第九届PostgreSQL中国技术大会

2019 PostgreSQL Conference China

开源驱动 自主研发

主办：PostgreSQL中文社区

协办：ITPUB

2019年11月29日-30日

北京维景国际大酒店

FDW原理与应用

冯若航

探探 PostgreSQL DBA

大纲

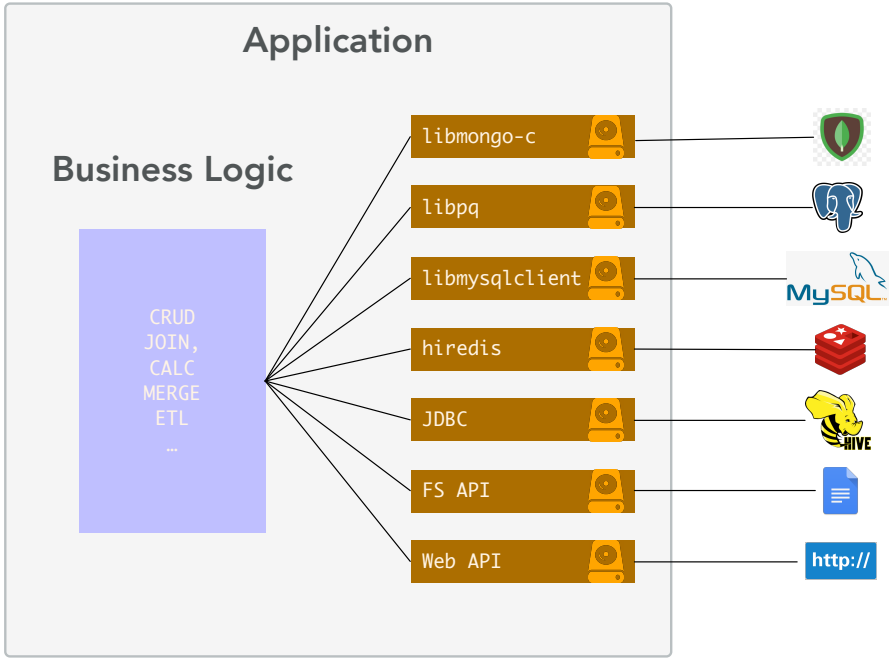
- FDW 意义
- FDW 前生今世
- FDW 走马观花
- FDW 实例 ： file_fdw
- FDW 实例 ： postgres_fdw
- FDW 工作原理
- FDW 如何编写

FDW 的意义

各种强大的数据组件层出不穷

应用成为各种数据系统间的搬运工

应对各种异构的接口



用于构建数据系统的工具正在变得越来越好，但还缺少一个重要的组件：与Unix shell类似的数据库集成设施。

即一种声明式的，简单的，用于组装数据系统的高级语言，数据库世界的巴别塔

```
postgresql | elasticsearch
```

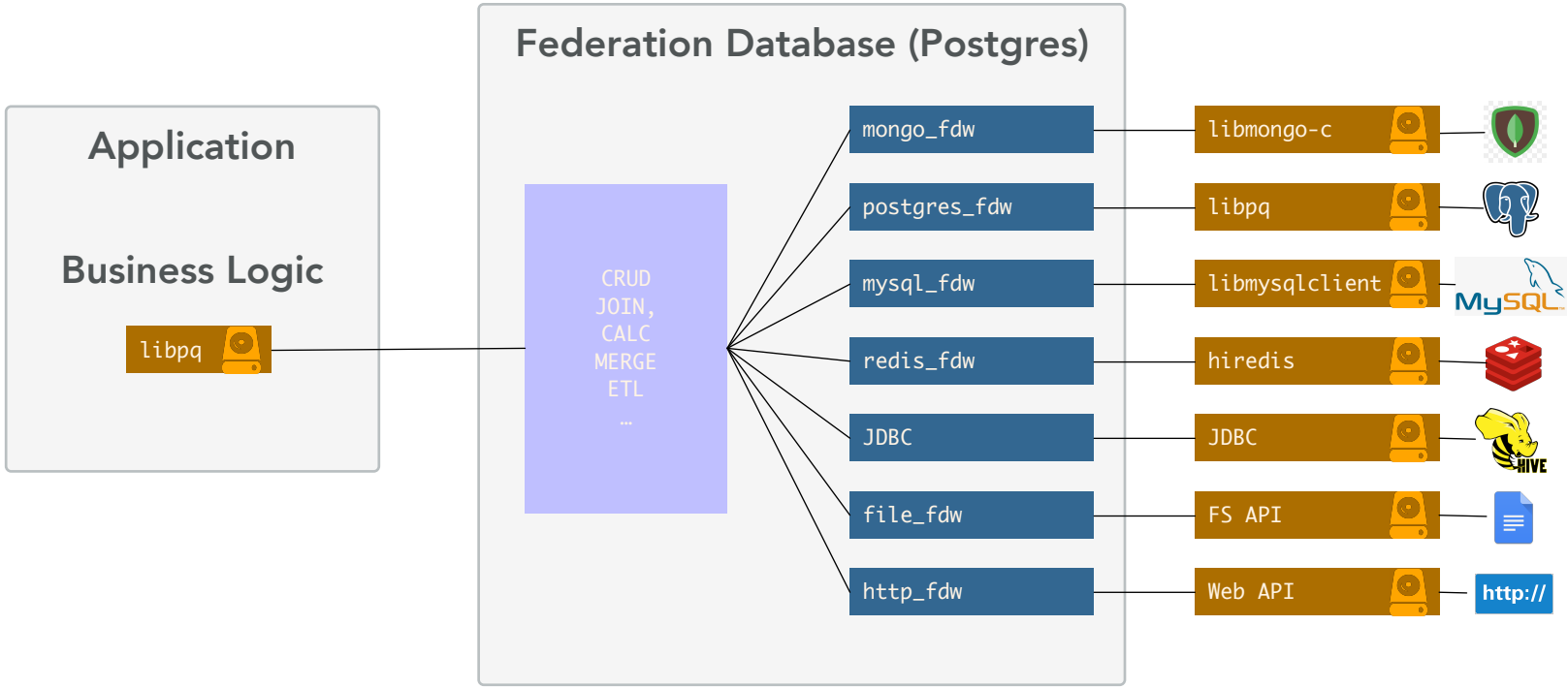
自动创建pgsql中表的全文索引?

SQL更像是数据系统世界的“世界通用语”

```
INSERT INTO redis.daily_summary
    SELECT * FROM pgsql.metadata mt
             JOIN mysql.realtime_counters USING(id)
             JOIN mongo.history_counters  USING(id)
```

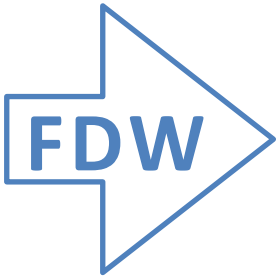
PostgreSQL FDW提供了组装这类数据系统的基础设施，是一项很有趣的技术

FDW 愿景展望


Federation Database (Postgres)
Application
Business Logic
libpq
CRUD
JOIN,
CALC
MERGE
ETL
…
mongo_fdw
postgres_fdw
mysql_fdw
redis_fdw
JDBC
file_fdw
http_fdw
libmongo-c
libpq
libmysqlclient
hiredis
JDBC
FS API
Web API
MySQL
HIVE
http://

FDW 愿景展望

ONE DATABASE RULE THEM ALL!



