TOSHIBA

オープンソースカンファレンス2022 Spring セミナープログラム

PostgreSQLベースの分散SQLエンジンPGSpiderによるデータ仮想化の実現

株式会社東芝 ソフトウェア技術センター

幸田 拓耶
koda.takuya@toshiba.co.jp

李 威廷
weiting1.lee@toshiba.co.jp

本日の内容

01 PGSpider 概要

02 データソース接続モジュール FDW 概要

03 データソース接続モジュール FDW デモ

04 PGSpider マルチテナントテーブル デモ

05 まとめ

自己紹介

・幸田 拓耶
　　コンシューマ向け製品組み込みソフトウェア開発等に従事
　　現在はデータベース関連基礎技術開発を担当

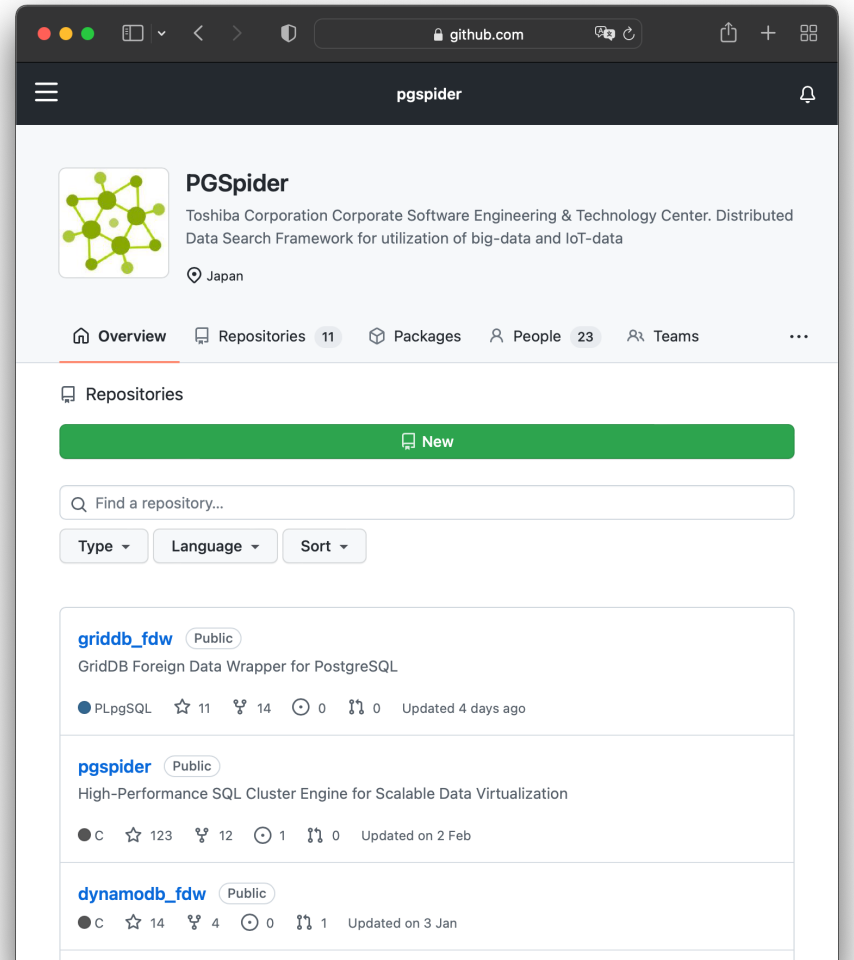
・李 威廷 (Lee Weiting)
　　台湾出身
　　データベース関連基礎技術開発を担当

PGSpider™ について

PostgreSQL をベースに開発している分散SQLエンジン

- オープンソースソフトウェアとして開発しており、関連モジュールを含めGitHub上にて公開中
  https://github.com/pgspider/pgspider

