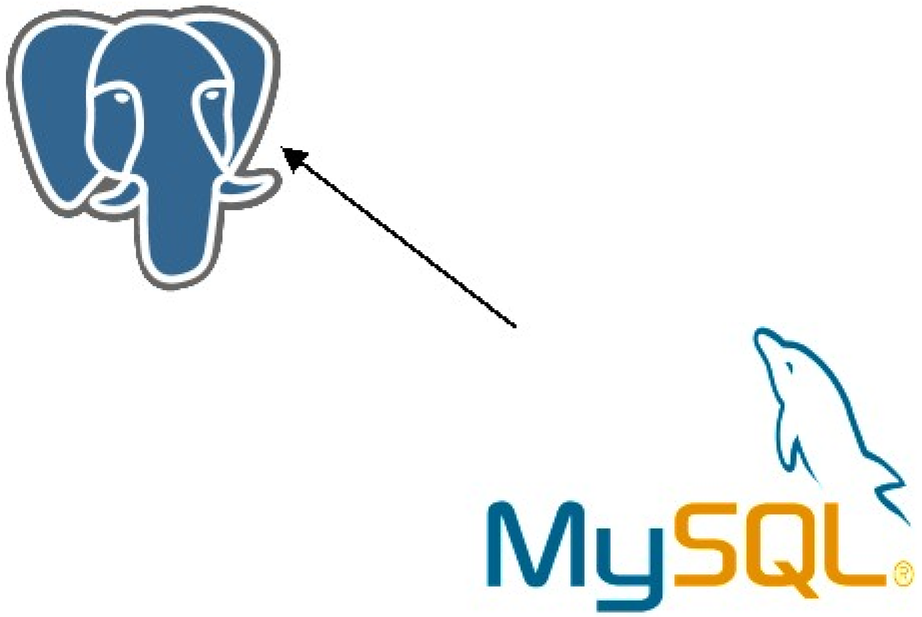
- You can’t always migrate to PostgreSQL
- Sometimes you don’t want to migrate
- The other system might be the right tool