Federation Database / Polystore

Presto

AWS Athena

```
WITH logs
     AS (SELECT log_stream,
                message                                          AS
                order_processor_log,
                Regexp_extract(message, '.*orderId=(\d+) .*', 1) AS orderId,
                Regexp_extract(message, '(.*):.*', 1)            AS log_level
         FROM
     "lambda:cloudwatch"."/var/ecommerce-engine/order-processor".all_log_streams
         WHERE  Regexp_extract(message, '(.*):.*', 1) != 'WARN'),
     active_orders
     AS (SELECT *
         FROM   redis.redis_db.redis_customer_orders),
     order_processors
     AS (SELECT instanceid,
                publicipaddress,
                state.NAME
         FROM   awscmdb.ec2.ec2_instances),
     customer
     AS (SELECT id,
                email
         FROM   docdb.customers.customer_info),
     addresses
     AS (SELECT id,
                is_residential,
                address.street AS street
         FROM   docdb.customers.customer_addresses),
     shipments
     AS ( SELECT order_id,
                 shipment_id,
                 from_unixtime(cast(shipped_date as double)) as shipment_time,
                 carrier
        FROM lambda_ddb.default.order_shipments),
     payments
     AS ( SELECT "summary:order_id",
                 "summary:status",
                 "summary:cc_id",
                 "details:network"
        FROM "hbase".hbase_payments.transactions)

SELECT _key_            AS redis_order_id,
       customer_id,
       customer.email   AS cust_email,
       "summary:cc_id"  AS credit_card,
       "details:network" AS CC_type,
       "summary:status" AS payment_status,
       status           AS redis_status,
       addresses.street AS street_address,
       shipments.shipment_time as shipment_time,
       shipments.carrier as shipment_carrier,
       publicipaddress  AS ec2_order_processor,
       NAME             AS ec2_state,
       log_level,
       order_processor_log
FROM   active_orders
       LEFT JOIN logs
              ON logs.orderid = active_orders._key_
       LEFT JOIN order_processors
              ON logs.log_stream = order_processors.instanceid
       LEFT JOIN customer
              ON customer.id = customer_id
       LEFT JOIN addresses
              ON addresses.id = address_id
       LEFT JOIN shipments
              ON shipments.order_id = active_orders._key_
       LEFT JOIN payments
              ON payments."summary:order_id" = active_orders._key_
```

FDW 前生今世

SQL/MED： Management of External Data

SQL Standard ISO/IEC 9075-9:2008

实现者寥寥无几

PostgreSQL

IBM DB2

MariaDB

……

INTERNATIONAL STANDARD

ISO/IEC 9075-9

Fourth edition
2016-12-15

Information technology — Database languages — SQL —

Part 9:
Management of External Data (SQL/MED)

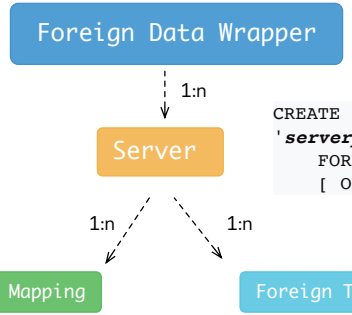
Technologies de l'information — Langages de base de données — SQL —

Partie 9: Gestion des données externes (SQL/MED)

FDW 基本模型

```
CREATE FOREIGN DATA WRAPPER name
    [ HANDLER handler_function | NO HANDLER ]
    [ VALIDATOR validator_function | NO VALIDATOR ]
    [ OPTIONS ( option 'value' [, ... ] ) ]
```

一个FDW处理同一类数据库，如MySQL，MongoDB



一个Server对应一台外部”数据库”

```
CREATE SERVER [ IF NOT EXISTS ] server_name [ TYPE 'server_type' ] [ VERSION
'server_version' ]
    FOREIGN DATA WRAPPER fdw_name
    [ OPTIONS ( option 'value' [, ... ] ) ]
```

User Mapping指明本地用户与远程用户的对应关系

Foreign Table 对应一个远程数据对象（表，文件，URL……）

```
CREATE USER MAPPING [ IF NOT EXISTS ] FOR { user_name | USER | CURRENT_USER | PUBLIC }
    SERVER server_name
    [ OPTIONS ( option 'value' [ , ... ] ) ]
```

FDW 前生今世

- 2008-10-27 PostgreSQL 8.4 开始提供对SQL/MED标准进行支持的基础设施
- 2011-09-12 PostgreSQL 9.1 提供了外部表机制，支持读取外部数据，提供官方扩展`file_fdw`
- 2013-09-09 PostgreSQL 9.3 支持FDW写入（对外部表做增删改），提供官方扩展`postgres_fdw`
- 2014-12-18 PostgreSQL 9.4 支持在外部表上创建触发器
- 2016-01-07 PostgreSQL 9.5 添加了`IMPORT FOREIGN SCHEMA`功能，允许外部表上的`CHECK`约束，允许外部表参与继承，允许将JOIN下推
- 2016-09-29 PostgreSQL 9.6 `postgres_fdw`支持将增删改/JOIN操作下推。
- 2017-10-05 PostgreSQL 10 `file_fdw`支持以任意程序输出作为输入来源，`postgres_fdw`支持聚合函数下推。
- 2018-10-18 PostgreSQL 11 `postgres_fdw`对分区表进行支持，并允许将带JOIN的远程增删改下推。
- 2019-10-03 PostgreSQL 12 `postgres_fdw`支持下推`ORDER BY/LIMIT`子句。

FDW 走马观花

CREATE FORIEIGN DATA WRAPPER

CREATE SERVER

CREATE USER MAPPING

CREATE FOREIGN TABLE

```
CREATE FOREIGN DATA WRAPPER file_fdw;
CREATE SERVER IF NOT EXISTS pg_log FOREIGN DATA WRAPPER file_fdw;
CREATE USER MAPPING FOR postgres SERVER pg_log;
CREATE FOREIGN TABLE pg_log_monday
    (
        log_time                timestamptz,
        user_name               text,
        database_name           text,
        process_id              integer,
        ....................
        location                text,
        application_name        text
        ) SERVER pg_log OPTIONS (filename '/pg/data/log/postgresql-Mon.csv', format 'csv');
SELECT * FROM pg_log_monday;
```

log_time	...	.	...	connect...	session_id		.	session_start_time	.			.	message
2019-11-18 12:09:54.279000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	1	SET	2019-11-18 12:09:54.000000	7/1560	0	LOG	00000	execute <unnamed>: SET extra_float_digits = 3
2019-11-18 12:09:54.279000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	2	SET	2019-11-18 12:09:54.000000	7/1561	0	LOG	00000	execute <unnamed>: SET application_name = 'PostgreSQL JDBC Driver'
2019-11-18 12:09:54.281000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	3	SELECT	2019-11-18 12:09:54.000000	7/1562	0	LOG	00000	execute <unnamed>: select version()
2019-11-18 12:09:54.294000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	4	SELECT	2019-11-18 12:09:54.000000	7/1564	0	LOG	00000	execute <unnamed>: select current_database() as a, current_schemas(false) as b
2019-11-18 12:09:54.301000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	5	SELECT	2019-11-18 12:09:54.000000	7/1565	0	LOG	00000	execute <unnamed>: SELECT e.typdelim FROM pg_catalog.pg_type t, pg_catalog.pg_type e WHE…
2019-11-18 12:09:54.332000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	6	SELECT	2019-11-18 12:09:54.000000	7/1566	0	LOG	00000	execute <unnamed>: select current_database(), current_schema(), current_user
2019-11-18 12:09:54.375000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	7	SELECT	2019-11-18 12:09:54.000000	7/1567	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ datname as name,⏎ D.descript…
2019-11-18 12:09:54.395000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	8	SELECT	2019-11-18 12:09:54.000000	7/1568	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ N.xmin as state_number,⏎ nsp…
2019-11-18 12:09:54.400000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	9	SELECT	2019-11-18 12:09:54.000000	7/1569	0	LOG	00000	execute <unnamed>: SELECT n.nspname = ANY(current_schemas(true)), n.nspname, t.typname F…
2019-11-18 12:09:54.402000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	10	SELECT	2019-11-18 12:09:54.000000	7/1570	0	LOG	00000	execute <unnamed>: SELECT typinput='array_in'::regproc, typtype FROM pg_catalog.pg_typ…
2019-11-18 12:09:54.411000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	11	SELECT	2019-11-18 12:09:54.000000	7/1571	0	LOG	00000	execute <unnamed>: select R.oid::bigint as role_id, rolname as role_name,⏎ rolsuper is_…
2019-11-18 12:09:54.416000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	12	SELECT	2019-11-18 12:09:54.000000	7/1572	0	LOG	00000	execute <unnamed>: SELECT e.typdelim FROM pg_catalog.pg_type t, pg_catalog.pg_type e WHE…
2019-11-18 12:09:54.422000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	13	SELECT	2019-11-18 12:09:54.000000	7/1573	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ datname as name,⏎ D.descript…
2019-11-18 12:09:54.428000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	14	SELECT	2019-11-18 12:09:54.000000	7/1574	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ N.xmin as state_number,⏎ nsp…
2019-11-18 12:09:54.436000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	15	SELECT	2019-11-18 12:09:54.000000	7/1575	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ N.xmin as state_number,⏎ nsp…