- 東芝 ソフトウェア技術センター
  共創ソフトウェア開発技術部が開発を推進
  社内で必要となる共通基盤ソフトウェアの開発やその開発を支える部門



01

PGSpider概要

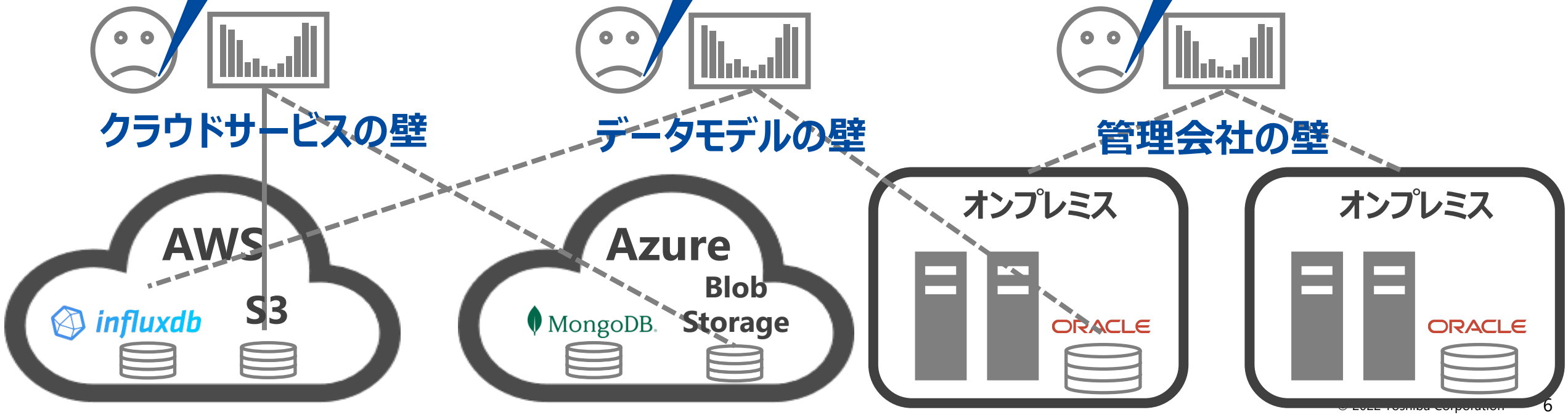
PGSpider開発の背景

データサイロ化：蓄積したデータが組織で分離し、活用・連携が困難な状況

別のクラウドサービスの
データにもアクセスしたい！
新しく開発しないと・・・

異なる種類のデータベースを組み合わせたサービスを作りたい！
異なるデータモデルを扱う処理を開発しないと・・・

異なる会社が持つ**データを組み合わせ**たサービスを提案したい！
どうやったら簡単にできる？



データ仮想化技術とは？

データ仮想化（Data Virtualization）の特徴

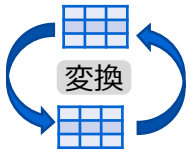
データコンシューマ

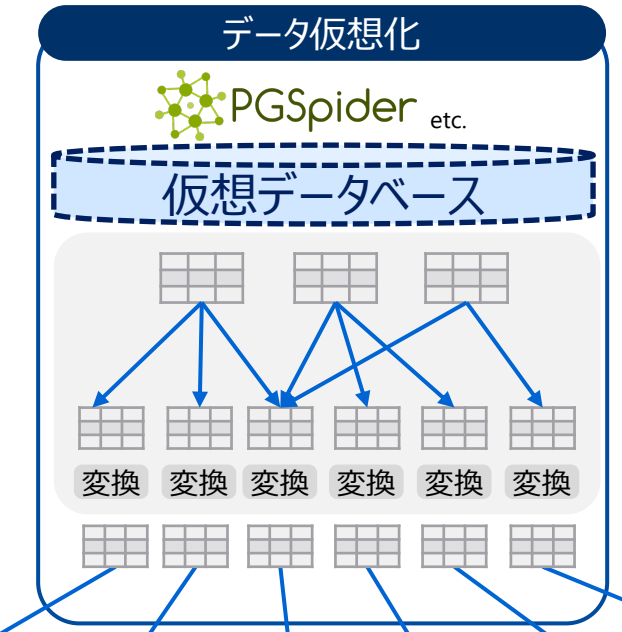
データ利活用が容易

共通インタフェースによるアクセス

データは、共通のデータモデルで表現
（仮想データレイヤ）

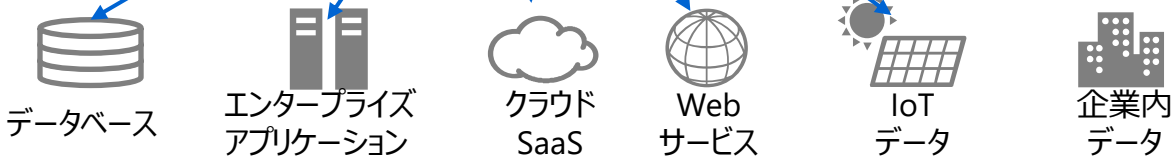
多様なデータ/アクセス手段を変換





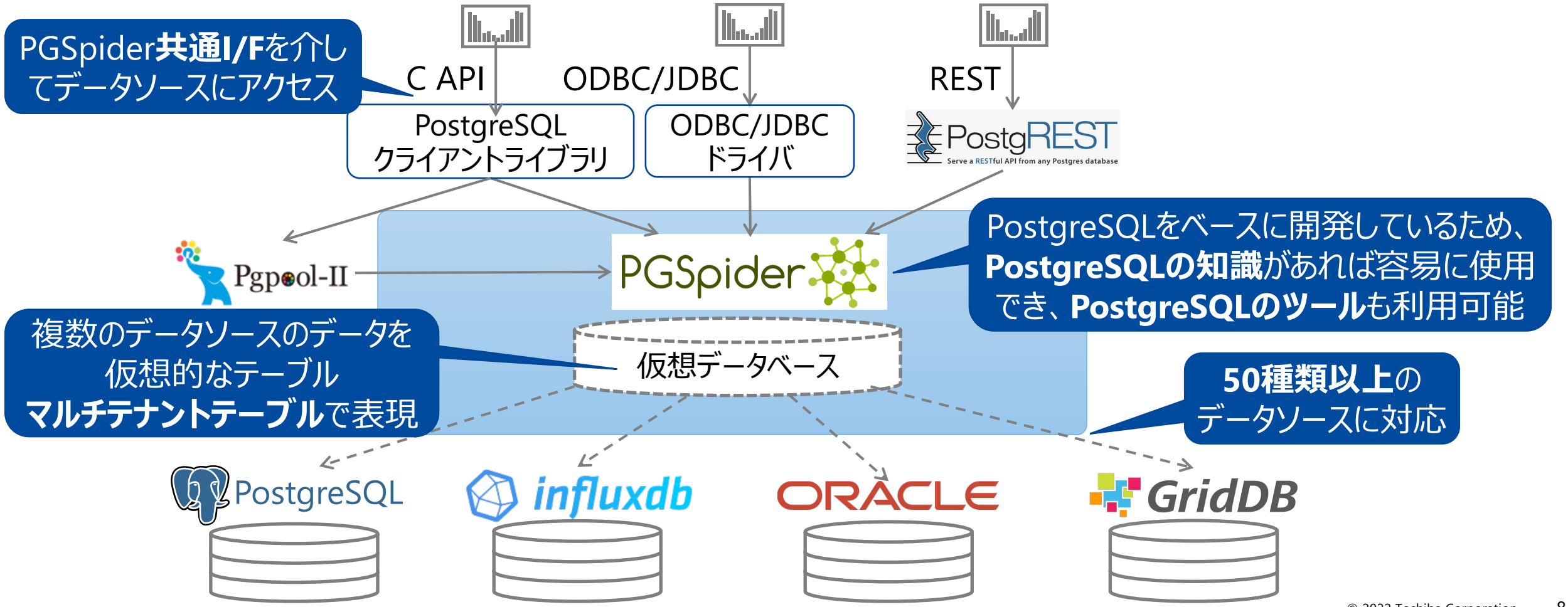
データソース

データは必要な範囲へアクセス
（コピーして集めない）



分散SQLエンジンPGSpiderの概要

共通I/Fで種類の異なるデータソースにアクセス可能



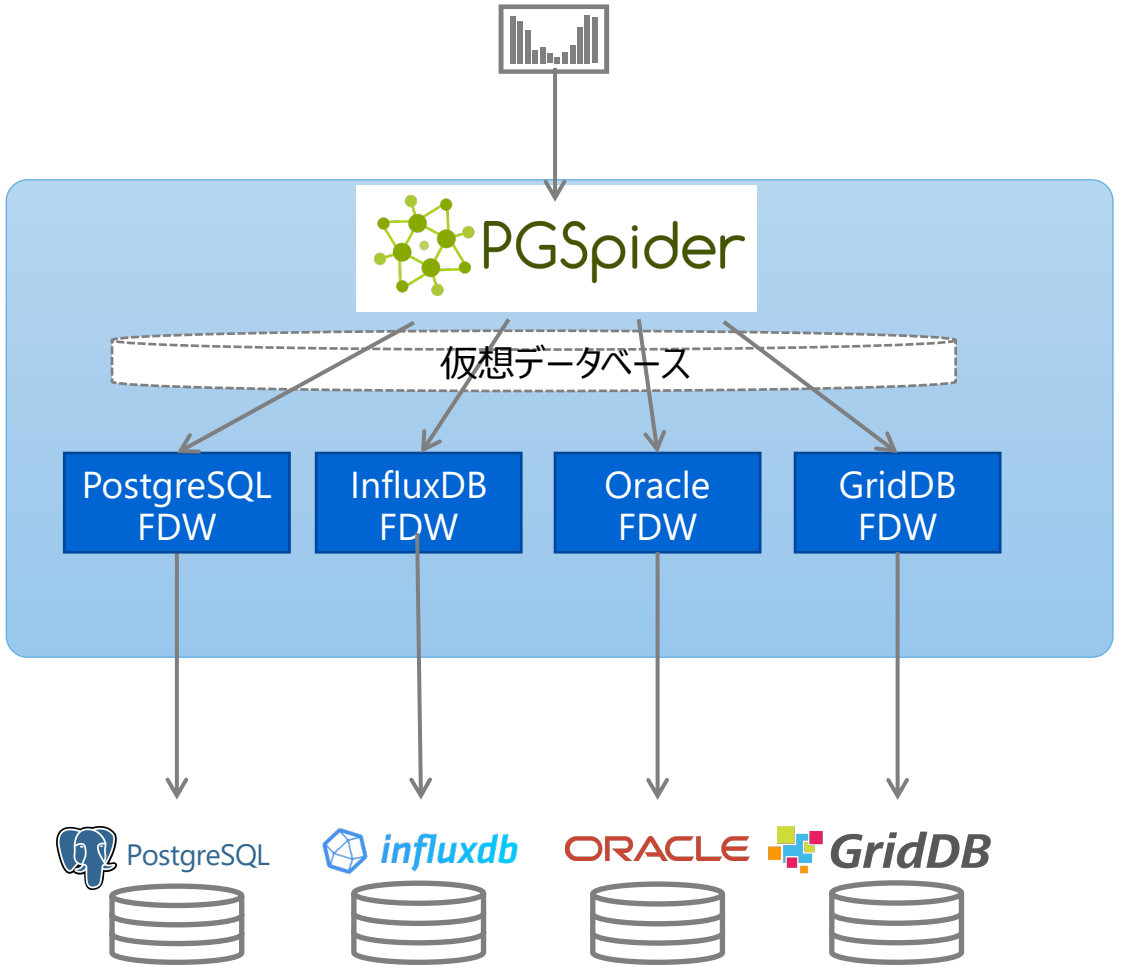
分散SQLエンジンPGSpiderの基本

特徴を３つ紹介

② 複数データソースの統合機能

③ データ検索の高速化
（パラレル処理・プッシュダウン）

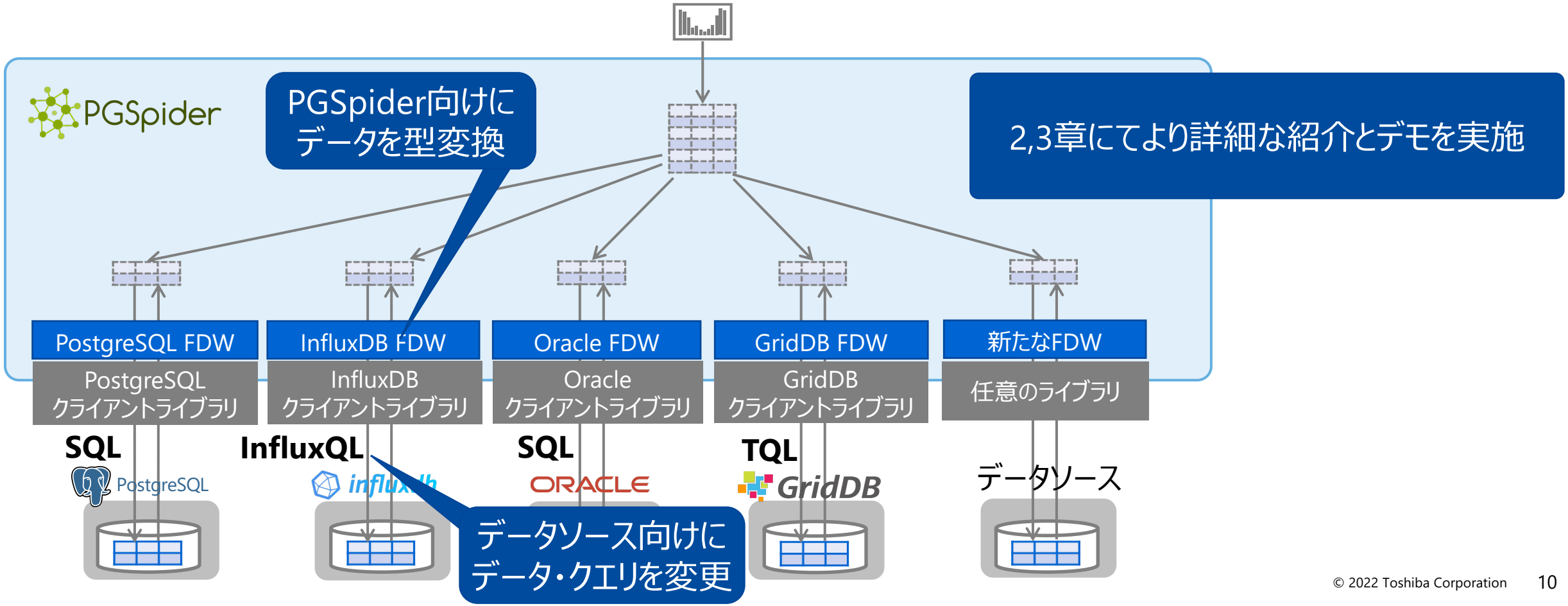
① 多様なデータソースへのアクセス機能



① 多様なデータソースへのアクセス機能

データソース接続モジュール(FDW)は、アプリからデータソースの差異を隠蔽

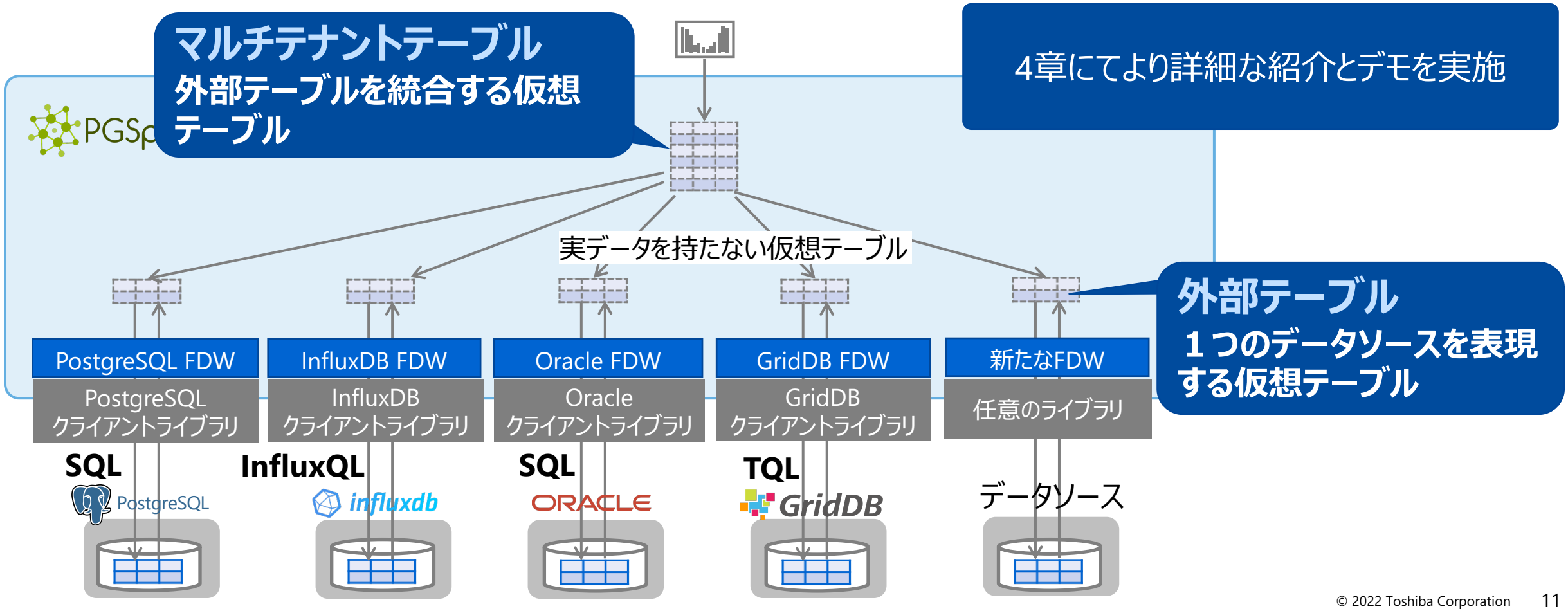
FDWは、データソース向けのクエリの構築、問い合わせ、結果取得およびデータ型の変換を行う。



② 複数データソースの統合機能

複数のデータソースに分散していても、１つのテーブルですべてにアクセス可能

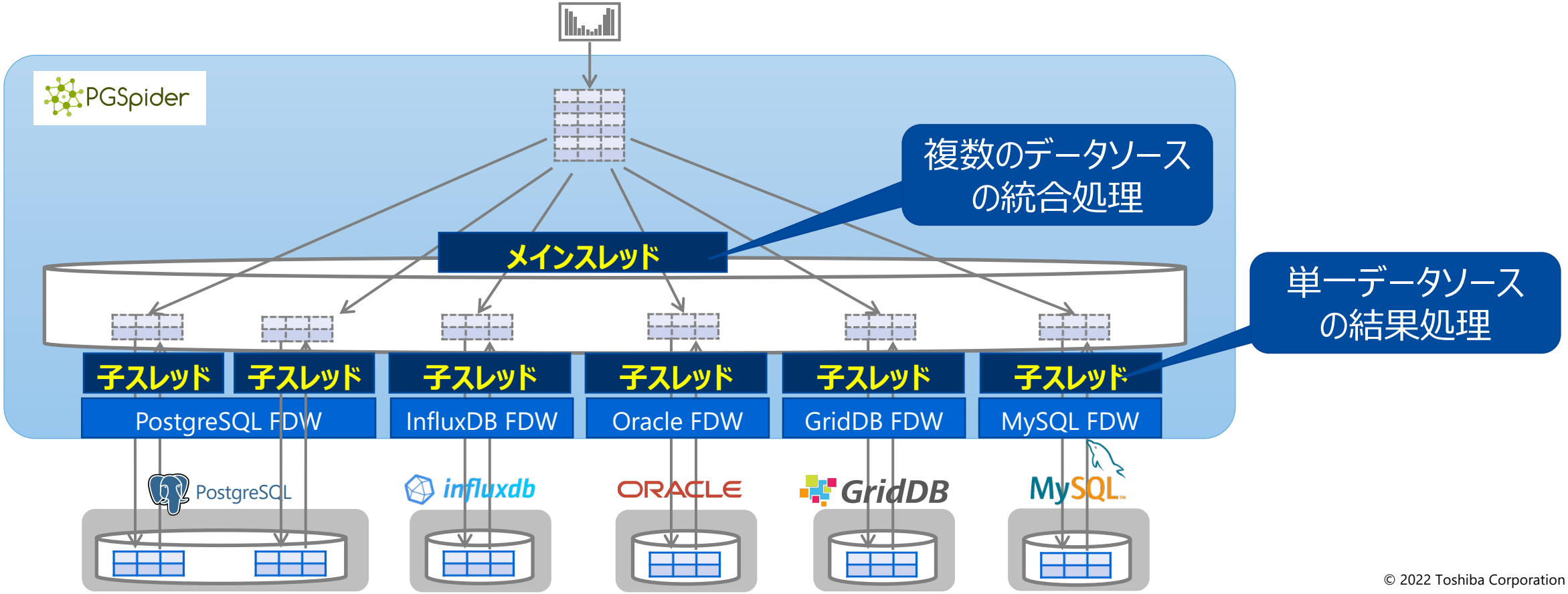
マルチテナントテーブルは、複数の外部テーブルのデータを統合した結果を返すテーブル