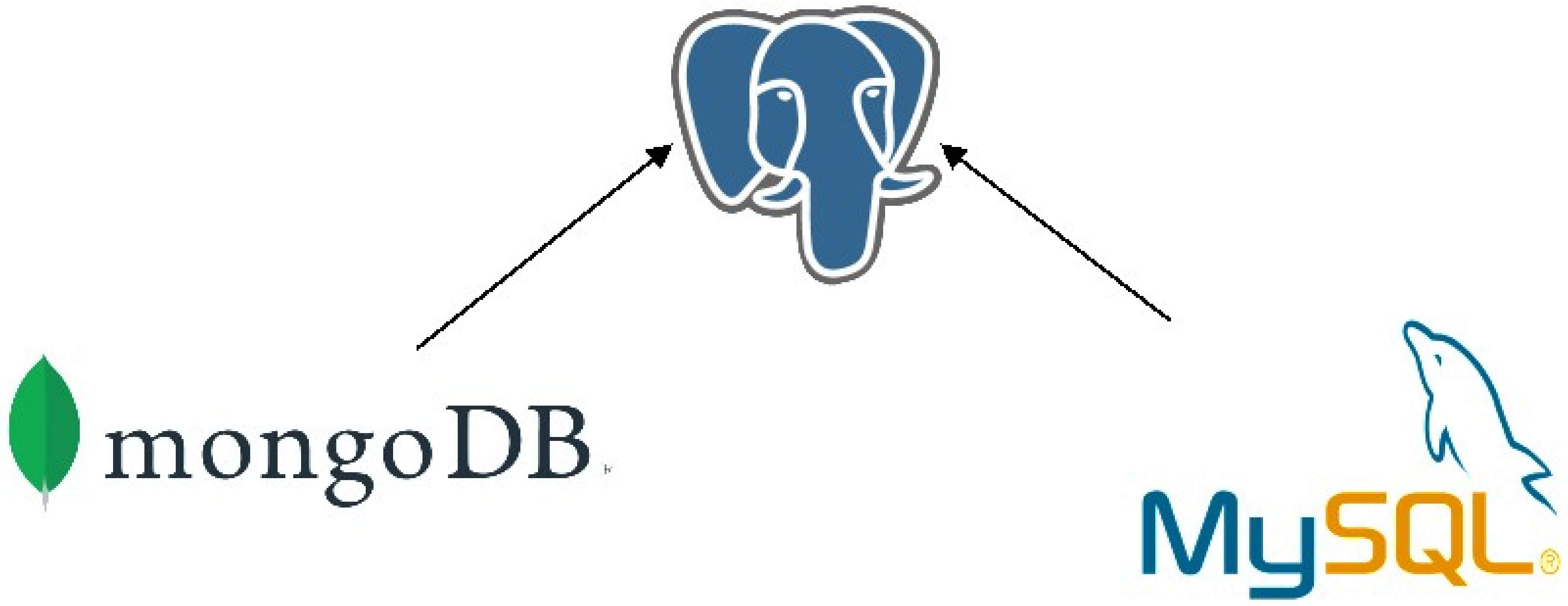
Planet PostgreSQL in the Real World

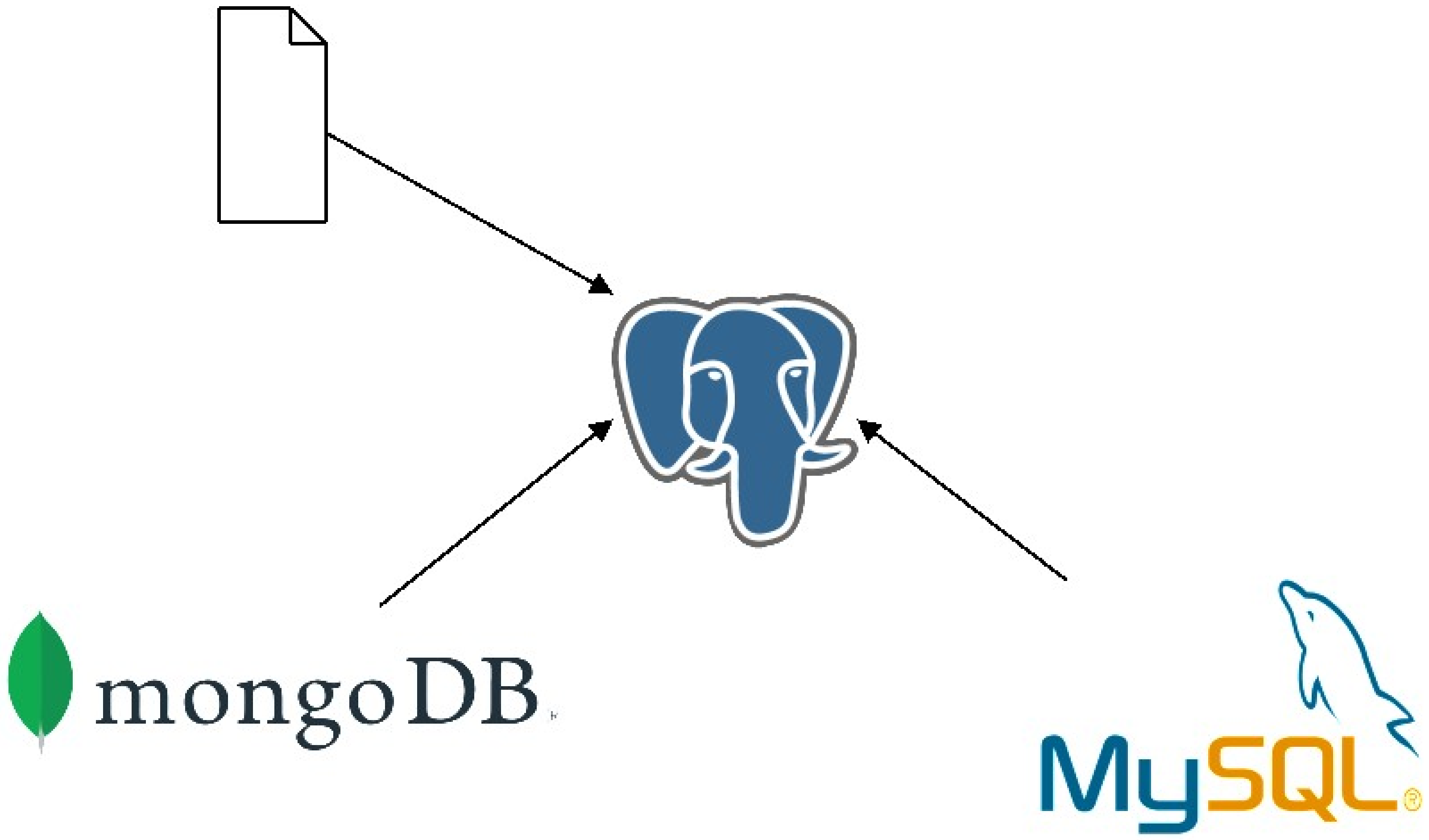
- You can’t always migrate to PostgreSQL
- Sometimes you don’t want to migrate
- The other system might be the right tool
- Data Integration from different departments/companies/software