FDW 走马观花

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

Generic SQL Database Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
ODBC	Native		github			CartoDB took over active development of the ODBC FDW for PG 9.5+
JDBC	Native		github			Not maintained ?
JDBC2	Native		github			
SQL_Alchemy	Multicorn	PostgreSQL	GitHub	PGXN	documentation	Can be used to access data stored in any database supported by the sqlalchemy python toolkit.
VirtDB	Native	GPL	GitHub			A generic FDW to access VirtDB data sources (SAP ERP, Oracle RDBMS)

Specific SQL Database Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
PostgreSQL	Native	PostgreSQL	git.postgresql.org		documentation	
Oracle	Native	PostgreSQL	github	PGXN	website	
MySQL	Native		github	PGXN	example	FDW for MySQL
Informix	Native	PostgreSQL	github			
Firebird	Native	PostgreSQL	github	PGXN	README	version 1.1 released (2019-05)
SQLite	Native		github			An FDW for SQLite3 (read-only)
SQLite	Native	PostgreSQL	github	PGXN	README	An FDW for SQLite3 (write support and several pushdown optimization)
Sybase / MS SQL Server	Native		github	PGXN		An FDW for Sybase and Microsoft SQL server
MonetDB	Native		github			

SQL的

NoSQL Database Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
BigTable or HBase	Native Rust Binding (RPGFFI)	MIT	Github			
Cassandra	Multicorn	MIT	Github	Rankactive		
Cassandra2	Native	MIT	Github			
Cassandra	Multicorn	PostgreSQL	Github			
ClickHouse	Multicorn	BSD	Github		README	
ClickHouse	Native	Apache	Github		README	
CouchDB	Native	PostgreSQL	Github	PGXN		Original version
CouchDB	Native	PostgreSQL	Github			golgauth version (9.1 - 9.2+ compatible)
GridDB	Native	PostgreSQL	Github		README	
InfluxDB	Native	PostgreSQL	Github		README	
Kafka	Native	PostgreSQL	GitHub		README	
Kyoto Tycoon	Native	MIT	Github			
MongoDB	Native	GPL3+	Github	PGXN	README	EDB version
MongoDB	Multicorn	MIT	Github			
MongoDB	Multicorn		Github			Yet Another Postgres FDW for MongoDB
Neo4j	Multicorn	GPLv3	Github		README	FWD for Neo4j and also add a Cypher function to Pg
Neo4j	Native	?	Github			
Quasar	Native	Apache	Github			
Redis	Native	PostgreSQL	Github			
Redis	Native	BSD	Github			
RethinkDB	Multicorn	MIT	Github		blog	
Riak	Multicorn	PostgreSQL	Github			
WhiteDB	Native	MIT	Github			

NoSQL的

BigData的

Big Data Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
Elasticsearch	Multicorn	MIT	GitHub			Supports up to PG 11, ES 7.
Google BigQuery	Multicorn	MIT	GitHub		Documentation	bigquery_fdw is a BigQuery FDW compatible with PostgreSQL >= 9.5
file_fdw-gds (Hadoop)	Native		GitHub			Hadoop file_fdw is a slightly modified version of PostgreSQL 9.3's file_fdw module.
Hadoop	Native	PostgreSQL	Bitbucket			Allows read and write access to HBase as well as to HDFS via Hive.
HDFS	Native	Apache	GitHub			
Hive	Multicorn		GitHub			Used to access Apache Hive tables.
Hive / ORC File	Native		GitHub			
Impala	Native	BSD	GitHub			
Apache Arrow	Native	GPLv2	GitHub			A part of PG-Strom feature; as a columnar data source with support of SSD-to-GPU Direct SQL

LocalFile的

File Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
CSV	Native	PostgreSQL	git.postgresql.org		documentation	Delivered as an official extension of PostgreSQL 9.1 / example / Another example
CSV	Multicorn	PostgreSQL	GitHub	PGXN	documentation	Each column defined in the table will be mapped, in order, against columns in the CSV file.
CSV / Text Array	Native		GitHub		How to	Another CSV wrapper
CSV / Fixed-length	Native		GitHub			
CSV / gzipped	Multicorn		GitHub			PostgreSQL Foreign Data Wrapper for gzipped cvs file
Compressed File	Native		GitHub			
Document Collection	Native	PostgreSQL	GitHub		wiki	
JSON	Native	GPL3	GitHub		Example	
Multi-File	Multicorn	PostgreSQL	GitHub	PGXN	doc	Access data stored in various files in a filesystem. The files are looked up based on a pattern, and parts of the file's path are mapped to various columns, as well as the file's content itself.
Multi CDR	Native	PostgreSQL	GitHub	PGXN		
Parquet	Native	PostgreSQL	GitHub			Foreign data wrapper for reading Parquet files using libarrow/libparquet
pg_dump	Native	New BSD	GitHub			Allows querying of data directly against Postgres custom format files created by pg_dump
TAR Files	Native		GitHub			
XML	Multicorn	PostgreSQL	GitHub	PGXN		
ZIP Files	Native		GitHub			

Cloud的

Specific Web Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
Database.com	Multicorn	BSD	GitHub			
Dun & Badstreet	Multicorn	PostgreSQL	GitHub			Access to the Data Universal Numbering System (DUNS)
DynamoDB	Multicorn	GPL	GitHub			
Facebook	Multicorn		GitHub			
Fixer.io	based on www_fdw		GitHub			
Google	Multicorn	PostgreSQL	GitHub	PGXN		
Heroku dataclips	Native	PostgreSQL	GitHub			
Keycloak	Multicorn	MIT	GitHub	PGXN	README	Direct database integration with the Keycloak open-source Identity/Access Management solution.
Mailchimp	Multicorn	PostgreSQL	GitHub			Beta
Parse	Multicorn	MIT	GitHub			
S3	Native	PostgreSQL	GitHub	PGXN		
S3CSV	Multicorn	GPL 3	GitHub			This is meant to replace s3_fdw that does is not supported on PostgreSQL version 9.2+
Telegram	Multicorn	PostgreSQL	GitHub			telegram_fdw is a Telegram BOT implemented using the PostgreSQL foreign data wrapper interface.
Twitter	Native	PostgreSQL	GitHub	PGXN		A wrapper fetching text messages from Twitter over the Internet and returning a table
Treasure Data	Native	Apache	GitHub	PGXN		A FDW for Treasure Data internally using a Rust library
Treasure Data	Multicorn	Apache	GitHub			
Google Spreadsheets	Multicorn	MIT	GitHub			
Open Weather Map	Multicorn	MIT	GitHub			A FDW for Open Weather Map (single city)