③ データ検索の高速化：パラレル処理

複数のデータソースであっても、自動的に並列アクセスによる高速化が可能

FDWでの処理は、外部テーブルごとにスレッド化して並列に行う。
（一番処理に時間が掛かるデータソースで性能が決まる）



③ データ検索の高速化：プッシュダウン

異なるデータソースであっても、自動的に効率の良い適切なアクセス方法に変換

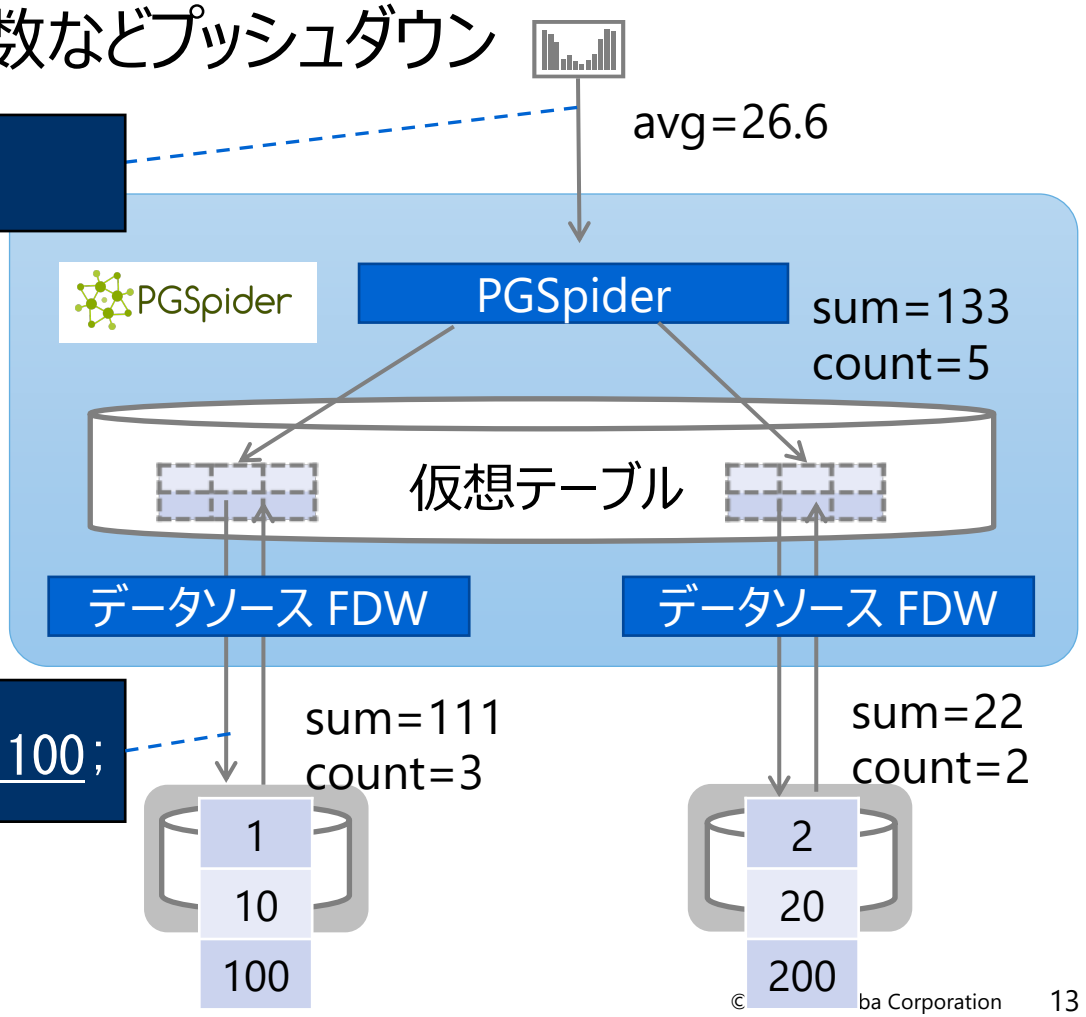
ターゲットリスト・WHERE条件・集約関数・SQL関数などプッシュダウン

```
SELECT avg(col1) FROM tbl WHERE col2 <= 100;
```

効率の良いデータソース向けのクエリを生成し、
なるべく必要なデータのみをデータソースから取得

```
SELECT sum(col1), count(col1) FROM tbl WHERE col2 <= 100;
```

PGSpiderではデータソース**独自関数もクエリ実行可能**

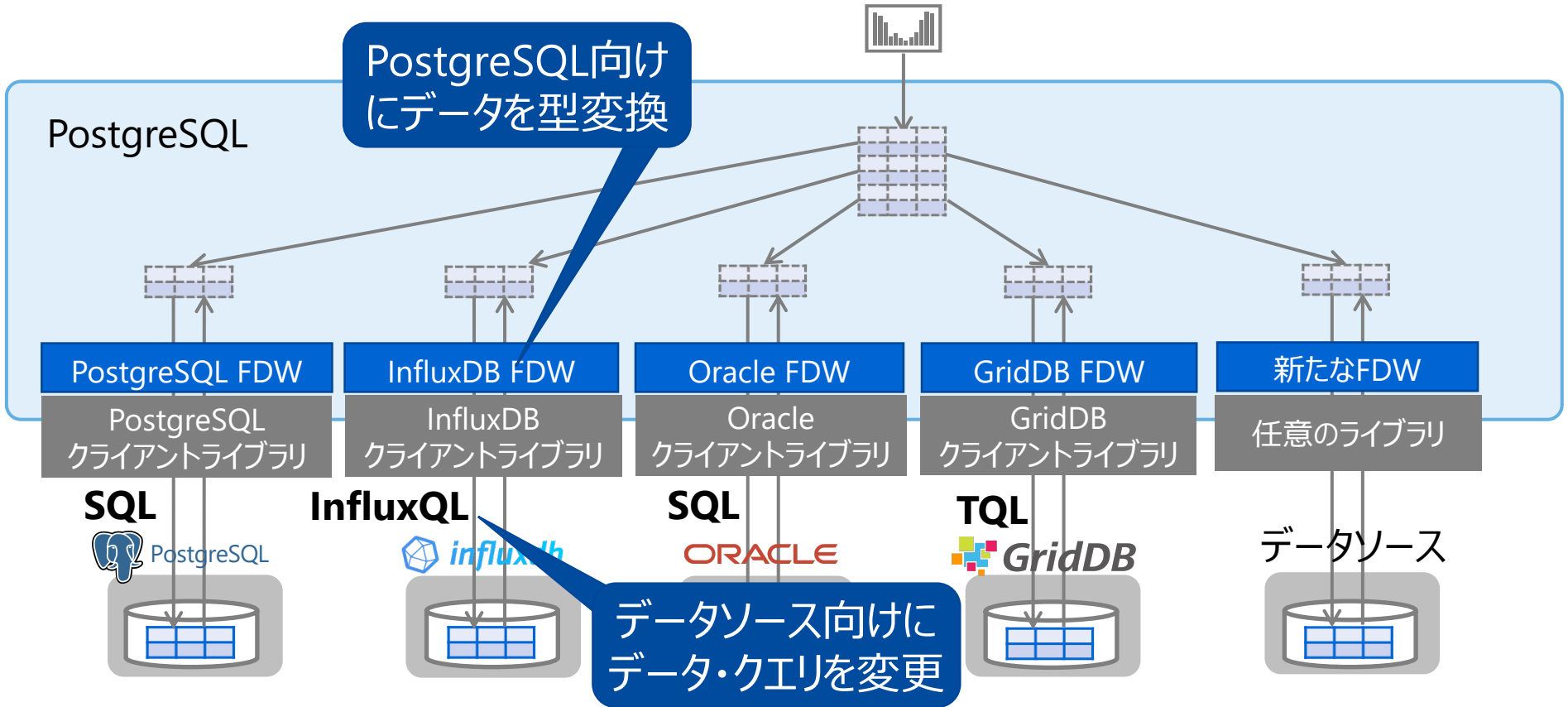


02

データソース接続モジュールFDW 概要

FDW (Foreign Data Wrapper) とは

- FDWはPostgreSQLから外部のデータソースへアクセスするための仕組み。
- 様々なFDWが開発されて公開されている。
  - FDWの一覧 (https://wiki.postgresql.org/wiki/Foreign_data_wrappers)



PGSpiderプロジェクトのFDWについて

公開場所
GitHub → https://github.com/pgspider

★ 独自開発

- **データソース対応拡充**
  - **時系列データベース** (InfluxDB★, GridDB★)
  - **組み込みDB** (SQLite★)
  - **オブジェクトストレージ連携** (ParquetS3)
- **データソース特有機能に対する機能強化**
  - **独自リモート関数**の実行機能追加 (MySQL, InfluxDB★, GridDB★, …)
  - **階層データ表現**へのアクセス対応 (MongoDB, DynamoDB★, ParquetS3)
- **アクセス性能改善 - 各種プッシュダウン機能強化**
  - 集約関数/関数/階層データ要素による条件句などの**処理をデータソース側で実行** (JDBC, ODBC, SQLite★, InfluxDB★, GridDB★, DynamoDB★, MySQL, MongoDB, ParquetS3)

データソースの特徴と分類

- RDBMS系　　主にSQL言語による問い合わせアクセスするタイプ
- ファイル系　　データの実体がデータファイル
- スキーマレス系　　表以外のデータ構造、ドキュメント/非構造型データ

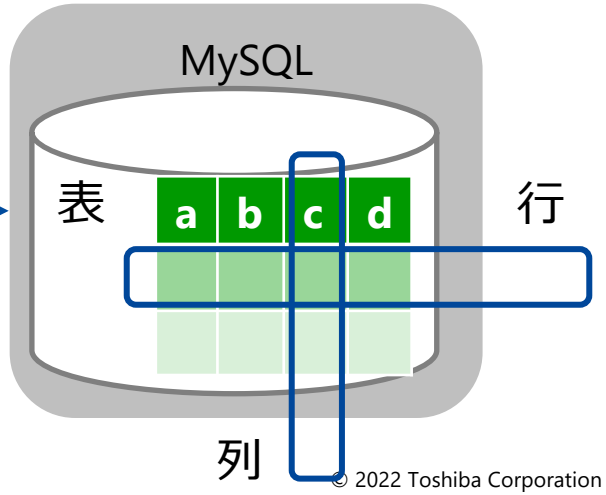
特徴種別	FDW	データソース	問い合わせ方法	データの形式
RDBMS系	PostgreSQL FDW	PostgreSQL	**SQL**	表形式
	SQLite FDW	SQLite	**SQL**	表形式
	MySQL FDW	MySQL	**SQL**	表形式
	ODBC FDW	ODBC接続可能なサービス	**SQL**	表形式
	JDBC FDW	JDBC接続可能なサービス	**SQL**	表形式
	PGSpider FDW	PGSpider (PGSpider間を接続)	**SQL**	表形式
	GridDB FDW	GridDB	**TQL**	Key-Value（スキーマあり）
ファイル系	File FDW	**テキストファイル（csvなど）**	行単位で直接取得	表形式相当
	ParquetS3 FDW	**Parquetファイル（ローカル＆AWS S3）**	列単位で直接取得	表形式相当
スキーマレス系	InfluxDB FDW	InfluxDB	InfluxQL	**Key-Value（スキーマレス）**
	DynamoDB FDW	DynamoDB	PartiQL	**Key-Value（スキーマレス）**
	MongoDB FDW	MongoDB	独自方式	**ドキュメント型（スキーマレス）**

RDBMS系のデータソースとFDW

- RDBMSは関係データベースの管理システムのことで、データを「表（テーブル）」形式で表現し、縦・横方向のデータを「列（カラム）」と「行（レコード）」と呼ぶ。
- テーブルに格納されたこれらのデータを処理するためには、「SQL」というデータベースを制御するための言語を用いる。SQLでは、データの定義、データの操作が可能。
- オープンソースのRDBMSとして、PostgreSQLやMySQLが有名。
- ここでは、RDBMSをデータソースとするFDWの例としてMySQL FDWを取り上げる。

SQLの例：

```
SELECT a,b,c FROM 表;
```



MySQL FDW (RDBMS系の例)