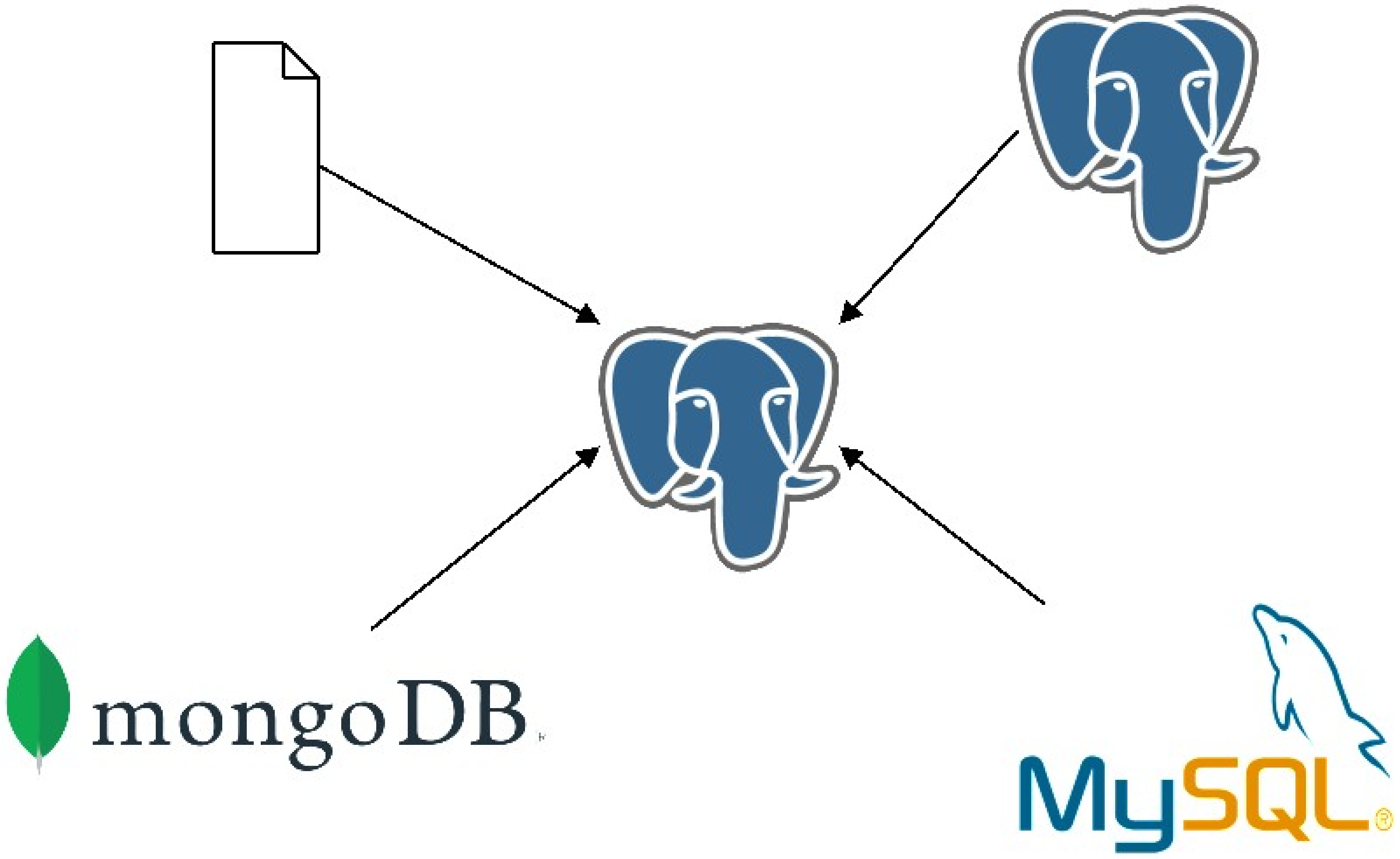
Planet PostgreSQL in the Real World

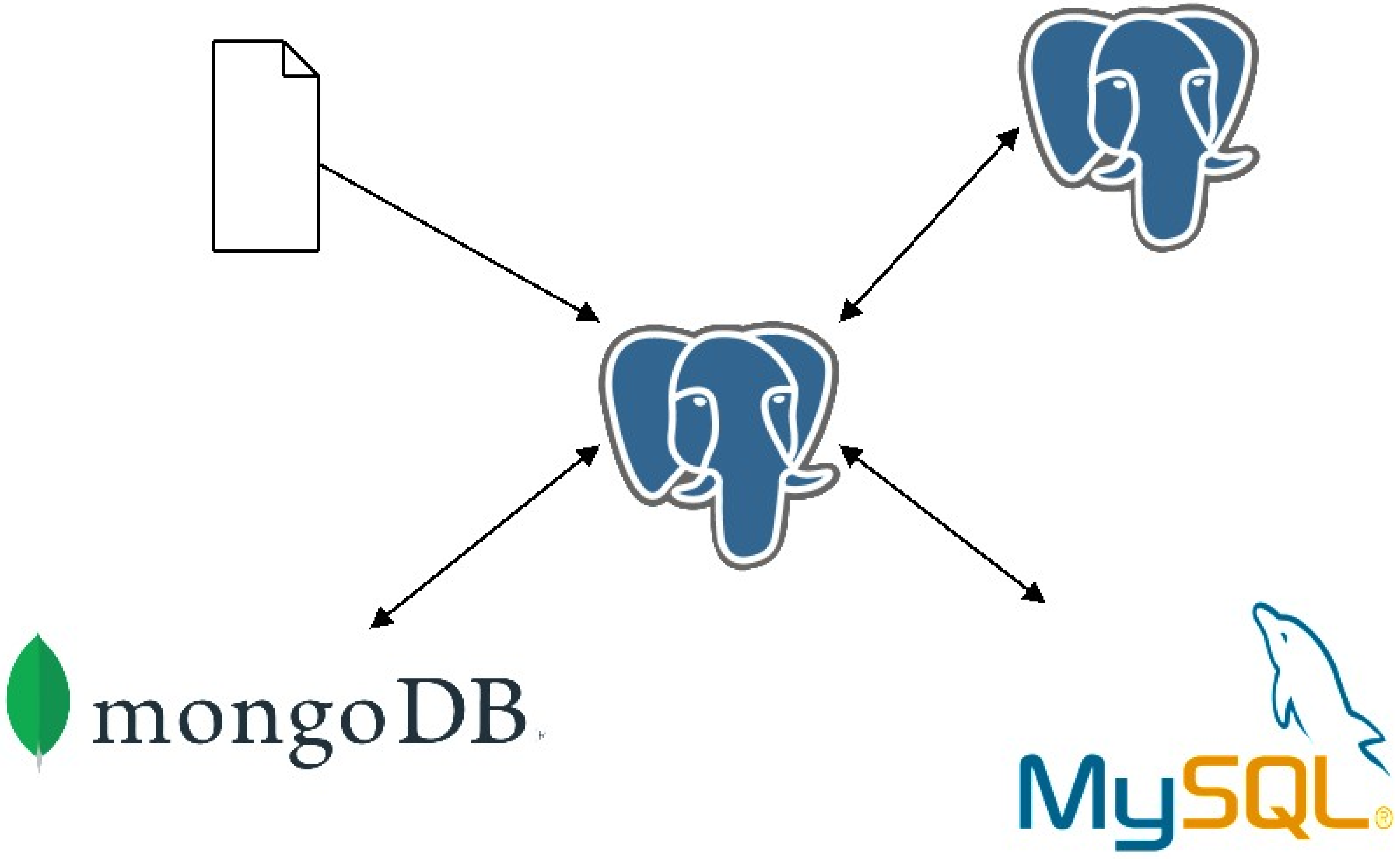
- You can’t always migrate to PostgreSQL
- Sometimes you don’t want to migrate
- The other system might be the right tool
- Data Integration from different departments/companies/software
- Avant Garde


MySQL


mongoDB
MySQL


mongoDB
MySQL


mongoDB
MySQL


mongoDB
MySQL

mongoDB
MySQL
MySQL

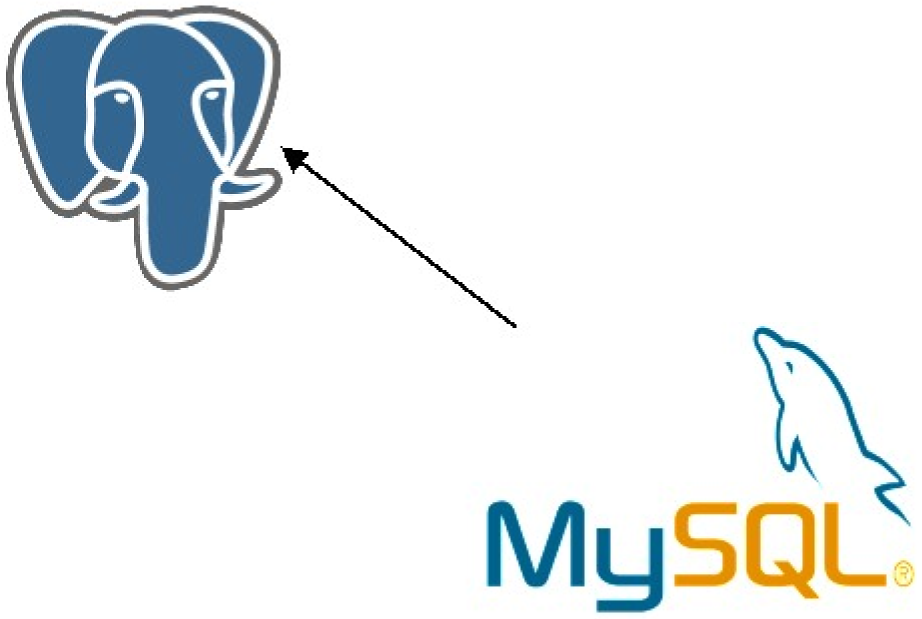
Postgres Setup

```
shared_preload_libraries = 'mongo_fdw, mysql_fdw'
```

- And install software

```
sudo apt install postgresql-plpython-11
sudo apt install postgresql-11-mysql-fdw
```

compile `mongo_fwd`


MySQL

MySQL/MariaDB

```
CREATE DATABASE mypgconfeu;
CREATE USER 'milanese'@'%';
GRANT ALL ON mypgconfeu.* TO 'milanese'@'%';

CREATE TABLE hitchhikers (
    id         INTEGER PRIMARY KEY AUTO_INCREMENT,
    hitchhiker TEXT,
    last_seen  TIMESTAMP
);
```