百花齐放

Geo Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
GDAL/OGR	Native	MIT	GitHub			A wrapper for data sources with a GDAL/OGR driver, including databases like Oracle, Informix, SQLite, SQL Server, ODBC as well as file formats like Shape, FGDB, MapInfo, CSV, Excel, OpenOffice, OpenStreetMap PBF and XML, OGC WebServices, and more Spatial columns are linked in as PostGIS geometry if PostGIS is installed.
Geocode / GeoJSON	Multicorn	GPL	GitHub			a collection of PostGIS-related foreign data wrappers
Open Street Map PBF	Native	PostgreSQL	GitHub			

LDAP Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
LDAP	Native		GitHub	PGXN		Allows to query an LDAP server and retrieve data from some pre-configured Organizational Unit
LDAP	Multicorn	PostgreSQL	GitHub	PGXN	documentation	

Generic Web Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
Git	Multicorn	PostgreSQL	GitHub	PGXN		
Git	Native	MIT	GitHub			
ICAL	Multicorn	PostgreSQL	GitHub		pdf	
IMAP	Multicorn	PostgreSQL	GitHub	PGXN	documentation	
RSS	Multicorn	PostgreSQL	GitHub	PGXN	documentation	This fdw can be used to access items from an rss feed.
www	Native	PostgreSQL	GitHub	PGXN	wiki	Allows to query different web services

Operating System Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
Docker	Multicorn	Expat	GitHub			
Log files	Multicorn	PostgreSQL	GitHub			
OpenStack / Telemetry	Multicorn	PostgreSQL	GitHub			
OS Query	Multicorn	PostgreSQL	GitHub			Like Facebook's OSQuery, but for Postgres
Passwd	Native	PostgreSQL	GitHub			reads linux/unix password and group files.
Process	Multicorn	PostgreSQL	GitHub			A foreign datawrapper for querying system stats based on statgrab
Environment Variables	Multicorn	MIT	GitHub			envFDW is a forign data wrapper for processing environment variables

Exotic Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
faker_fdw	Multicorn	PostgreSQL	GitHub			faker_fdw is a foreign data wrapper for PostgreSQL that generates fake data.
fdw_fdw	Multicorn	PostgreSQL	GitHub			the Meta FDW ! reads this page and returns the list of all the FDW
PPG	Native		GitHub			distributed parallel query engine, based on fdw and hooks of PG
Open Civic Data	Multicorn	Expat	GitHub			
Phillips Hue Lighting Systems	Multicorn	MIT	GitHub			
Random Number	Multicorn	PostgreSQL	GitHub			A random number generator foreign data wrapper for postgres
Rotfang	Multicorn	Native	BitBucket	PostgreSQL	slides	Advanced random number generator
Template Tables	Native	BSD	GitHub			PostgreSQL data wrapper for template tables - any DML and SELECT operations are disallowed
VMware vSphere	Multicorn	MIT	GitHub			A PostgreSQL FDW to query your VMware vSphere service

Example Wrappers

Data Source	Type	License	Code	Install	Doc	Notes
Dummy	Native	BSD	GitHub			Readable null FDW for testing
Hello World			GitHub			
Black Hole			bitbucket			a skeleton FDW pre-populated with relevant excerpts from the documentation

FDW实例

file_fdw　　postgres_fdw

原装品质，由PostgreSQL全球开发组维护

原版自带，PostgreSQL默认自带这两个扩展

File FDW

访问外部表（文件，如CSV）
读取任意程序的输出并将其转换为数据表
只读

Version	Date	Feature
9.1	2011-09	`file_fdw`出现，允许 `CREATE FOREIGN TABLE`，只读。
10	2017-10	`file_fdw`可以使用 `program` 参数执行任意程序获取输入

访问CSV格式的PG日志

与外部软件（Excel，其他数据库）交换数据

```
create foreign table pg_log
(
    log_time               timestamp(3) with time zone,
    user_name              text,
    database_name          text,
    process_id             integer,
    connection_from        text,
    session_id             text   not null,
    session_line_num       bigint not null,
    command_tag            text,
    session_start_time     timestamp with time zone,
    virtual_transaction_id text,
    transaction_id         bigint,
    error_severity         text,
    sql_state_code         text,
    message                text,
    detail                 text,
    hint                   text,
    internal_query         text,
    internal_query_pos     integer,
    context                text,
    query                  text,
    query_pos              integer,
    location               text,
    application_name       text
) server pg_log
options ( filename '/pg/data/log/postgresql-Mon.csv', format 'csv' );;
```

log_time	...	.	...	connect...	session_id		.	session_start_time	.			.	message
2019-11-18 12:09:54.279000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	1	SET	2019-11-18 12:09:54.000000	7/1560	0	LOG	00000	execute <unnamed>: SET extra_float_digits = 3
2019-11-18 12:09:54.279000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	2	SET	2019-11-18 12:09:54.000000	7/1561	0	LOG	00000	execute <unnamed>: SET application_name = 'PostgreSQL JDBC Driver'
2019-11-18 12:09:54.281000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	3	SELECT	2019-11-18 12:09:54.000000	7/1562	0	LOG	00000	execute <unnamed>: select version()
2019-11-18 12:09:54.294000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	4	SELECT	2019-11-18 12:09:54.000000	7/1564	0	LOG	00000	execute <unnamed>: select current_database() as a, current_schemas(false) as b
2019-11-18 12:09:54.301000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	5	SELECT	2019-11-18 12:09:54.000000	7/1565	0	LOG	00000	execute <unnamed>: SELECT e.typdelim FROM pg_catalog.pg_type t, pg_catalog.pg_type e WHE…
2019-11-18 12:09:54.332000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	6	SELECT	2019-11-18 12:09:54.000000	7/1566	0	LOG	00000	execute <unnamed>: select current_database(), current_schema(), current_user
2019-11-18 12:09:54.375000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	7	SELECT	2019-11-18 12:09:54.000000	7/1567	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ datname as name,⏎ D.descript…
2019-11-18 12:09:54.395000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	8	SELECT	2019-11-18 12:09:54.000000	7/1568	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ N.xmin as state_number,⏎ nsp…
2019-11-18 12:09:54.400000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	9	SELECT	2019-11-18 12:09:54.000000	7/1569	0	LOG	00000	execute <unnamed>: SELECT n.nspname = ANY(current_schemas(true)), n.nspname, t.typname F…
2019-11-18 12:09:54.402000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	10	SELECT	2019-11-18 12:09:54.000000	7/1570	0	LOG	00000	execute <unnamed>: SELECT typinput='array_in'::regproc, typtype FROM pg_catalog.pg_typ…
2019-11-18 12:09:54.411000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	11	SELECT	2019-11-18 12:09:54.000000	7/1571	0	LOG	00000	execute <unnamed>: select R.oid::bigint as role_id, rolname as role_name,⏎ rolsuper is_…
2019-11-18 12:09:54.416000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	12	SELECT	2019-11-18 12:09:54.000000	7/1572	0	LOG	00000	execute <unnamed>: SELECT e.typdelim FROM pg_catalog.pg_type t, pg_catalog.pg_type e WHE…
2019-11-18 12:09:54.422000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	13	SELECT	2019-11-18 12:09:54.000000	7/1573	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ datname as name,⏎ D.descript…
2019-11-18 12:09:54.428000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	14	SELECT	2019-11-18 12:09:54.000000	7/1574	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ N.xmin as state_number,⏎ nsp…
2019-11-18 12:09:54.436000	postgres	test	30916	10.10.10.1:60825	5dd28a12.78c4	15	SELECT	2019-11-18 12:09:54.000000	7/1575	0	LOG	00000	execute <unnamed>: select N.oid::bigint as id,⏎ N.xmin as state_number,⏎ nsp…