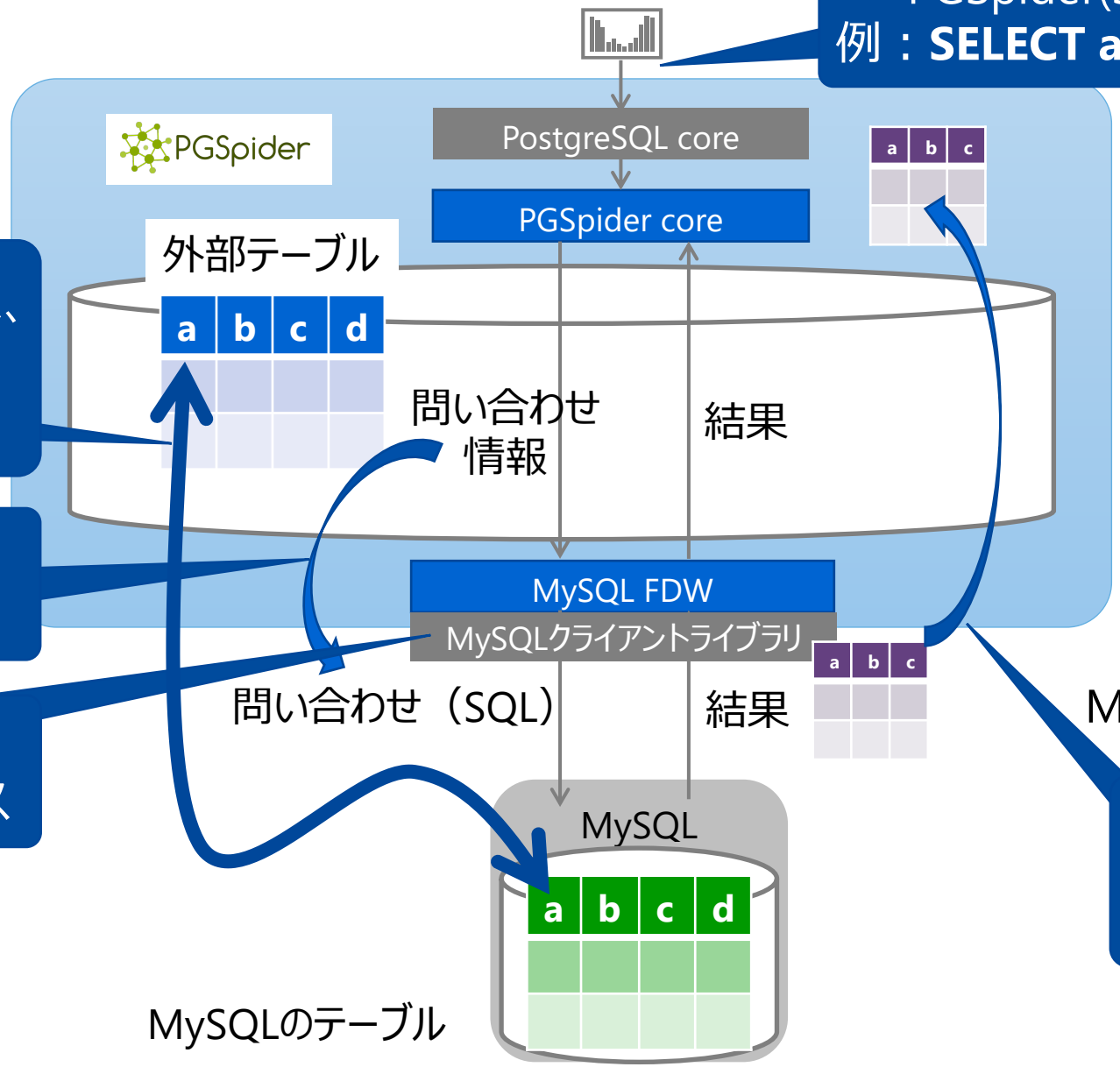
データ型マッピングの例
PGSpider：smallint
↓
MySQL：tinyint

外部テーブルのカラム型は、PGSpider内の同等のデータ型へマッピング

MySQL用のSQLを組み立て

MySQLのクライアントライブラリを介してアクセス

PGSpider(SQL言語)で問い合わせ
例：**SELECT a,b,c FROM 外部テーブル;**



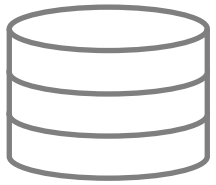
結果データの変換
PGSpider：short
↑
MySQL：MYSQL_TYPE_SHORT

MySQLのデータ型からPGSpiderのデータ型へ変換

スキーマレス系のデータソースとFDW

- スキーマレスとは、ここではRDBMSの表データのように、テーブルの列要素が事前に定まっていないデータ表現のことを意味する。
- その一例として、**MongoDB FDW**では、スキーマレスなデータベースであるMongoDBをデータソースとする。
- MongoDBは、JSON形式のデータを蓄える、ドキュメント指向データベース。
- データ処理にはSQLライクな専用のクエリが用意されている。

クエリの例： `db.data1.find( {"key1" : "record1-1"} )`



MongDB

data1

```
{"time":"2021/11/01", "key1"="record1-1", "key2"="record1-2"                    }
{"time":"2021/11/02", "key1"="record2-1", "key2"="record2-2", "key3"="record2-3"}
{"time":"2021/11/03", "key1"="record3-1",                     "key3"="record3-3"}
```

MongoDB FDW (スキーマレス系の例)

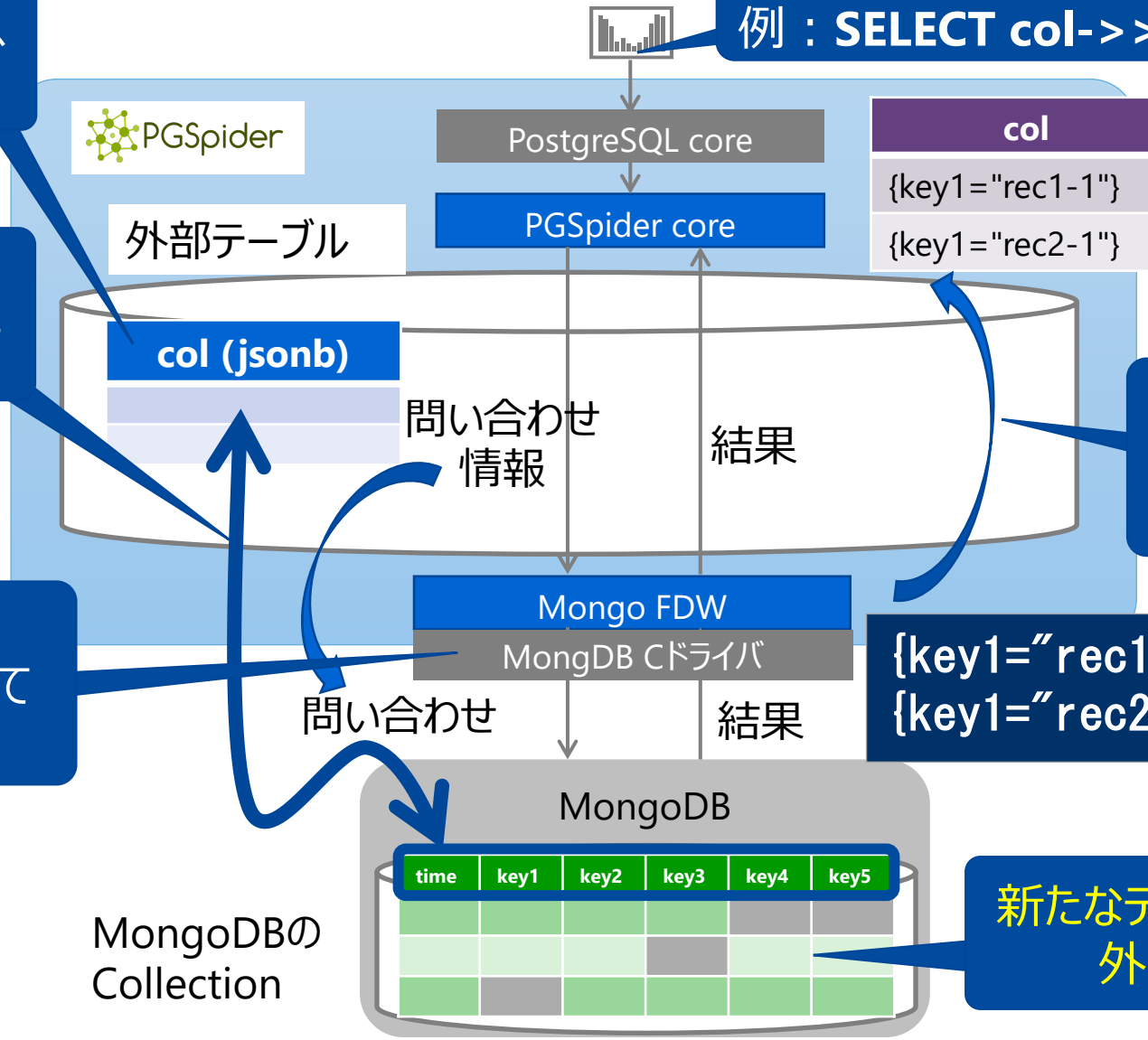
PGSpider(SQL言語)で問い合わせ
JSONデータとして非構造データにアクセス
例：**SELECT col->>'key1' FROM 外部テーブル;**

外部テーブルのカラム型は、
JSONB型にマッピング

MongoDB用の
問い合わせ文を組み立て

```
{ "pipeline" : [ { "$project" :
{ "key1" : { "$numberInt" :
"1" } } } ] }
```