MySQL/MariaDB

```
INSERT INTO hitchhikers (hitchhiker)
     VALUES ('Ford Prefect');

INSERT INTO hitchhikers (hitchhiker)

     VALUES ('Zaphod Beeblebrox');
```

MySQL FDW - Setup

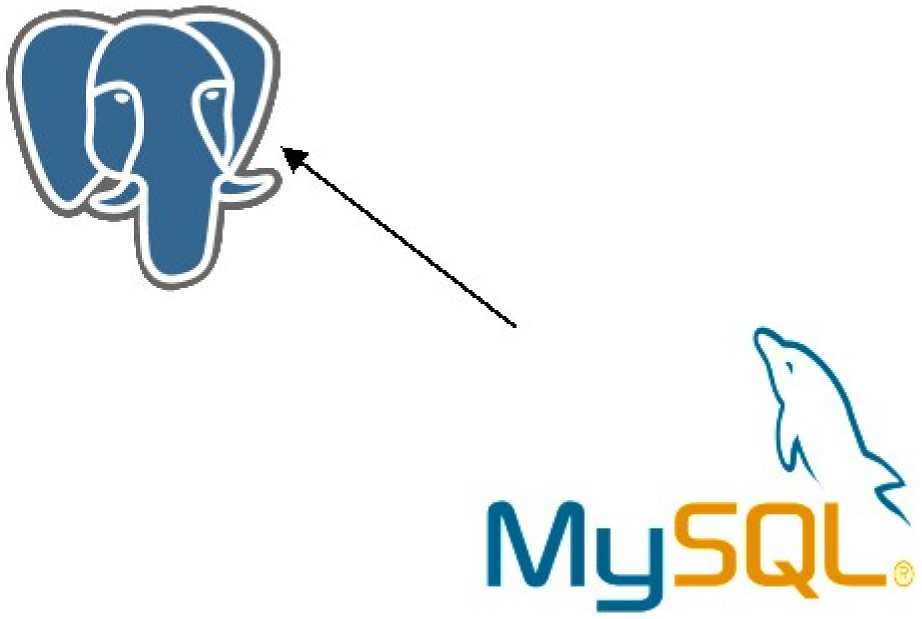
```
CREATE EXTENSION mysql_fdw;

CREATE SERVER mysql_pgconfeu
    FOREIGN DATA WRAPPER mysql_fdw
    OPTIONS (host 'localhost');

CREATE USER MAPPING FOR douglas
    SERVER mysql_pgconfeu
    OPTIONS (username 'milanese'
           , password 'cappuccino');
```

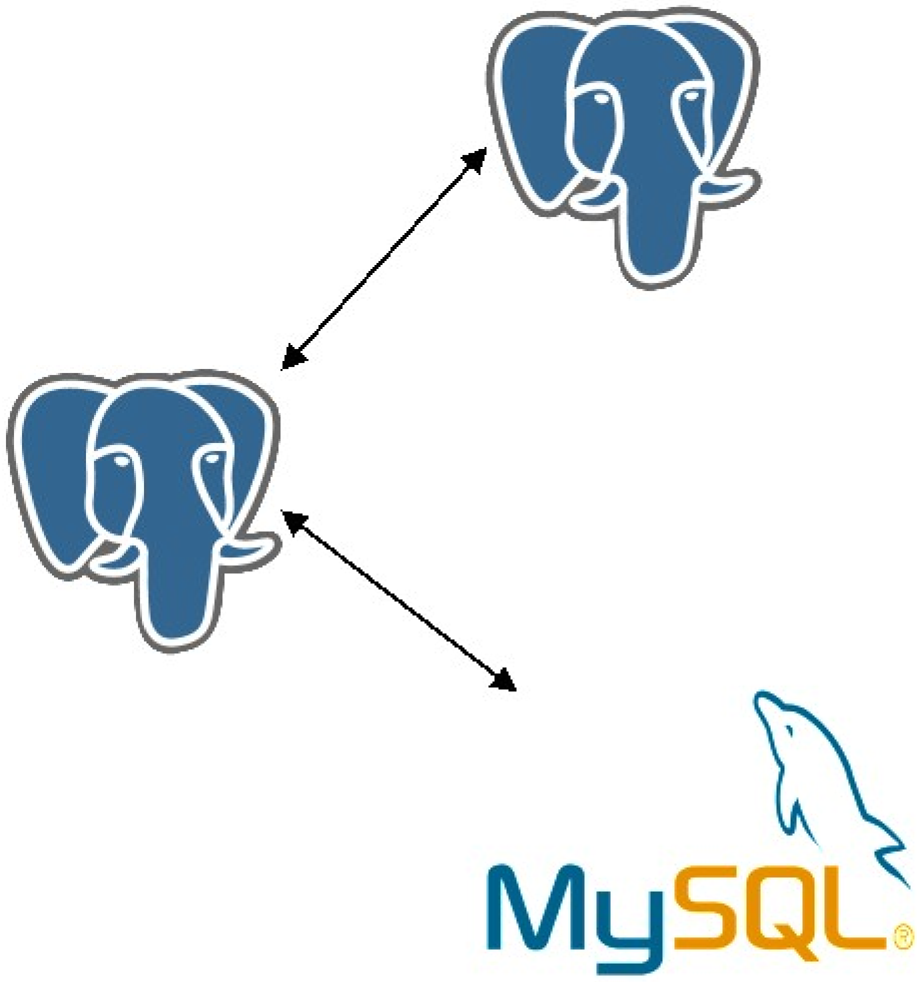
MySQL FDW – Import Schema

```
CREATE SCHEMA mysql;

IMPORT FOREIGN SCHEMA mypgconfeu
    LIMIT TO (hitchhikers)
    FROM SERVER mysql_pgconfeu
    INTO mysql;
```

MySQL FDW – Read and Writes

```
SELECT * FROM mysql.hitchhikers;

INSERT INTO mysql.hitchhikers (hitchhiker)
     VALUES ('Arthur Dent')

SELECT * FROM mysql.hitchhikers;
```


MySQL


MySQL

Another PostgreSQL

```
CREATE USER milanese;
CREATE DATABASE theguide OWNER milanese;

CREATE TABLE hitchhikers (
    id          SERIAL PRIMARY KEY,
    hitchhiker  TEXT,
    last_seen   TIMESTAMP DEFAULT current_timestamp
);
```

Another PostgreSQL

```
INSERT INTO hitchhikers (hitchhiker)
     VALUES ('Trillian');
INSERT INTO hitchhikers (hitchhiker)
     VALUES ('Marvin');
```