除了CSV文件，任何输出csv的程序都可以作为输入

```
CREATE FOREIGN TABLE etc_password (
    username  TEXT,
    password  TEXT,
    user_id   INTEGER,
    group_id  INTEGER,
    user_info TEXT,
    home_dir  TEXT,
    shell     TEXT
) SERVER fs OPTIONS (
    PROGRAM $$awk -F: 'NF && !/^[:space:]*#/ {print $1,$2,$3,$4,$5,$6,$7}' OFS='\037' /etc/passwd$$,
    FORMAT 'csv', DELIMITER E'\037'
);
```

执行任意的Shell命令获取输出
比如打印出 /etc/passwd

username	password	user_id	group_id	user_info	home_dir	shell
root	x	0	0	root	/root	/bin/bash
bin	x	1	1	bin	/bin	/sbin/nologin
daemon	x	2	2	daemon	/sbin	/sbin/nologin
adm	x	3	4	adm	/var/adm	/sbin/nologin
lp	x	4	7	lp	/var/spool/lpd	/sbin/nologin
sync	x	5	0	sync	/sbin	/bin/sync
shutdown	x	6	0	shutdown	/sbin	/sbin/shutdown
halt	x	7	0	halt	/sbin	/sbin/halt

```
CREATE EXTENSION file_fdw;
CREATE SERVER fs FOREIGN DATA WRAPPER file_fdw;

CREATE FOREIGN TABLE adcode
    (
        code         BIGINT,
        parent       BIGINT,
        name         VARCHAR(64),
        level        VARCHAR(16),
        rank         INTEGER,
        adcode       INTEGER,
        post_code    VARCHAR(8),
        area_code    VARCHAR(4),
        ur_code      VARCHAR(4),
        municipality BOOLEAN,
        virtual      BOOLEAN,
        dummy        BOOLEAN,
        longitude    FLOAT,
        latitude     FLOAT,
        center       GEOMETRY,
        province     VARCHAR(64),
        city         VARCHAR(64),
        county       VARCHAR(64),
        town         VARCHAR(64),
        village      VARCHAR(64)
) SERVER fs OPTIONS (PROGRAM $$curl -s https://raw.githubusercontent.com/Vonng/adcode/master/data/adcode/110101.csv$$,
FORMAT 'csv', DELIMITER ',');
```

或者通过ssh/curl/wget从网络获取数据

```
vonng=# select * from adcode LIMIT 20;
    code    |   parent    |       name        |  level  | rank | adcode | post_code | area_code | ur_code | municipality | virtual | dummy | longitude | latitude |                  center                  | province |  city  | county |       town        |     village
------------+-------------+-------------------+---------+------+--------+-----------+-----------+---------+--------------+---------+-------+-----------+----------+------------------------------------------+----------+--------+--------+-------------------+------------------
 110101000000 | 110100000000 | 东城区            | county  |    3 | 110101 |           | 010       |         | f            | f       | f     |  116.41649 | 39.928341 | 01010000001B47ACC5A71A5D409A44BDE0D3F64340 | 北京市   | 市辖区 | 东城区 |                   |
 110101001000 | 110101000000 | 东华门街道办事处  | town    |    4 | 110101 |           | 010       |         | f            | f       | f     | 116.407239 | 39.916094 | 0101000000F8A92A34101A5D40EA0775914 2F54340 | 北京市   | 市辖区 | 东城区 | 东华门街道办事处  |
 110101002000 | 110101000000 | 景山街道办事处    | town    |    4 | 110101 |           | 010       |         | f            | f       | f     | 116.410575 | 39.926781 | 01010000000B8635DDC461A5D40EFC682C2A0F64340 | 北京市   | 市辖区 | 东城区 | 景山街道办事处    |
 110101003000 | 110101000000 | 交道口街道办事处  | town    |    4 | 110101 |           |           |         | f            | f       | f     |            |          |                                          | 北京市   | 市辖区 | 东城区 | 交道口街道办事处  |
 110101004000 | 110101000000 | 安定门街道办事处  | town    |    4 | 110101 |           | 010       |         | f            | f       | f     | 116.413073 | 39.944171 | 01010000001477BCC9GF1A5D4072GA6798DAF84340 | 北京市   | 市辖区 | 东城区 | 安定门街道办事处  |
 110101005000 | 110101000000 | 北新桥街道办事处  | town    |    4 | 110101 |           |           |         | f            | f       | f     |            |          |                                          | 北京市   | 市辖区 | 东城区 | 北新桥街道办事处  |
 110101006000 | 110101000000 | 东四街道办事处    | town    |    4 | 110101 |           | 010       |         | f            | f       | f     | 116.424029 | 39.930311 | 01010000008FE3874A231B5D40F60D4C6E14F74340 | 北京市   | 市辖区 | 东城区 | 东四街道办事处    |
```

postgres_fdw

原装品质，可读可写可下推

ETL / Sharding / CMDB

Version	Date	Feature
9.3	2003-09	提供`postgres_fdw`，允许对外部表执行`INSERT, UPDATE, DELETE`
9.5	2016-01	`postgres_fdw`支持下推`JOIN`，支持`IMPORT FOREIGN SCHEMA`
9.6	2016-09	`postgres_fdw`支持下推带`JOIN`的`UPDATE|INSERT|DELETE`
10	2017-10	`postgres_fdw`支持下推聚合函数
11	2018-10	外部表提供了一系列对声明式分区的支持
12	2019-10	`postgres_fdw`支持下推`ORDER BY/LIMIT`