MongoDBアクセス用
ドライバ(ライブラリ)を介して
アクセス



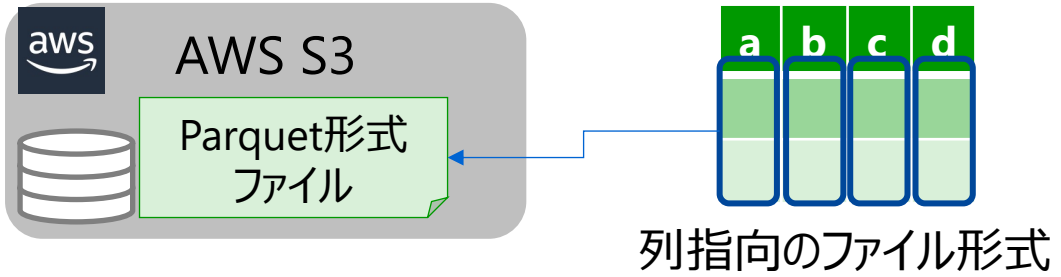
MongoDBのデータを
PGSpiderのJSONB型へ変換

MongoDBの
Collection

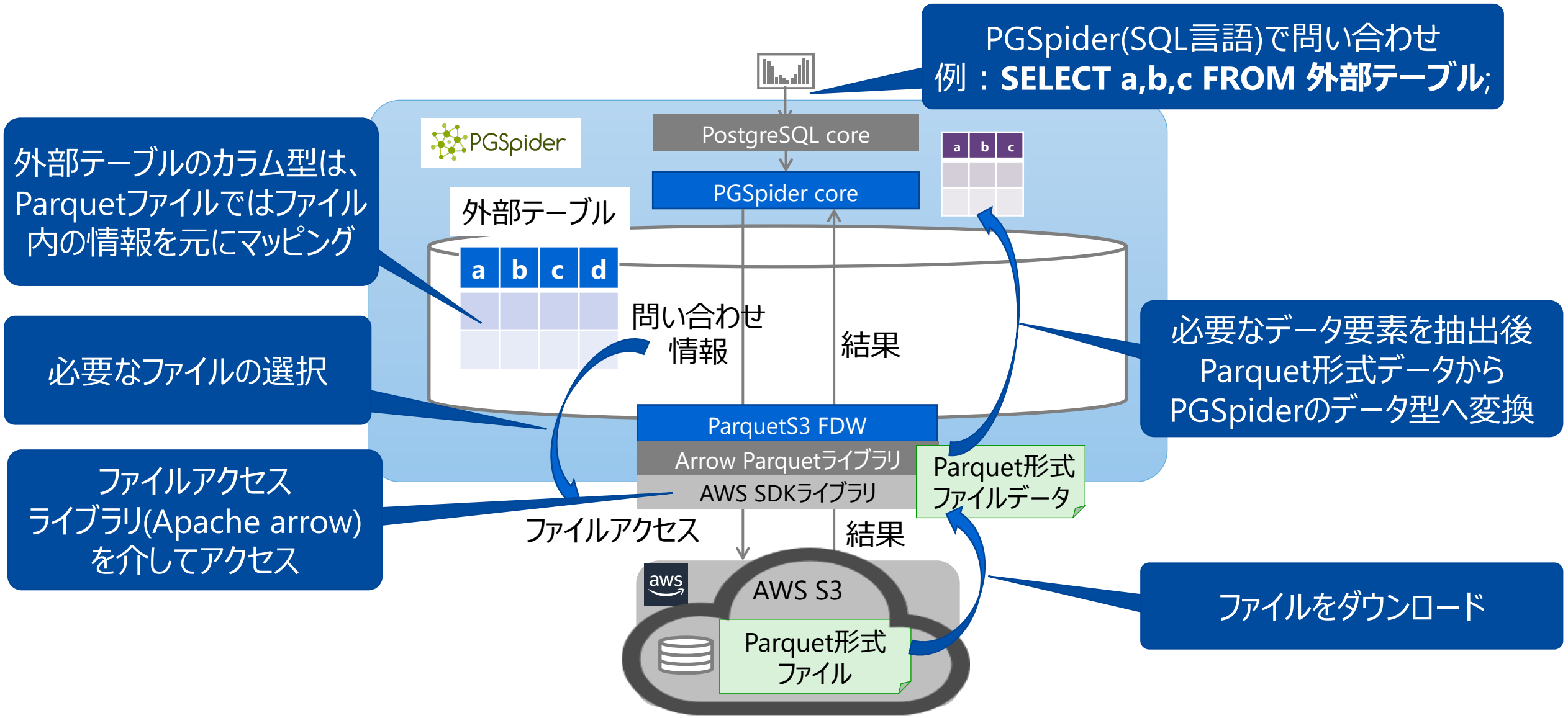
新たなデータ要素が追加されたとしても
外部テーブルの変更は不要

ファイル系のデータソースとFDW

- ファイルをデータソースとするFDWの例として、**ParquetS3 FDW**を説明する。
- **ParquetS3 FDW**では、**Amazon S3**上に配置された、**Parquet**形式のファイルをデータソースとする。
- **Amazon S3**（AWS Simple Storage Service）は、データファイルを「バケット」と呼ばれるリソース内のオブジェクトと表現し、Webサービス上のストレージへ保存する。
- **Parquet**形式のファイルは、列指向のデータファイル形式であり、列単位でデータをまとめることで圧縮効率や特定の列に限定した処理を効率的に扱うことができる。



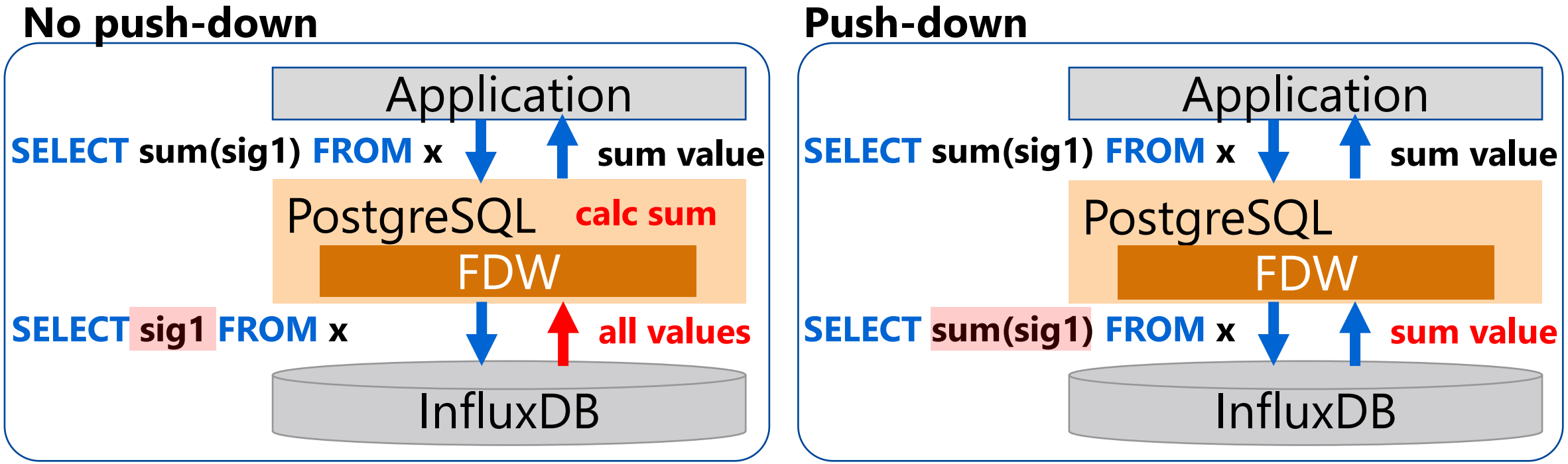
ParquetS3 FDW (ファイル系の例)



FDWの性能を決める機能　ー　プッシュダウン

- プッシュダウンとはクライアントから問い合わせのあったSQL文に含まれるWHERE句（絞り込み処理）、ORDER BY句（ソート処理）、集約関数などをリモート側で実行させる機構。

集約関数プッシュダウン



03

データソース接続モジュールFDW デモ

デモの流れの概要説明

- データソースへの接続する設定手順を紹介する。
- 以下の３つのデータソースへの接続デモを紹介する。

01 MySQL

02 MongoDB

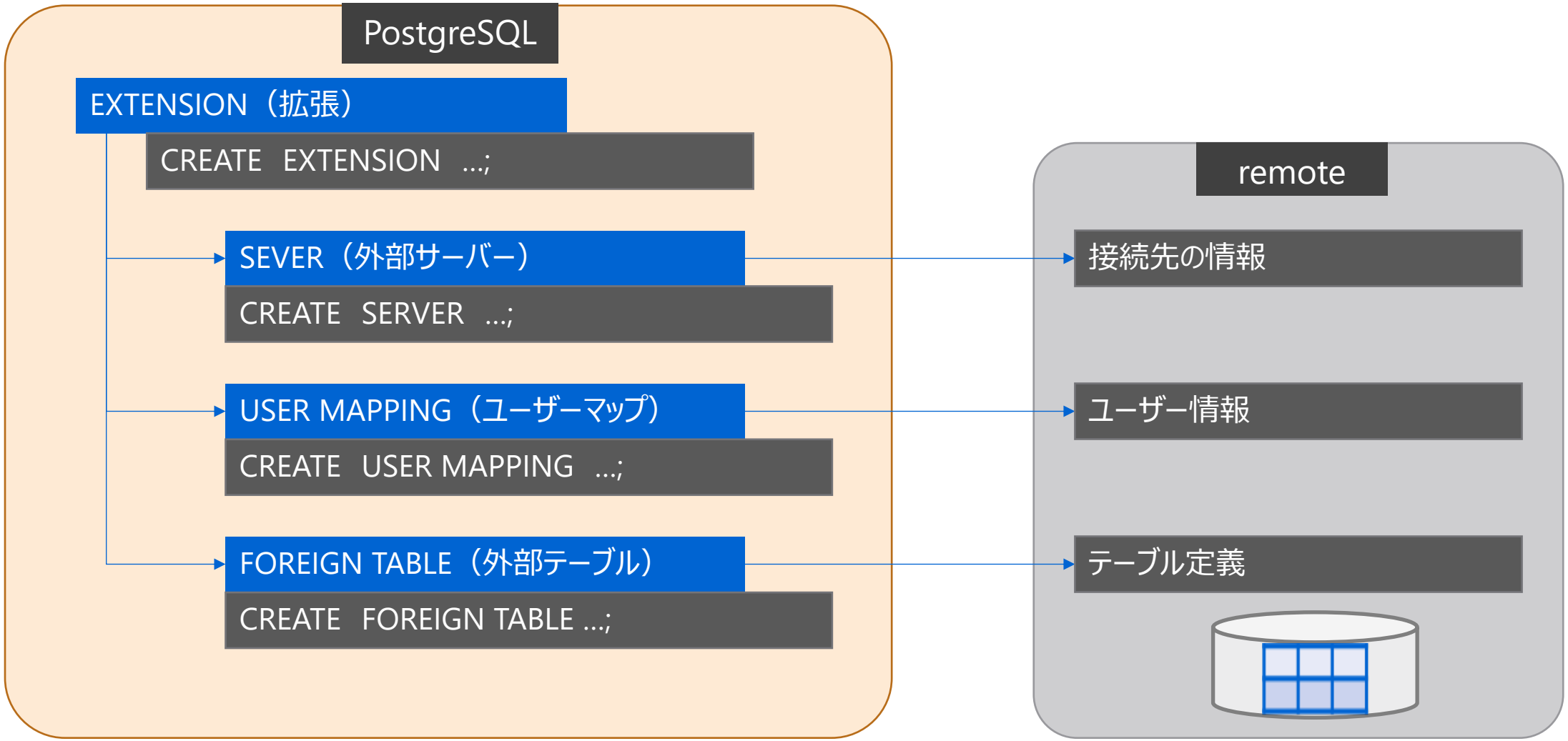
03 ParquetS3

ソースコードの入手とインストールについて

- PGSpider
  - GitHubより入手
    - https://github.com/pgspider/pgspider
    - READMEに沿ってビルド及びインストールを行う
- 各種データソース接続モジュール

接続先データソース	データソースタイプ	ソースコード	ビルド、インストール方法
Amazon S3	File系	https://github.com/pgspider/parquet_s3_fdw	READMEに沿ってビルド インストール
MySQL	RDB系	https://github.com/pgspider/mysql_fdw	
MongoDB	スキーマレス	https://github.com/pgspider/mongo_fdw	

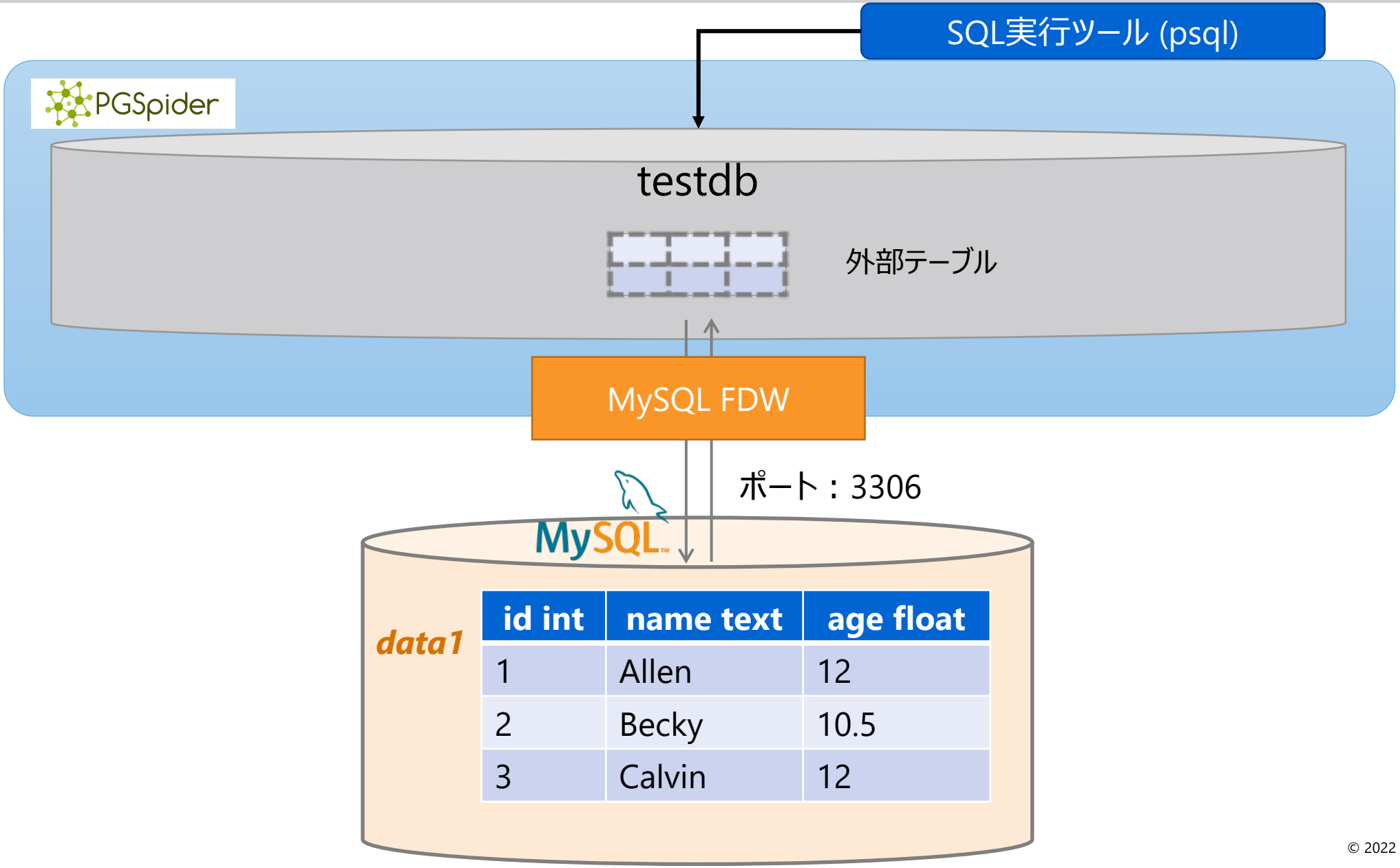
FDWの基本利用について



03-01

MySQL FDWデモ

PGSpider + MySQL構成



MySQL データ

data1

```
mysql> SELECT `id`, `name`, `age` FROM `mysql_demo`.`data1`;
+------+--------+------+
| id   | name   | age  |
+------+--------+------+
|    1 | Allen  |   12 |
|    2 | Becky  | 10.5 |
|    3 | Calvin |   12 |
|    4 | Daniel |   11 |
+------+--------+------+
4 rows in set (0.01 sec)
```