PostgreSQL FDW - Setup

```
CREATE EXTENSION postgres_fdw;

CREATE SERVER planet_postgresql
    FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS (dbname 'theguide'
           , host 'localhost'
           , port '5666');

CREATE USER MAPPING FOR douglas
    SERVER planet_postgresql
    OPTIONS (USER 'milanese');
```

PostgreSQL FDW – Import Schema

```
CREATE SCHEMA pgsql;

IMPORT FOREIGN SCHEMA PUBLIC
    LIMIT TO (hitchhikers)
    FROM SERVER planet_postgresql
    INTO pgsql;
```

PostgreSQL FDW – Read and Write

```
SELECT * FROM pgsql.hitchhikers;

INSERT INTO pgsql.hitchhikers
VALUES (3, 'Slartibartfast', now());

SELECT * FROM pgsql.hitchhikers;
```

Statistical Anomaly

```
EXPLAIN SELECT * FROM pgsql.hitchhikers;

                    QUERY PLAN
----------------------------------------------------------------
 Foreign Scan on hitchhikers  (cost=100.00..146.12 rows=1204 width=44)


ANALYZE pgsql.hitchhikers;

EXPLAIN SELECT * FROM pgsql.hitchhikers;

                    QUERY PLAN
----------------------------------------------------------------
 Foreign Scan on hitchhikers  (cost=100.00..101.09 rows=3 width=20)
```

Statistical Anomaly

```
EXPLAIN SELECT * FROM pgsql.hitchhikers;

                    QUERY PLAN
----------------------------------------------------------------------
 Foreign Scan on hitchhikers  (cost=100.00..146.12 rows=1204 width=44)


ANALYZE pgsql.hitchhikers;

EXPLAIN SELECT * FROM pgsql.hitchhikers;

                    QUERY PLAN
----------------------------------------------------------------------
 Foreign Scan on hitchhikers  (cost=100.00..101.09 rows=3 width=20)
```

Let’s add more tables - location

```
CREATE TABLE location (
  id            INT PRIMARY KEY,
  location_name VARCHAR NOT NULL
);

INSERT INTO location (id, location_name)
    SELECT s.id, 'Location ' || s.id::TEXT
      FROM generate_series(1, 1000) s(id);

ANALYZE location;
```

Let’s add more tables – sensor log

```
CREATE TABLE sensor_log (
  id             INT PRIMARY KEY,
  location_id    INT NOT NULL,
  reading        BIGINT NOT NULL,
  reading_date   TIMESTAMP NOT NULL
);

INSERT INTO sensor_log (id, location_id,
                        reading, reading_date)
    SELECT s.id, s.id % 1000, s.id % 100,
       CURRENT_DATE - ((s.id * 10) || 's')::INTERVAL
    FROM generate_series(1, 50000) s(id);
```

2ndQuadrant® PostgreSQL

Let’s add more tables – and indexes

```
CREATE INDEX idx_sensor_log_location
          ON sensor_log (location_id);
CREATE INDEX idx_sensor_log_date
          ON sensor_log (reading_date);

ANALYZE sensor_log;
```

PostgreSQL FDW – Import new tables

```
IMPORT FOREIGN SCHEMA PUBLIC
    LIMIT TO (location, sensor_log)
    FROM SERVER planet_postgresql
    INTO pgsql;

ANALYZE pgsql.location;
ANALYZE pgsql.sensor_log;
```

Let's do a JOIN

```
EXPLAIN
SELECT l.location_name, s.reading
    FROM pgsql.sensor_log s
    JOIN pgsql.location l ON (l.id = s.location_id)
    WHERE s.reading_date >= '2019-10-2';
```

Let's do a JOIN on the source

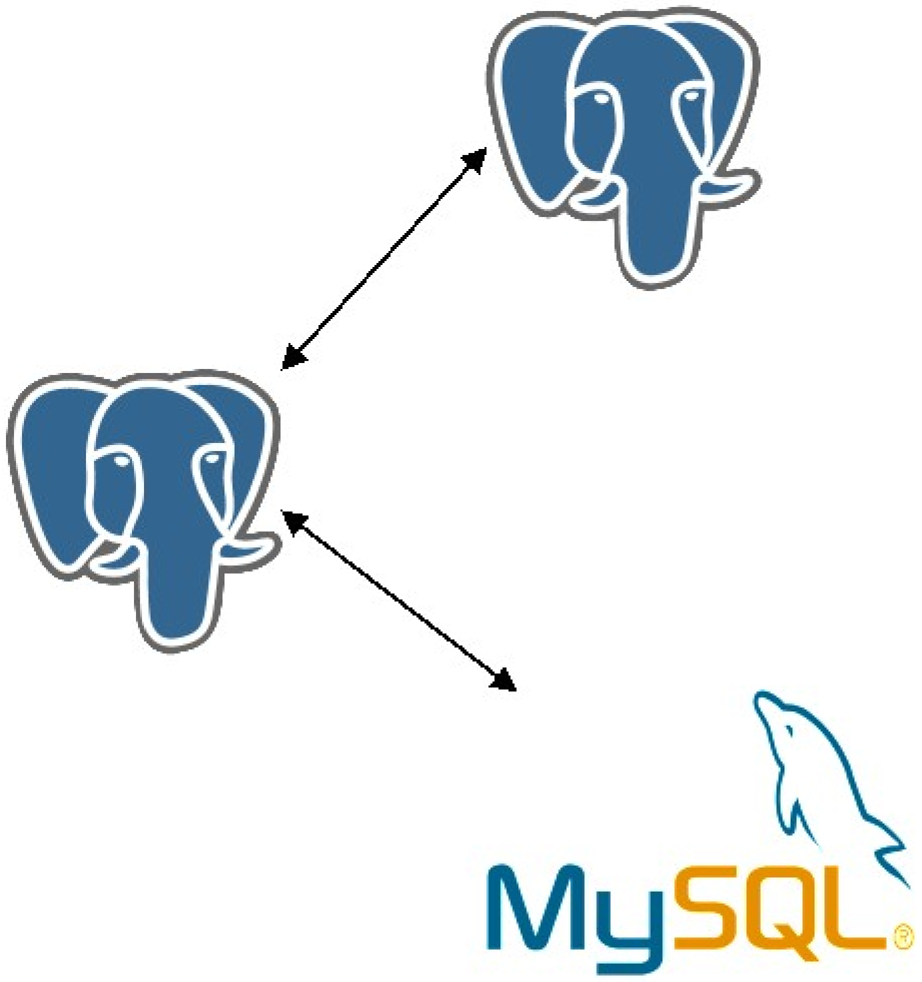
```
CREATE VIEW v_sensor_details AS
SELECT s.*, l.location_name
    FROM sensor_log s
    JOIN location l ON (l.id = s.location_id);
```