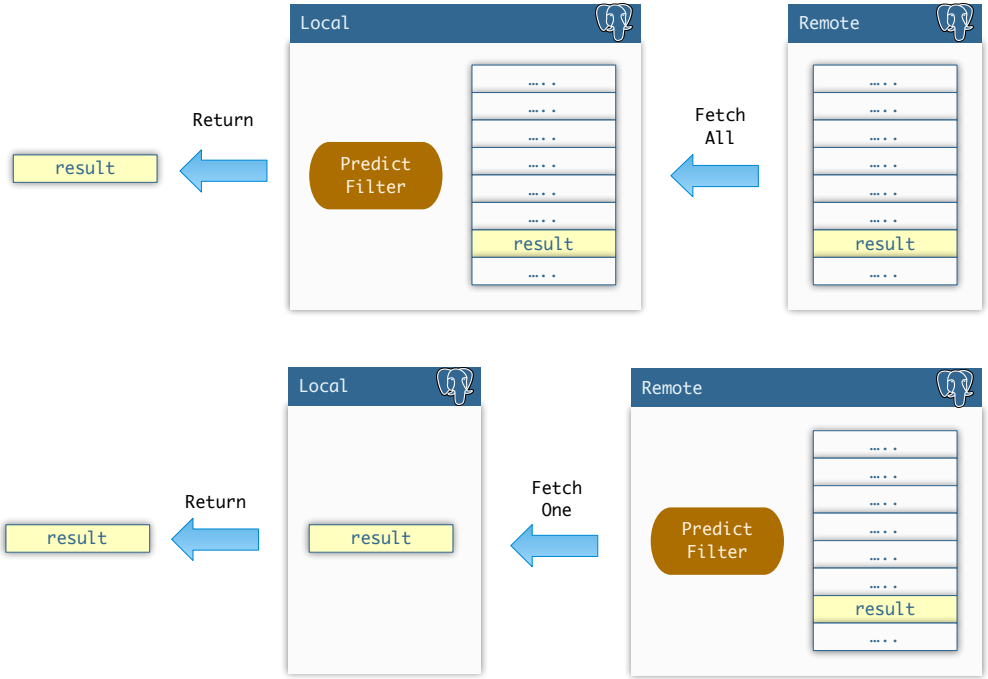
postgres_fdw Quick Start

```
-- node 2
CREATE TABLE users
(
    id   SERIAL PRIMARY KEY,
    name TEXT UNIQUE
);
INSERT INTO users (name) SELECT 'user #' || i FROM generate_series(1,100000) i;

-- node 1
CREATE EXTENSION postgres_fdw;
CREATE SERVER n2 FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host 'n2', port '5432', dbname 'test');
CREATE USER MAPPING FOR postgres SERVER n2 OPTIONS(user 'postgres', password 'postgres');
CREATE FOREIGN TABLE users(
    id integer,
    name text
) SERVER n2;
IMPORT FOREIGN SCHEMA public LIMIT TO (users) FROM SERVER n2 INTO public; -- alternate approach
```

```
-- execute on node 1
SELECT * FROM users LIMIT 200;


test=# EXPLAIN VERBOSE SELECT * FROM users LIMIT 200;
                                QUERY PLAN
-------------------------------------------------------------------------
 Foreign Scan on public.users  (cost=100.00..107.15 rows=200 width=36)
   Output: id, name
   Remote SQL: SELECT id, name FROM public.users LIMIT 200::bigint
```

```
15,"idle",2019-11-20 06:08:14 UTC,11/70,0,LOG,00000,"statement: START TRANSACTION ISOLATION LEVEL REPEATABLE READ",,,,,,,,,"postgres_fdw"
16,"DECLARE CURSOR",2019-11-20 06:08:14 UTC,11/70,0,LOG,00000,"execute <unnamed>: DECLARE c1 CURSOR FOR SELECT id, name FROM public.users LIMIT 200::bigint",,,,,,,,,"postgre
17,"idle in transaction",2019-11-20 06:08:14 UTC,11/70,0,LOG,00000,"statement: FETCH 100 FROM c1",,,,,,,,,"postgres_fdw"
18,"idle in transaction",2019-11-20 06:08:14 UTC,11/70,0,LOG,00000,"statement: FETCH 100 FROM c1",,,,,,,,,"postgres_fdw"
19,"idle in transaction",2019-11-20 06:08:14 UTC,11/70,0,LOG,00000,"statement: FETCH 100 FROM c1",,,,,,,,,"postgres_fdw"
20,"idle in transaction",2019-11-20 06:08:14 UTC,11/70,0,LOG,00000,"statement: CLOSE c1",,,,,,,,,"postgres_fdw"
21,"idle in transaction",2019-11-20 06:08:14 UTC,11/70,0,LOG,00000,"statement: COMMIT TRANSACTION",,,,,,,,,"postgres_fdw"
```

postgres_fdw pushdown



下推是FDW性能的决定性因素

谓词（Where条件），
聚合函数（avg, max），
连接子句 （JOIN）
TopN （ORDER x LIMIT x）

在本地判断需要取回大量数据
下推至远程执行，只取回极少量需要的结果

postgres_fdw pushdown

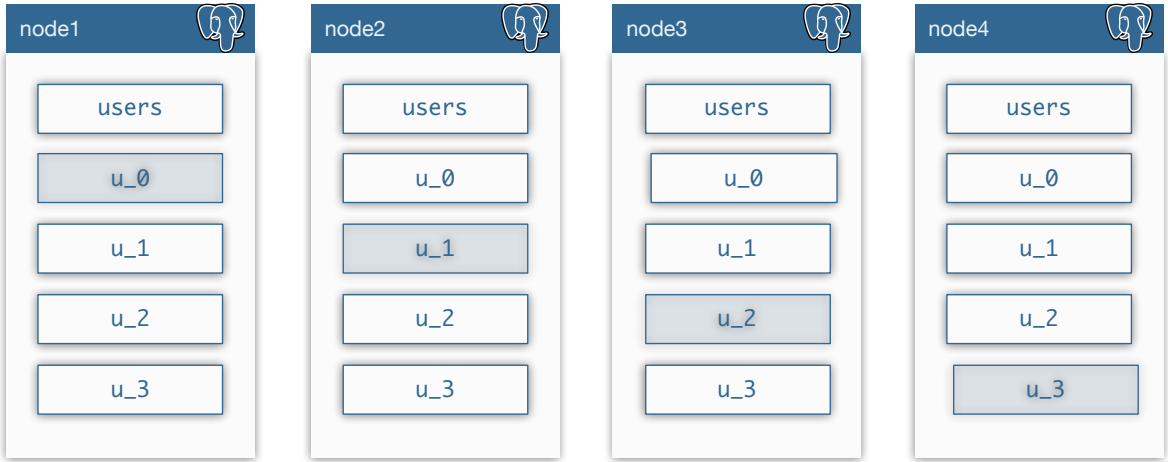
PostgresFDW支持很多下推

```
EXPLAIN VERBOSE SELECT * FROM users;
--   Remote SQL: SELECT id, username, password FROM public.users

EXPLAIN VERBOSE SELECT * FROM users WHERE id = 1234;
--   Remote SQL: SELECT id, username, password FROM public.users WHERE ((id = 1234))

EXPLAIN VERBOSE SELECT avg(id) FROM users;
--   Remote SQL: SELECT avg(id) FROM public.users

EXPLAIN VERBOSE SELECT * FROM users u JOIN pwd p USING(id) WHERE id = 1234;
--   Remote SQL: SELECT r1.id, r1.name, r2.pwd FROM (public.users r1 INNER JOIN public.pwd r2 ON (((r2.id = 1234)) AND ((r1.id = 1234)

EXPLAIN VERBOSE SELECT * FROM users u JOIN pwd p USING(id) ORDER BY id DESC LIMIT 1;
--   Remote SQL: SELECT r1.id, r1.name, r2.pwd FROM (public.users r1 INNER JOIN public.pwd r2 ON (((r1.id = r2.id))))
—                    ORDER BY r1.id DESC NULLS FIRST LIMIT 1::bigint
```