外部データソースMySQLを利用するための定義

```
CREATE EXTENSION mysql_fdw;
CREATE SERVER mysql_svr FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host '127.0.0.1', port '3306');
CREATE USER MAPPING FOR CURRENT_USER SERVER mysql_svr options(username 'admin', password
'admin');

CREATE FOREIGN TABLE data1__mysql__0 (id integer, name text, age float) SERVER mysql_svr OPTIONS
(dbname 'mysql_demo', table_name 'data1');
```

外部データソースMySQLのデータを検索

```
SELECT * FROM data1__mysql__0;
EXPLAIN VERBOSE SELECT * FROM data1__mysql__0;
```

```
pgspider=# SELECT * FROM data1__mysql__0;
 id |  name  | age
----+--------+-----
  1 | Allen  |   12
  2 | Becky  | 10.5
  3 | Calvin |   12
  4 | Daniel |   11
(4 rows)

pgspider=# EXPLAIN VERBOSE SELECT * FROM data1__mysql__0;
                                 QUERY PLAN
-----------------------------------------------------------------------------
 Foreign Scan on public.data1__mysql__0  (cost=100.00..146.12 rows=1204 width=44)
   Output: id, name, age
   Local server startup cost: 10
   Remote query: SELECT `id`, `name`, `age` FROM `mysql_demo`.`data1`
(4 rows)
```

外部データソースMySQLのデータを検索

Columnプッシュダウン

```
SELECT id, name FROM data1__mysql__0;
EXPLAIN VERBOSE SELECT id, name FROM data1__mysql__0;
```

```
pgspider=# SELECT id, name FROM data1__mysql__0;
 id |  name
----+--------
  1 | Allen
  2 | Becky
  3 | Calvin
  4 | Daniel
(4 rows)

pgspider=# EXPLAIN VERBOSE SELECT id, name FROM data1__mysql__0;
                                QUERY PLAN
---------------------------------------------------------------------------
 Foreign Scan on public.data1__mysql__0  (cost=100.00..150.95 rows=1365 width=36)
   Output: id, name
   Local server startup cost: 10
   Remote query: SELECT `id`, `name` FROM `mysql_demo`.`data1`
(4 rows)
```

外部データソースMySQLのデータを検索

WHERE句プッシュダウン

```
SELECT id, name FROM data1__mysql__0 WHERE age > 11;
EXPLAIN VERBOSE SELECT id, name FROM data1__mysql__0 WHERE age > 11;
```

```
pgspider=# SELECT id, name FROM data1__mysql__0 WHERE age > 11;
 id |  name
----+--------
  1 | Allen
  3 | Calvin
(2 rows)

pgspider=# EXPLAIN VERBOSE SELECT id, name FROM data1__mysql__0 WHERE age > 11;
                                QUERY PLAN
---------------------------------------------------------------------------
 Foreign Scan on public.data1__mysql__0  (cost=100.00..136.16 rows=455 width=36)
   Output: id, name
   Local server startup cost: 10
   Remote query: SELECT `id`, `name` FROM `mysql_demo`.`data1` WHERE ((`age` > 11))
(4 rows)
```

外部データソースMySQLのデータを検索

集約関数プッシュダウン

```
SELECT count(id) FROM data1__mysql__0;
EXPLAIN VERBOSE SELECT count(id) FROM data1__mysql__0;
```

```
pgspider=# SELECT count(id) FROM data1__mysql__0;
 count
-------
     4
(1 row)

pgspider=# EXPLAIN VERBOSE SELECT count(id) FROM data1__mysql__0;
                          QUERY PLAN
---------------------------------------------------------------
 Foreign Scan  (cost=107.31..146.59 rows=1 width=8)
   Output: (count(id))
   Local server startup cost: 10
   Remote query: SELECT count(`id`) FROM `mysql_demo`.`data1`
(4 rows)
```

外部データソースMySQLのデータを検索

PostgreSQLにない(MySQL特有)関数プッシュダウン

- CONV関数:第一引数の数値または文字列をX進数からY進数へ変換する

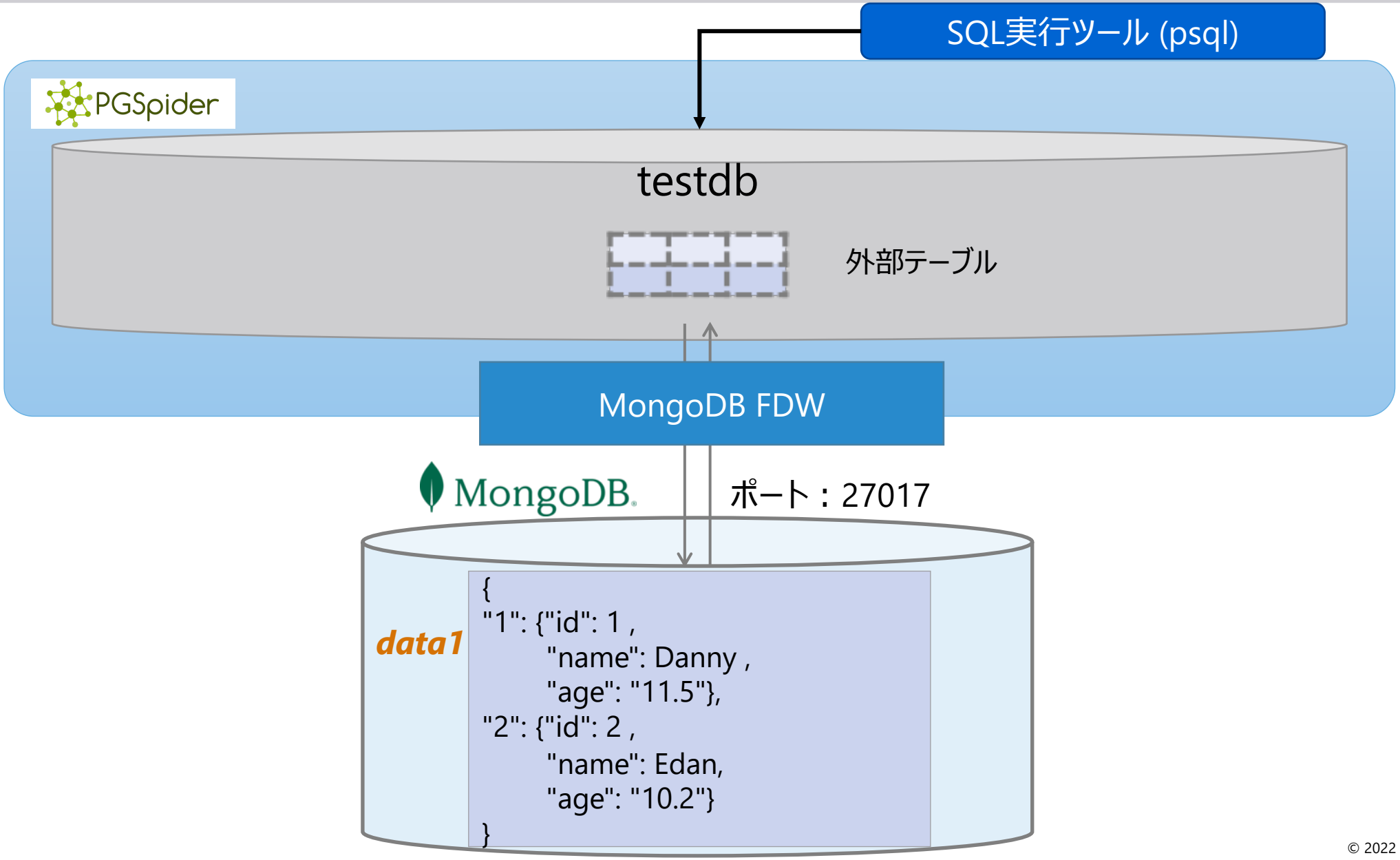
```
SELECT conv(age, 10, 16) FROM data1__mysql__0;
EXPLAIN VERBOSE SELECT conv(age, 10, 16) FROM data1__mysql__0;
```

```
pgspider=# SELECT conv(age, 10, 16) FROM data1__mysql__0;
 conv
------
 C
 A
 C
 B
(4 rows)
pgspider=# EXPLAIN VERBOSE SELECT conv(age, 10, 16) FROM data1__mysql__0;
                                QUERY PLAN
---------------------------------------------------------------------------
 Foreign Scan on public.data1__mysql__0  (cost=0.00..6.40 rows=2560 width=32)
   Output: (conv(age, 10, 16))
   Local server startup cost: 10
   Remote query: SELECT conv(`age`, 10, 16) FROM `mysql_demo`.`data1`
(4 rows)
```

03-02

MongoDB FDWデモ

PGSpider + MongoDB構成



MongoDB データ

data1

```
> db.data1.find()
{ "_id" : ObjectId("6221edfe11ff63885b21d783"), "id" : 1, "name" : "Ann", "age" : 10 }
{ "_id" : ObjectId("6221ee0a11ff63885b21d784"), "id" : 2, "name" : "Ben", "age" : 10.5 }
{ "_id" : ObjectId("6221ee1911ff63885b21d785"), "id" : 3, "name" : "Cindy", "age" : 10.5 }
```

data2

```
> db.data2.find()
{ "_id" : ObjectId("6221edd811ff63885b21d781"), "col1" : 401, "col2" : { "msg" : "success",
"code" : "ok" } }
{ "_id" : ObjectId("6221edda11ff63885b21d782"), "col1" : 402, "col2" : { "err" : "bad access",
"code" : "ng" } }
```

外部データソースMongoDBを利用するための定義

```
CREATE EXTENSION mongo_fdw;
CREATE SERVER mongo_svr FOREIGN DATA WRAPPER mongo_fdw OPTIONS(address '127.0.0.1', port
'27017');
CREATE USER MAPPING FOR CURRENT_USER SERVER mongo_svr OPTIONS(username 'admin', password
'admin');

CREATE FOREIGN TABLE data1__mongo_svr__0 (id int, name text, age float)
SERVER mongo_svr OPTIONS (database 'admin', collection 'data1');
```

外部データソースMongoDBのデータを検索

```
SELECT * FROM data1__mongo_svr__0;
EXPLAIN VERBOSE SELECT * FROM data1__mongo_svr__0;
```

```
pgspider=# SELECT * FROM data1__mongo_svr__0;
 id | name  | age
----+-------+------
  1 | Ann   |   10
  2 | Ben   | 10.5
  3 | Cindy | 10.5
(3 rows)

pgspider=# EXPLAIN VERBOSE SELECT * FROM data1__mongo_svr__0;
                                                                          QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------
 Foreign Scan on public.data1__mongo_svr__0  (cost=0.00..0.00 rows=1000 width=44)
   Output: id, name, age
   Foreign Namespace: admin.t1
   Query document: { "pipeline" : [ { "$project" : { "id" : { "$numberInt" : "1" }, "name" :
{ "$numberInt" : "1" }, "age" : { "$numberInt" : "1" } } } ] }
(4 rows)
```