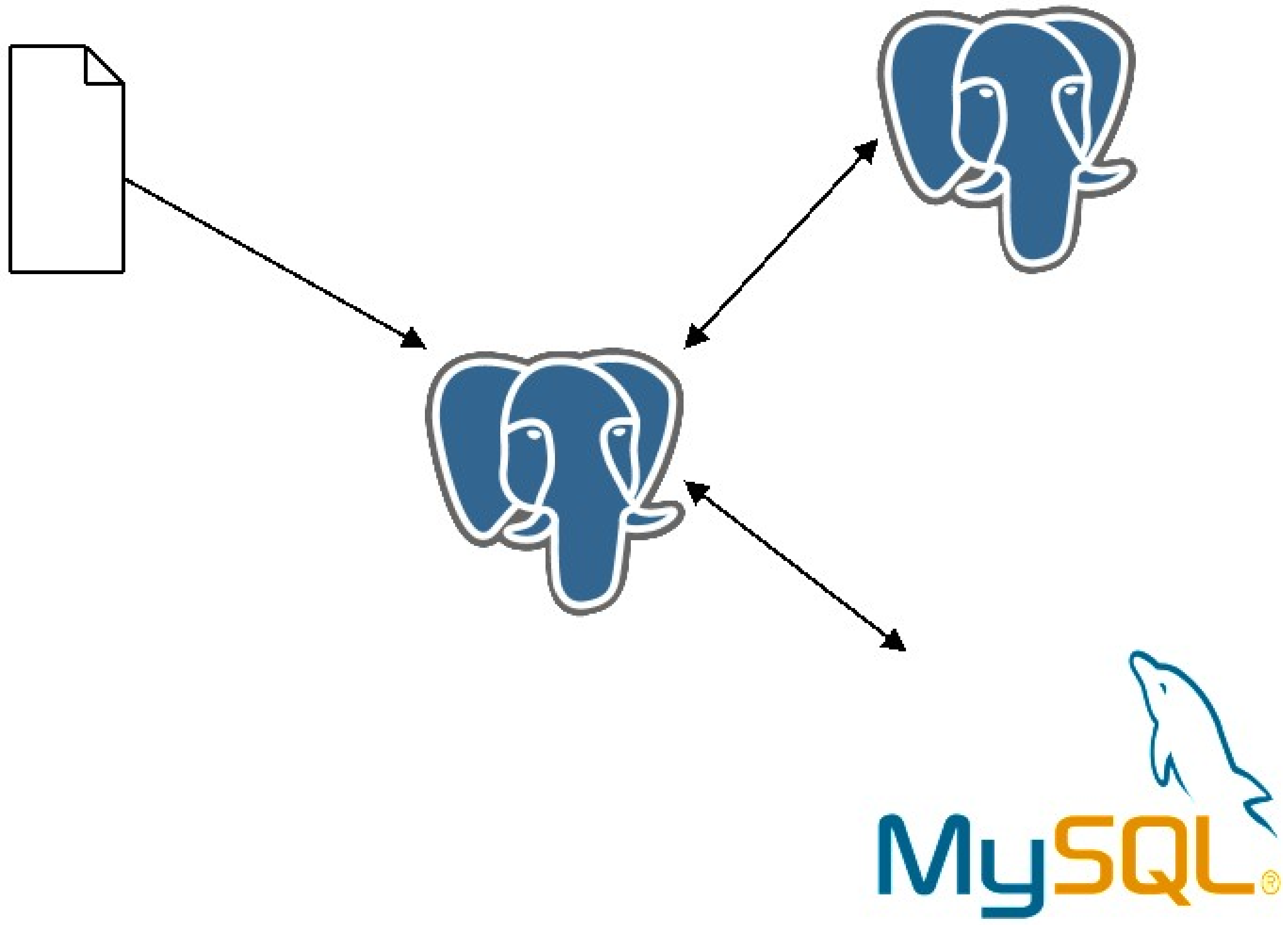
PostgreSQL FDW – Import the View

```
IMPORT FOREIGN SCHEMA PUBLIC
    LIMIT TO (v_sensor_details)
    FROM SERVER planet_postgresql
    INTO pgsql;

ANALYZE pgsql.v_sensor_details;
```

PostgreSQL FDW – Verify improvement

```
EXPLAIN
SELECT location_name, reading
    FROM pgsql.v_sensor_details
    WHERE reading_date >= '2019-10-2';
```


MySQL


MySQL

Import Data from Files with Python

```
CREATE SCHEMA python;
CREATE LANGUAGE plpythonu;
```

With a Stored Procedure

```
CREATE OR REPLACE FUNCTION python.yield_dictionary()
RETURNS TABLE (id INT, word TEXT) AS
$$
   for i, word in enumerate(open('/usr/share/dict/words', 'r')):
       yield (i, word.strip())

$$ LANGUAGE plpythonu;
```

Search for Words

```
SELECT *
FROM python.yield_dictionary()
WHERE word LIKE 'fun%'
LIMIT 5;
```