postgres_fdw based sharding

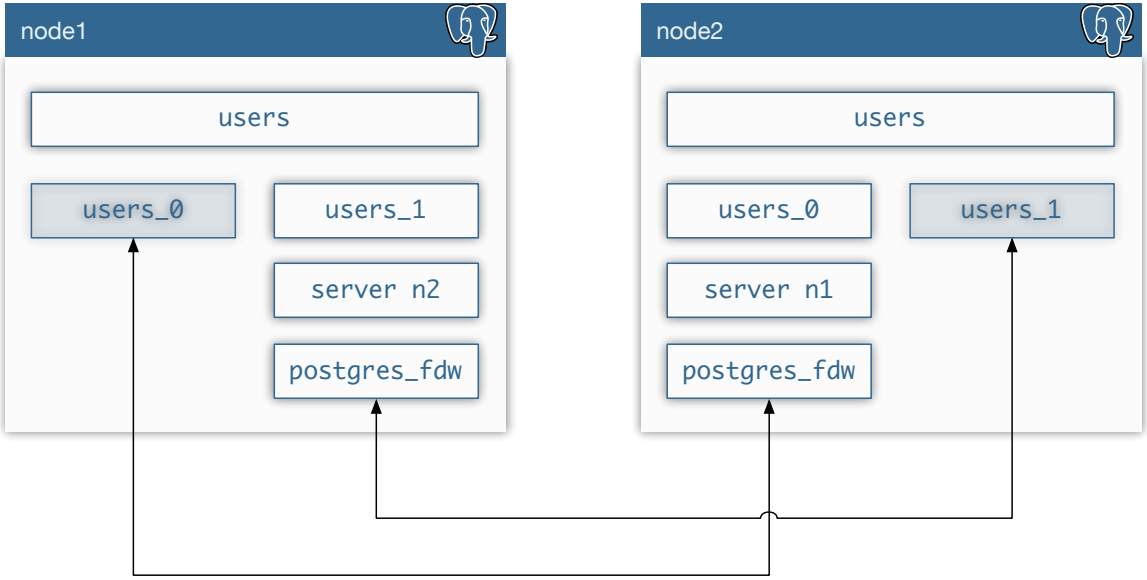
利用外部表机制与分区表机制，优雅地实现分片

每个实例上都有一张分区表，其中一或N个分区为真实存储数据的本地物理表。
其他分区为外部表，映射到对应真实表所在的Server



分片样例

```
CREATE EXTENSION postgres_fdw;

CREATE SERVER n2 FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS (host 'n2', port '5432', dbname 'test');

CREATE USER MAPPING FOR public SERVER n2
    OPTIONS (user 'postgres', password 'postgres');

CREATE TABLE users
(
    id      INTEGER,
    name    TEXT,
    password text
) PARTITION BY HASH ( id );

CREATE TABLE users_0 PARTITION OF users(id PRIMARY KEY )
    FOR VALUES WITH (MODULUS 2, REMAINDER 0);

CREATE FOREIGN TABLE users_1 PARTITION OF users
    FOR VALUES WITH (MODULUS 2, REMAINDER 1) SERVER n2;
```



```
-- 插入10000条测试数据并查看分布
INSERT INTO users (id,username) SELECT i, 'user #' || i FROM generate_series(1, 10000) i;
SELECT tableoid::RegClass, array_agg(id) FROM users GROUP BY tableoid;
```

	tableoid	array_agg
1	users_0	{1,2,12,13,14,16,17,18,23,25,26,27,28,30,32,34,39,42,43,…
2	users_1	{3,4,5,6,7,8,9,10,11,15,19,20,21,22,24,29,31,33,35,36,37…

```
-- 当查询落在外部分片上的数据时，会自动取回相应结果

test=# EXPLAIN VERBOSE SELECT * FROM users WHERE id = 3;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Foreign Scan on public.users_1  (cost=100.00..121.21 rows=4 width=68)
   Output: users_1.id, users_1.username, users_1.password
   Remote SQL: SELECT id, username, password FROM public.users_1 WHERE ((id = 3))
```

-- 当扫描整个分区表时，查询会自动路由至本地分区和远程分区

```
test=# EXPLAIN VERBOSE SELECT * FROM users LIMIT 200;
                                    QUERY PLAN
-------------------------------------------------------------------------------------
 Limit  (cost=0.00..18.84 rows=200 width=68)
   Output: users_0.id, users_0.name, users_0.password
   ->  Append  (cost=0.00..163.90 rows=1740 width=68)
         ->  Seq Scan on public.users_0  (cost=0.00..18.50 rows=850 width=68)
               Output: users_0.id, users_0.name, users_0.password
         ->  Foreign Scan on public.users_1  (cost=100.00..136.70 rows=890 width=68)
               Output: users_1.id, users_1.name, users_1.password
               Remote SQL: SELECT id, name, password FROM public.users_1
```

FDW sharding FAQ

外部表的约束问题

外部表的死锁检测问题

外部表的下推与性能问题

外部表与主从Failover

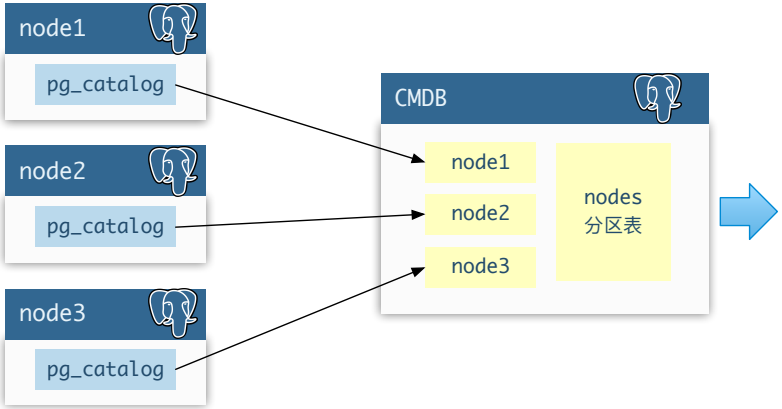
FDW的连接复用

FDW的search_path

postgres_fdw for CMDB

postgres_fdw 可以读取远程PostgreSQL的数据，自然也可以读取远程数据库实例的元数据（catalog）

例如：想知道某一个查询在主从离线三套库上各自的执行时长，平均执行时长，总执行时长



可以便利地从一个中心点实时查询所有线上数据库的元数据聚合

```
CREATE SCHEMA IF NOT EXISTS "${node_id}";

CREATE SERVER IF NOT EXISTS "${node_id}"
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host '${host}', port '${port}', dbname '${dbname}');

CREATE USER MAPPING IF NOT EXISTS FOR postgres
SERVER "${node_id}" OPTIONS (user '${user}', password '${pass}');

IMPORT FOREIGN SCHEMA pg_catalog
EXCEPT (pg_attribute, pg_replication_slots, pg_statistic, pg_stats)
FROM SERVER "${node_id}" INTO "${node_id}";
```

```
CREATE FOREIGN TABLE IF NOT EXISTS "${node_id}".pg_replication_slots(
    slot_name           name,
    plugin              name,
    slot_type           text,
    datoid              oid,
    database            name,
    temporary           boolean,
    active              boolean,
    active_pid          integer,
    catalog_xmin        text,
    restart_lsn         pg_lsn,
    confirmed_flush_lsn pg_lsn
    ) SERVER "${node_id}" OPTIONS (schema_name 'pg_catalog', table_name 'pg_replication_slots');
```

当使用postgres_fdw导入远程表时，如果远端表有字段使用了系统保留字段名，是没有办法导入该字段的，只能手工创建去除该字段的外部表定义，作为曲线救国的方式

例如pg_catalog.pg_replication_slots的xmin字段

mysql_fdw

https://github.com/EnterpriseDB/mysql_fdw

mysql_fdw与mongo_fdw均由EDB提供支持

连接mysql时，mysql_fdw实际会使用mysql的C语言驱动libmysqlclient

```
CREATE EXTENSION mysql_fdw;

CREATE SERVER mysql_server
    FOREIGN DATA WRAPPER mysql_fdw
    OPTIONS (host '127.0.0.1', port '3306');

CREATE USER MAPPING FOR postgres
    SERVER mysql_server
    OPTIONS (username 'foo', password 'bar');

CREATE FOREIGN TABLE warehouse(
    warehouse_id int,
    warehouse_name text,
    warehouse_created datetime)
SERVER mysql_server
OPTIONS (dbname 'db', table_name 'warehouse');
```

MongoFDW

https://github.com/EnterpriseDB/mongo_fdw

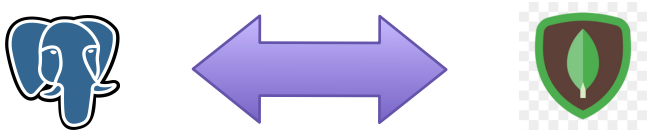
```
db.warehouse.find({"warehouse_id" : 1}).pretty()
{
    "_id" : ObjectId("53720b1904864dc1f5a571a0"),
    "warehouse_id" : 1,
    "warehouse_name" : "UPS",
    "warehouse_created" : ISODate("2014-12-12T07:12:10Z")
}
```

```
CREATE FOREIGN TABLE warehouse(
    _id NAME,
    warehouse_id int,
    warehouse_name text,
    warehouse_created timestamptz
)
SERVER mongo_server OPTIONS (database 'db', collection 'warehouse');
```

```
SELECT * FROM warehouse WHERE warehouse_id = 1;

           _id           | warehouse_id | warehouse_name |     warehouse_created
-------------------------+--------------+----------------+---------------------------
53720b1904864dc1f5a571a0|            1 | UPS            | 12-DEC-14 12:12:10 +05:00
```