外部データソースMongoDBのデータを検索

Column, WHERE句プッシュダウン

```
SELECT id, name FROM data1__mongo_svr__0 WHERE age > 10;
EXPLAIN VERBOSE SELECT id, name FROM data1__mongo_svr__0 WHERE age > 10;
```

```
pgspider=# SELECT id, name FROM data1__mongo_svr__0 WHERE age > 10;
 id | name
----+-------
  2 | Ben
  3 | Cindy
(2 rows)

pgspider=# EXPLAIN VERBOSE SELECT id, name FROM data1__mongo_svr__0 WHERE age > 10;
                                                                                            QUERY PLAN
------------------------------------------------------------------------------------------------------------
Foreign Scan on public.data1__mongo_svr__0  (cost=0.00..0.00 rows=1000 width=36)
   Output: id, name
   Foreign Namespace: admin.t1
   Query document: { "pipeline" : [ { "$match" : { "age" : { "$gt" : { "$numberDouble" : "10.0" } } } },
{ "$project" : { "id" : { "$numberInt" : "1" }, "name"
 : { "$numberInt" : "1" } } } ] }
(4 rows)
```

外部データソースMongoDBのデータを検索

集約関数プッシュダウン

```
SELECT sum(age) FROM data1__mongo_svr__0;
EXPLAIN VERBOSE SELECT sum(age) FROM data1__mongo_svr__0;
```

```
pgspider=# SELECT sum(age) FROM data1__mongo_svr__0;
 sum
-----
  31
(1 row)

pgspider=# EXPLAIN VERBOSE SELECT sum(age) FROM data1__mongo_svr__0;
                                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------
------------------------------------------------
 Foreign Scan  (cost=0.00..0.00 rows=1 width=8)
   Output: (sum(age))
   Foreign Namespace: admin.t1
   Query document: { "pipeline" : [ { "$group" : { "_id" : {  }, "ref0" : { "$sum" : "$age" } } },
{ "$project" : { "ref0" : { "$numberInt" : "1" } } } ] }
(4 rows)
```

外部データソースMongoDBを利用するための定義

```
CREATE FOREIGN TABLE data2__mongo_svr__0(col1 integer, col2 jsonb)
SERVER mongo_svr OPTIONS (database 'admin', collection 'data2');
```

外部データソースMongoDBのデータを検索

JSONデータとして非構造データにアクセス

```
SELECT * FROM data2__mongo_svr__0;
EXPLAIN VERBOSE SELECT * FROM data2__mongo_svr__0;
```

```
pgspider=# SELECT * FROM data2__mongo_svr__0;
 col1 |                 col2
------+--------------------------------------
  401 | {"msg": "success", "code": "ok"}
  402 | {"err": "bad access", "code": "ng"}
(2 rows)

pgspider=# EXPLAIN VERBOSE SELECT * FROM data2__mongo_svr__0;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
Foreign Scan on public.data2__mongo_svr__0  (cost=0.00..0.00 rows=1000 width=36)
   Output: col1, col2
   Foreign Namespace: admin.t2
   Query document: { "pipeline" : [ { "$project" : { "col1" : { "$numberInt" : "1" }, "col2" :
{ "$numberInt" : "1" } } } ] }
(4 rows)
```

外部データソースMongoDBのデータを検索

JSONデータとして非構造データにアクセス

```
SELECT col2->'code' code FROM data2__mongo_svr__0;
EXPLAIN VERBOSE SELECT col2->'code' code FROM data2__mongo_svr__0;
```

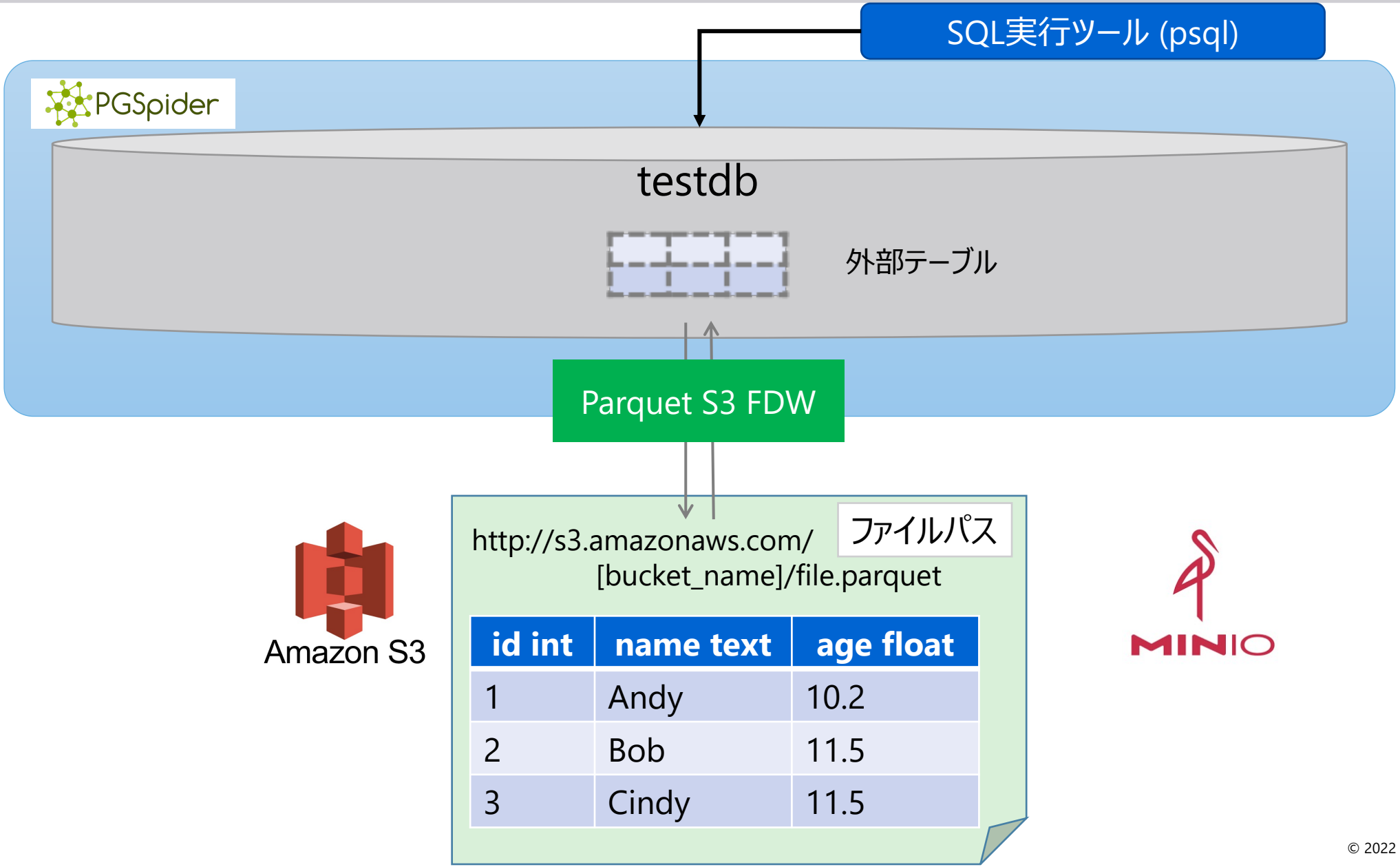
```
pgspider=# SELECT col2->'code' code FROM data2__mongo_svr__0;
 code
------
 "ok"
 "ng"
(2 rows)

pgspider=# EXPLAIN VERBOSE SELECT col2->'code' code FROM data2__mongo_svr__0;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Foreign Scan on public.data2__mongo_svr__0  (cost=0.00..2.50 rows=1000 width=32)
   Output: ((col2 -> 'code'::text))
   Foreign Namespace: admin.t2
   Query document: { "pipeline" : [ { "$project" : { "ref0" : "$col2.code" } } ] }
(4 rows)
```

03-03

ParquetS3 FDWデモ

PGSpider + Amazon S3 + Parquetファイル 構成



id int	name text	age float
1	Andy	10.2
2	Bob	11.5
3	Cindy	11.5

Parquet file データ

data1.parquet

```
   id   name   age
0   1   Alex  10.5
1   2    Ben  10.5
2   3  Cassa  12.0
3   4  David  12.0
```

data2.parquet

```
   id   name   age
0   5   Emma  13.0
1   6  Frank  10.5
2   7   Gino  12.0
```

外部データソースParquet S3を利用するための定義

１つのファイルをデータソースとする外部テーブルの定義

```
CREATE EXTENSION parquet_s3_fdw;
CREATE SERVER parquet_s3_srv FOREIGN DATA WRAPPER parquet_s3_fdw OPTIONS (use_minio 'true');
CREATE USER MAPPING FOR public SERVER parquet_s3_srv OPTIONS (user 'minioadmin', password
'minioadmin');

CREATE FOREIGN TABLE data1__parquet__0 ( id INT8, name TEXT, age FLOAT) SERVER parquet_s3_srv
OPTIONS ( filename 's3://test-bucket/dir1/data1.parquet');
```

外部データソースParquet S3のデータを検索

１つのファイルを検索

```
SELECT * FROM data1__parquet__0;
EXPLAIN VERBOSE SELECT * FROM data1__parquet__0;
```

```
pgspider=# SELECT * FROM data1__parquet__0;
 id | name  | age
----+-------+-----
  1 | Alex  | 10.5
  2 | Ben   | 10.5
  3 | Cassa |   12
  4 | David |   12
(4 rows)

pgspider=# EXPLAIN VERBOSE SELECT * FROM data1__parquet__0;
                                QUERY PLAN
---------------------------------------------------------------------------
 Foreign Scan on public.data1__parquet__0  (cost=0.00..10.00 rows=1000 width=48)
   Output: id, name, age
   Reader: Single File
   Row groups: 1
(4 rows)
```

外部データソースParquet S3を利用するための定義

ディレクトリ、複数ファイルをデータソースとする外部テーブルの定義

```
CREATE FOREIGN TABLE dir1__parquet__0 ( id INT8, name TEXT, age FLOAT) SERVER parquet_s3_srv OPTIONS
(dirname 's3://test-bucket/dir1');
```

Parquet S3　PGSpider経由で実行

ディレクトリ、複数ファイルを検索

```
SELECT * FROM dir1__parquet__0;
EXPLAIN VERBOSE SELECT * FROM dir1__parquet__0;
```

```
pgspider=# SELECT * FROM dir1__parquet__0;
XPLAIN SELECT * FROM dir1__parquet__0;
 id | name  | age
----+-------+-----
  1 | Alex  | 10.5
  2 | Ben   | 10.5
  3 | Cassa |   12
  4 | David |   12
  5 | Emma  |   13
  6 | Frank | 10.5
  7 | Gino  |   12
(7 rows)
```

```
pgspider=# EXPLAIN SELECT * FROM dir1__parquet__0;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Foreign Scan on dir1__parquet__0  (cost=0.00..10.00 rows=1000 width=48)
   Reader: Multifile
   Row groups:
     data1.parquet: 1
     data2.parquet: 1
(5 rows)
```