Let’s check performance

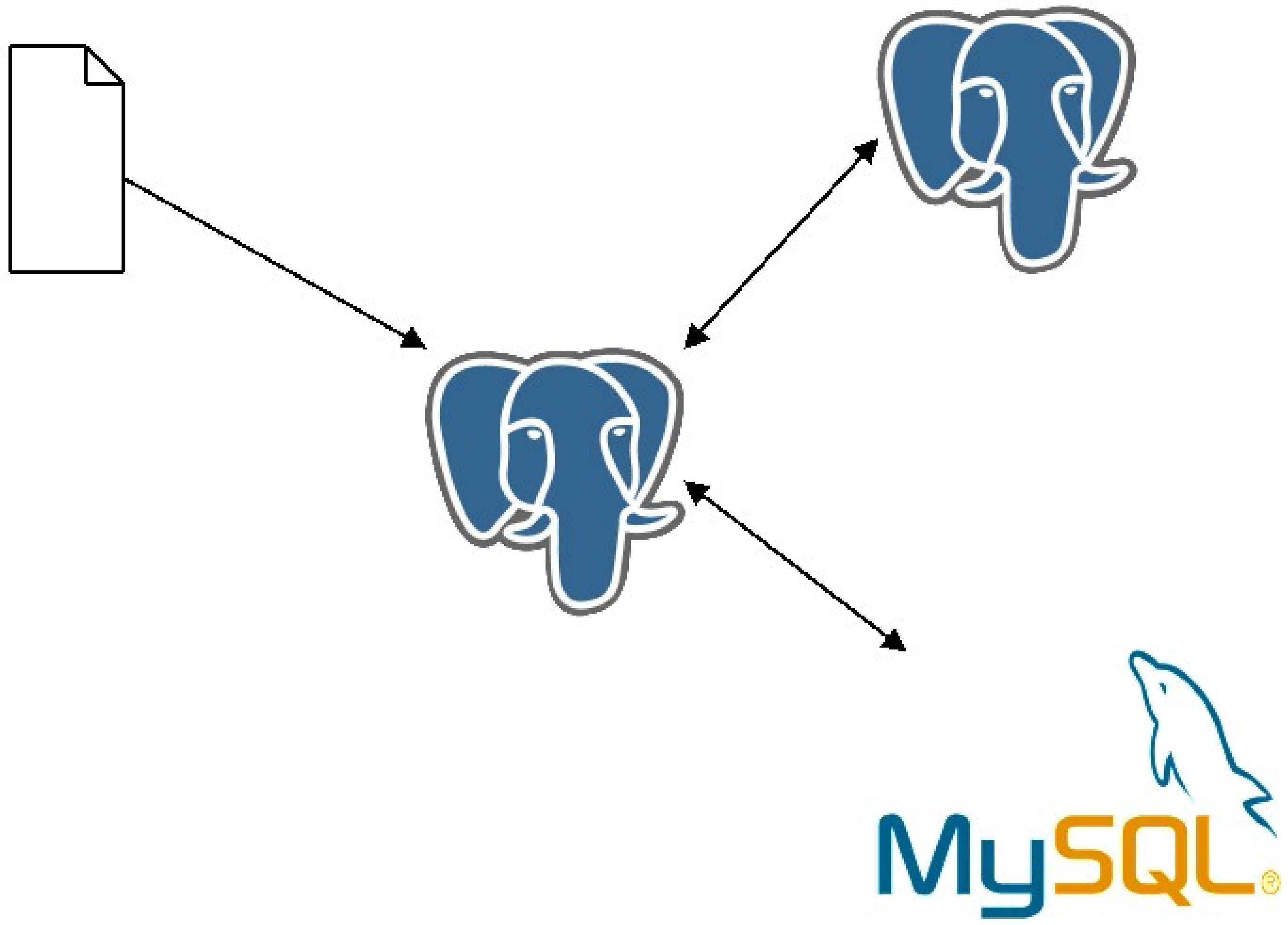
```
\timing on

SELECT *
FROM python.yield_dictionary()
WHERE word LIKE 'fun%'
LIMIT 5;
```

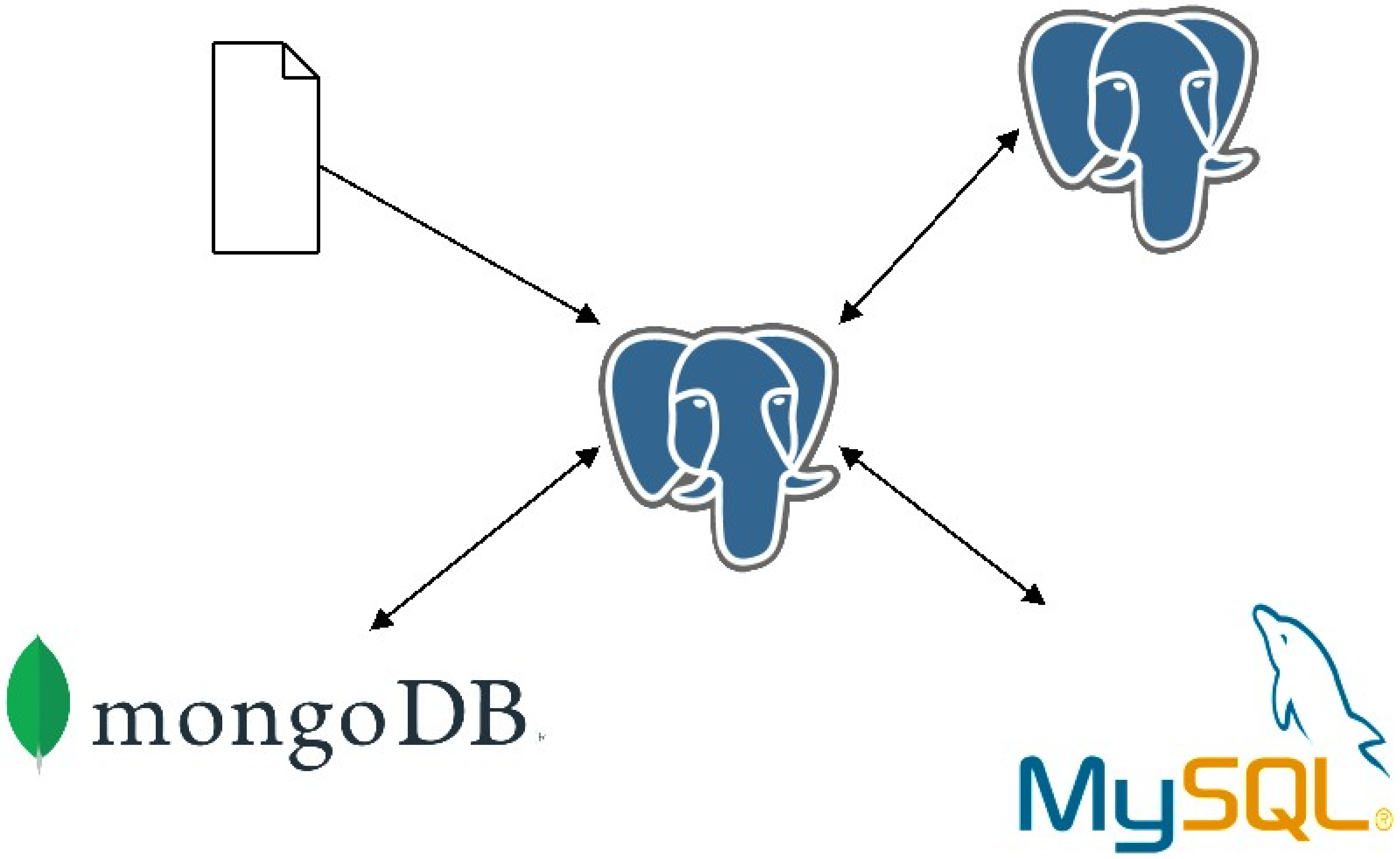
Good Old Cache to the Rescue

```
CREATE MATERIALIZED VIEW python.word_cache AS
    SELECT * FROM python.yield_dictionary();

ANALYZE python.word_cache;

CREATE INDEX idx_sensor_word_cache_word
    ON python.word_cache (word TEXT_PATTERN_OPS);
```