データソース接続モジュールFDWのデモまとめ

- データソースへの接続する設定手順を紹介した。
- 以下の３つのデータソースへの接続デモを紹介した。

01 MySQL

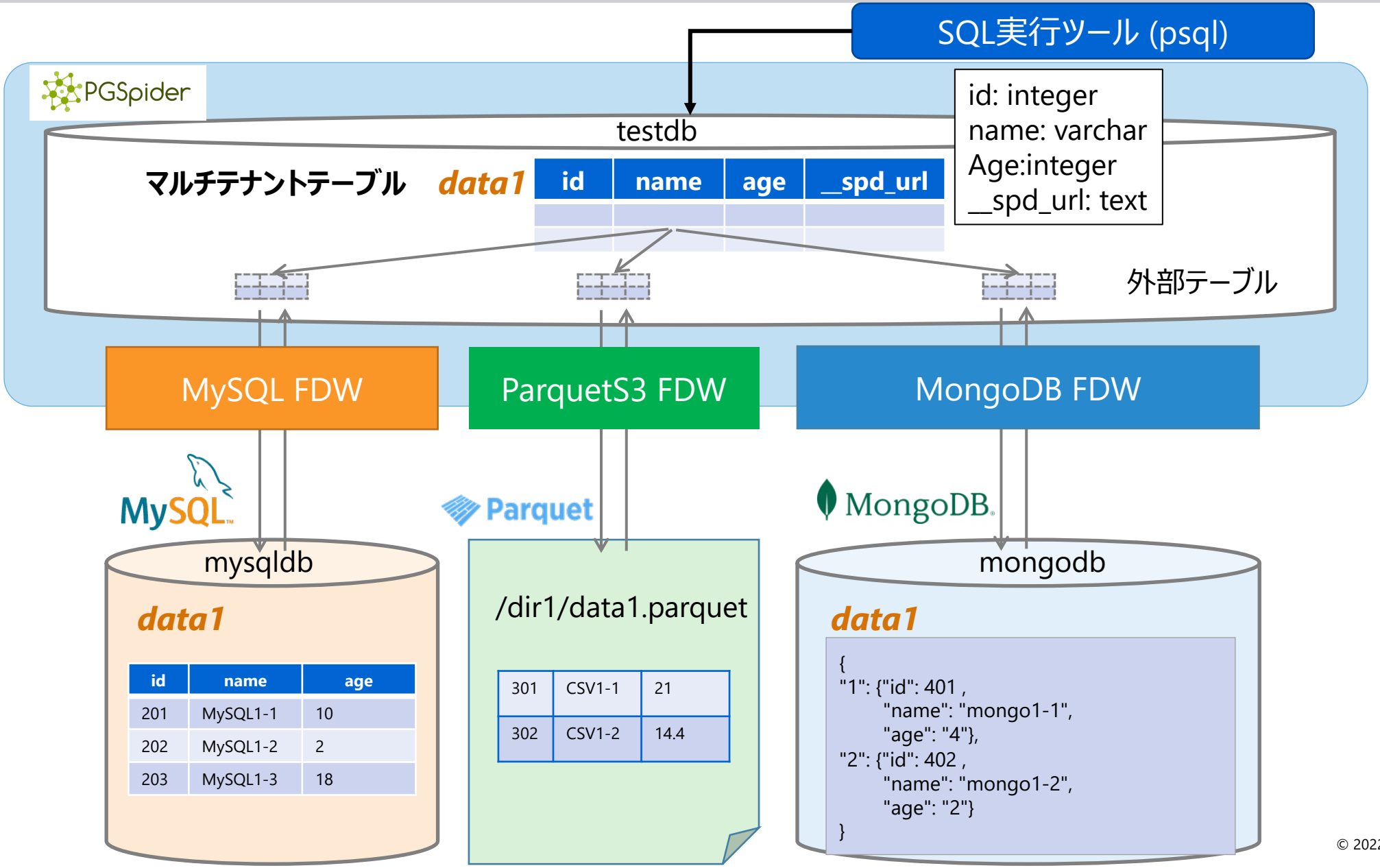
02 MongoDB

03 ParquetS3

04

PGSpider マルチテナントテーブル デモ

構築するデモシステム構成 - 各データソース定義とPGSpider上のテーブル定義



マルチテナントテーブル利用の準備

- SQL実行ツール(psql)を利用してPGSpiderへ接続

```
$ psql -p 4813 pgspiderdb
```

- 外部サーバ(マルチテナントテーブル用)拡張モジュールの組込み

```
CREATE EXTENSION pgspider_core_fdw;
```

- 外部サーバ(マルチテナントテーブル用)を定義する

```
CREATE SERVER pgspider FOREIGN DATA WRAPPER pgspider_core_fdw;
```

- 外部サーバ(マルチテナントテーブル用)への接続情報を設定する

```
CREATE USER MAPPING FOR CURRENT_USER SERVER pgspider;
```

PGSpider内部で利用する定義のため、接続先やアカウント情報等の設定は無し

マルチテナントテーブルの作成

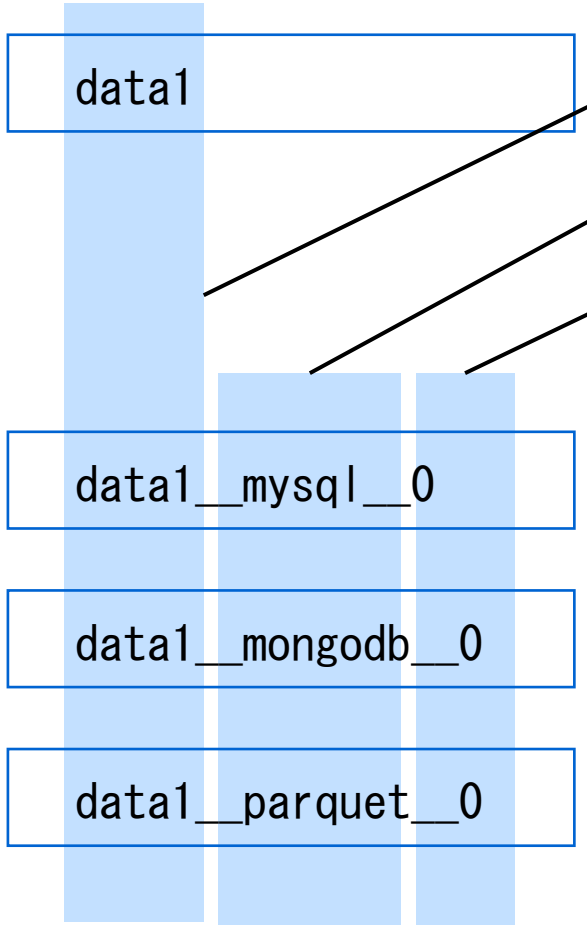
```
CREATE FOREIGN TABLE data1 (id int, name test, age float, __spd_url text) SERVER pgspider;
```

- テーブル
  - テーブル名の付けかたによってまとめる外部テーブルを決める

- 列
  - まとめる外部テーブルの列と同じ構成とする
  - 「__spd_url text」列を追加する
    - ノード名列として外部データソースを表しており、内部でも利用するため必須

マルチテナントテーブルにおけるテーブル名の関係

- マルチテナントテーブル
- 外部テーブル



まとめるマルチテナントテーブル名と同じ名前

サーバを示す名前

同じサーバの複数のテーブルを参照する場合は
番号を増やしていく

マルチテナントテーブルのデータ取得

- 外部データソースの確認

```
SELECT * FROM data1__mysql__0;
SELECT * FROM data1__mongodb__0;
SELECT * FROM data1__parquet__0;
```

data1__mysql__0

```
 id |  name  | age
----+--------+------
  1 | Allen  |   12
  2 | Becky  | 10.5
  3 | Calvin |   12
  4 | Daniel |   11
(4 rows)
```

data1__mongodb__0

```
 id | name  | age
----+-------+------
  3 | Cindy | 10.5
  1 | Ann   |   10
  2 | Ben   | 10.5
(3 rows)
```

data1__parquet__0

```
 id | name  | age
----+-------+------
  1 | Alex  | 10.5
  2 | Ben   | 10.5
  3 | Cassa |   12
  4 | David |   12
(4 rows)
```

- マルチテナントテーブルの確認

```
SELECT * FROM data1;
```

data1

```
 id |  name  | age  |    __spd_url
----+--------+------+-----------------
  1 | Allen  |   12 | /mysql/
  2 | Becky  | 10.5 | /mysql/
  3 | Calvin |   12 | /mysql/
  4 | Daniel |   11 | /mysql/
  1 | Alex   | 10.5 | /parquet_s3_srv/
  2 | Ben    | 10.5 | /parquet_s3_srv/
  3 | Cassa  |   12 | /parquet_s3_srv/
  4 | David  |   12 | /parquet_s3_srv/
  3 | Cindy  | 10.5 | /mongodb/
  1 | Ann    |   10 | /mongodb/
  2 | Ben    | 10.5 | /mongodb/
(11 rows)
```

プッシュダウンによる高速化の対応 データソースが一つの場合

- 平均(avg())のプッシュダウン

```
# SELECT avg(age) FROM data1__mysql__0;
  avg
--------
 11.375
(1 row)
```

- EXPLAINを用いたプッシュダウンの様子
(今回プッシュダウンに対応しているのはMySQL FDW)

```
# EXPLAIN VERBOSE SELECT avg(age) FROM data1__mysql__0;
                        QUERY PLAN
-----------------------------------------------------------
 Foreign Scan  (cost=106.40..142.03 rows=1 width=8)
   Output: (avg(age))
   Local server startup cost: 10
   Remote query: SELECT avg(`age`) FROM `mysqldb`.`data1`
(4 rows)
```

プッシュダウンによる高速化の対応 データソースが分散している場合

- 平均(avg())のプッシュダウン

```
# SELECT avg(age) FROM data1;
        avg
---------------------
 11.045454545454545
(1 row)
```

- EXPLAINを用いたプッシュダウンの様子
  (今回プッシュダウンに対応しているのはMySQL FDW)

```
# EXPLAIN VERBOSE SELECT avg(age) FROM data1;
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------
 Foreign Scan  (cost=0.00..0.00 rows=1 width=8)
   Output: (avg(age))
   Node: mongodb / Status: Alive
     Agg push-down: no
     Foreign Namespace: mongodb.data1
     Query document: { "pipeline" : [ { "$project" : { "age" : { "$numberInt" : "1" } } } ] }
   Node: mysql / Status: Alive
     Agg push-down: yes
     Local server startup cost: 10
     Remote query: SELECT count(`age`), sum(`age`) FROM `mysqldb`.`data1`
   Node: parquet_s3_srv / Status: Alive
     Agg push-down: no
     Reader: Single File
     Row groups: 1
(14 rows)
```

PGSpider マルチテナントテーブル デモ まとめ

- マルチテナントテーブル機能
  - 複数の外部データソースをひとつの仮想テーブルにまとめてアクセス

- プッシュダウンによる高速化
  - データソースが複数に分散していても効率の良い方式に自動的に変換

05

まとめ

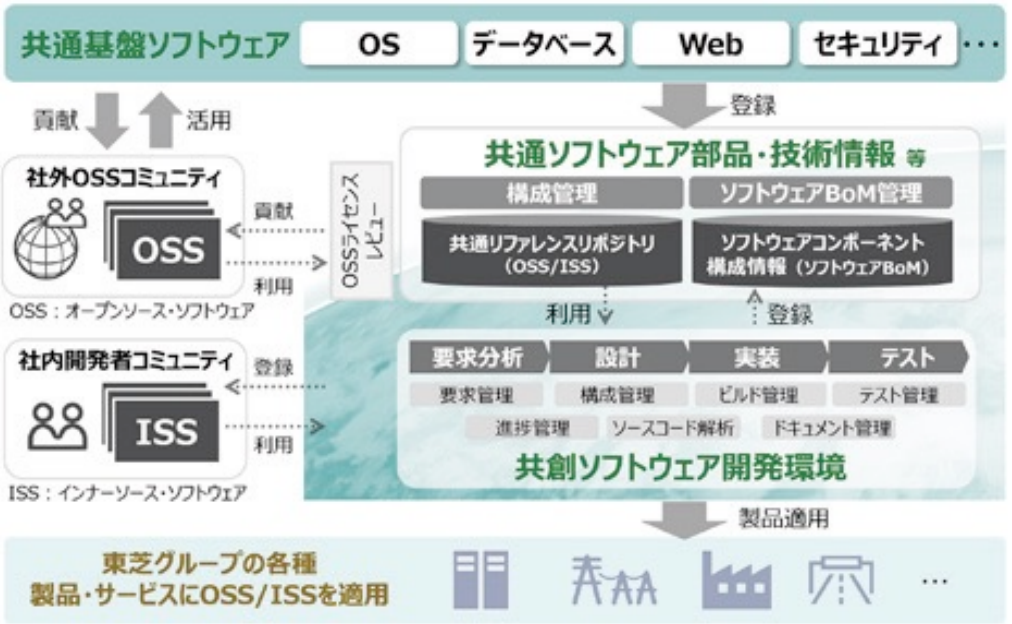
まとめ

- 分散SQLエンジン PGSpider によってデータ仮想化を実現
  - 多様なデータソースへのアクセス機能
  - 複数データソースの統合機能
  - データ検索の高速化
    - パラレル処理
    - プッシュダウン

- PGSpiderはオープンソースとして公開中！
  - GitHub https://github.com/pgspider
  - 紹介記事(Qiita) https://qiita.com/kanegoon/items/d0c344de8b62559bb493

仲間を募集しています！

株式会社 **東芝 ソフトウェア技術センター** では、本発表で紹介するような**オープンソースソフトウェア開発**をもとに、**様々な事業を横断的**に支えるための**基盤技術開発**に興味のある人を募集しております。



より詳しい情報をWEBで公開していますので是非ご覧ください。
https://www.global.toshiba/jp/technology/corporate/swc.html