Verify improvement

```
EXPLAIN
SELECT *
    FROM python.word_cache
    WHERE word LIKE 'fun%';

                          QUERY PLAN
-----------------------------------------------------------
 Index Scan using idx_sensor_word_cache_word on word_cache
   Index Cond: ((word ~>=~ 'fun'::text) AND (word ~<~ 'fuo'::text))
   Filter: (word ~~ 'fun%'::text)
```


MySQL


mongoDB
MySQL

A bit of MongoDB

```
use pgconfeu
db.createCollection('sensorLog')

db.sensorLog.ensureIndex( { readingDate: 1 } )
db.sensorLog.ensureIndex( { location: 1 } )

db.sensorLog.count()
```

Mongo FDW – Setup

```
CREATE EXTENSION mongo_fdw;

CREATE SERVER mongo_pgconfeu
    FOREIGN DATA WRAPPER mongo_fdw
    OPTIONS (address '127.0.0.1', port '27017');

CREATE USER MAPPING FOR douglas
    SERVER mongo_pgconfeu;
```

Mongo FDW – A table in PostgreSQL

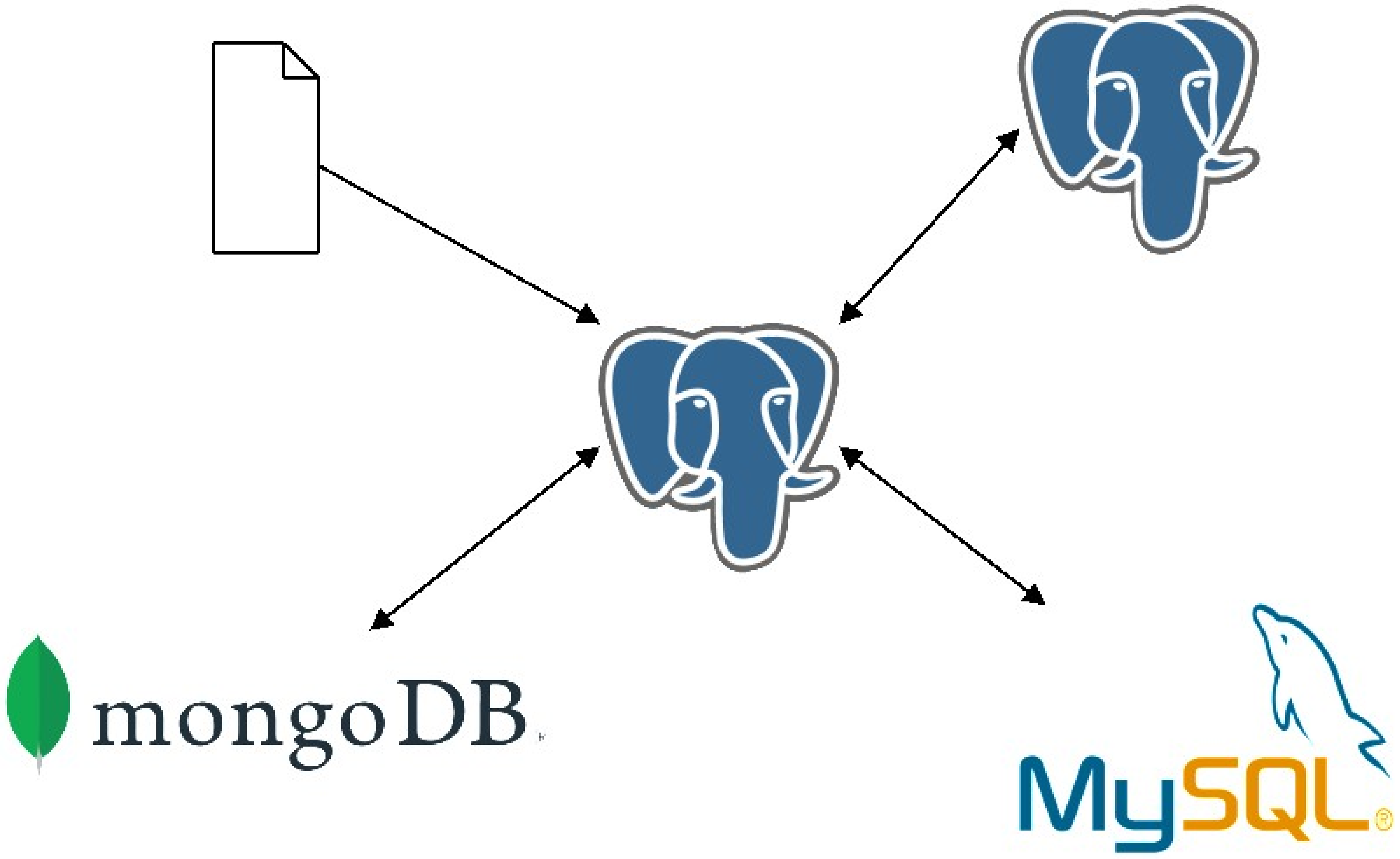
```
CREATE SCHEMA mongo;

CREATE FOREIGN TABLE mongo.sensor_log (
    _id             NAME,
    log_id          INT NOT NULL,
    location_id     INT NOT NULL,
    reading         BIGINT NOT NULL,
    reading_date    TIMESTAMP NOT NULL
)
SERVER mongo_pgconfeu
OPTIONS (database 'pgconfeu', collection 'sensorLog');
```

Mongo FDW – Read and Write

```
SELECT * FROM mongo.sensor_log;

INSERT INTO mongo.sensor_log
    (log_id, location_id, reading, reading_date)
     SELECT * FROM pgsql.sensor_log LIMIT 10;

SELECT * FROM mongo.sensor_log;
```

Sources of this talk

- Shaun M. Thomas's 2ndQuadrant Webinar
  `https://resources.2ndquadrant.com/webinar-data-integration-with-postgresql`
- Foreign Data Wrappers
  `https://wiki.postgresql.org/wiki/Foreign_data_wrappers`
- mongo_fdw
  `https://github.com/EnterpriseDB/mongo_fdw`


mongoDB
MySQL

Thoughts

- None of these tables exist in the central database
- We can read from different sources
- We can write to all of these sources
- We can construct extensions/FDWs to fill any gaps
- PostgreSQL works very well for data integration

Thanks and Remember
Benjamin Zander's Rule #6

Boriss Mejias
boriss.mejias@2ndquadrant.com
@tchorix