MongoFDW

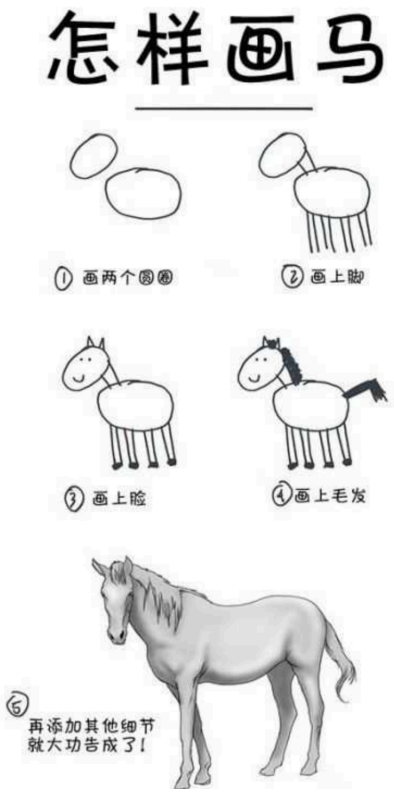
MongoFDW会将Where条件翻译成SON运算符



SQL Qual　　SON Operator

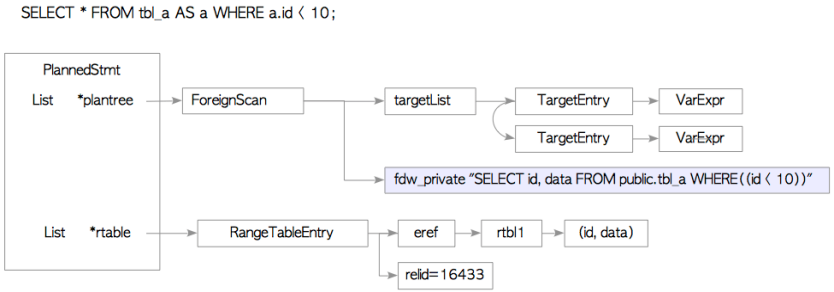
SQL	SON	SQL	SON
=	$eq	<	$lt
>	$gt	In	$in
>=	$gte	Not in	$nin
<=	$lte	- regexp	$regex
<>	$ne		

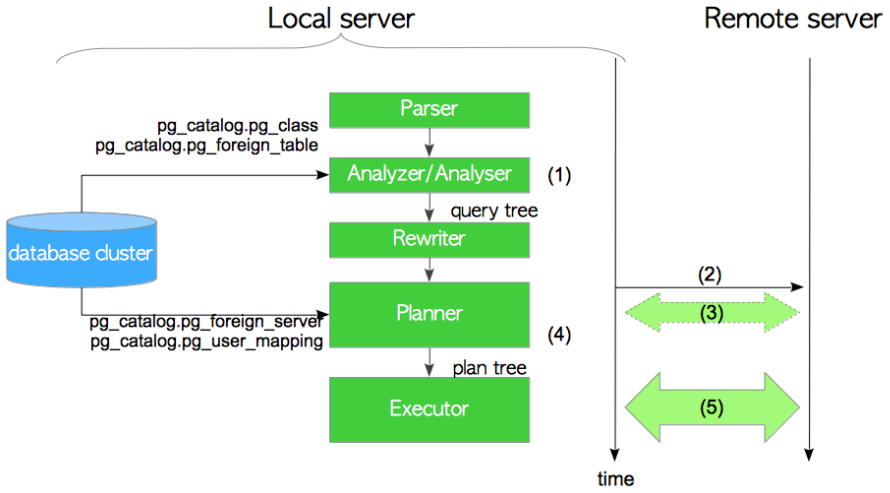
再添加其他细节，就大功告成了。



FDW 原理

1. 分析器为输入的SQL创建一颗查询树。
2. 计划器（或执行器）连接到远程服务器。
3. 如果启用了use_remote_estimate选项（默认关闭），则计划器将执行`EXPLAIN`命令以估计每条计划路径的代价。
4. 计划器按照计划树创建出纯文本SQL语句，在内部称该过程为逆解析（deparesing）。
5. 执行器将纯文本SQL语句发送到远程服务器并接收结果。





编写FDW

核心是实现一系列回调函数

file_fdw实现的钩子，总代码约1000行

```
Datum
file_fdw_handler(PG_FUNCTION_ARGS)
{
    FdwRoutine *fdwroutine = makeNode(FdwRoutine);

    fdwroutine->GetForeignRelSize = fileGetForeignRelSize;
    fdwroutine->GetForeignPaths = fileGetForeignPaths;
    fdwroutine->GetForeignPlan = fileGetForeignPlan;
    fdwroutine->ExplainForeignScan = fileExplainForeignScan;
    fdwroutine->BeginForeignScan = fileBeginForeignScan;
    fdwroutine->IterateForeignScan = fileIterateForeignScan;
    fdwroutine->ReScanForeignScan = fileReScanForeignScan;
    fdwroutine->EndForeignScan = fileEndForeignScan;
    fdwroutine->AnalyzeForeignTable = fileAnalyzeForeignTable;
    fdwroutine->IsForeignScanParallelSafe = fileIsForeignScanParallelSafe;

    PG_RETURN_POINTER(fdwroutine);
}
```

postgres_fdw实现的钩子

```
/* Functions for scanning foreign tables */
routine->GetForeignRelSize = postgresGetForeignRelSize;
routine->GetForeignPaths = postgresGetForeignPaths;
routine->GetForeignPlan = postgresGetForeignPlan;
routine->BeginForeignScan = postgresBeginForeignScan;
routine->IterateForeignScan = postgresIterateForeignScan;
routine->ReScanForeignScan = postgresReScanForeignScan;
routine->EndForeignScan = postgresEndForeignScan;

/* Functions for updating foreign tables */
routine->AddForeignUpdateTargets = postgresAddForeignUpdateTargets;
routine->PlanForeignModify = postgresPlanForeignModify;
routine->BeginForeignModify = postgresBeginForeignModify;
routine->ExecForeignInsert = postgresExecForeignInsert;
routine->ExecForeignUpdate = postgresExecForeignUpdate;
routine->ExecForeignDelete = postgresExecForeignDelete;
routine->EndForeignModify = postgresEndForeignModify;
routine->BeginForeignInsert = postgresBeginForeignInsert;
routine->EndForeignInsert = postgresEndForeignInsert;
routine->IsForeignRelUpdatable = postgresIsForeignRelUpdatable;
routine->PlanDirectModify = postgresPlanDirectModify;
routine->BeginDirectModify = postgresBeginDirectModify;
routine->IterateDirectModify = postgresIterateDirectModify;
routine->EndDirectModify = postgresEndDirectModify;
```

```
/* Function for EvalPlanQual rechecks */
routine->RecheckForeignScan = postgresRecheckForeignScan;
/* Support functions for EXPLAIN */
routine->ExplainForeignScan = postgresExplainForeignScan;
routine->ExplainForeignModify = postgresExplainForeignModify;
routine->ExplainDirectModify = postgresExplainDirectModify;

/* Support functions for ANALYZE */
routine->AnalyzeForeignTable = postgresAnalyzeForeignTable;

/* Support functions for IMPORT FOREIGN SCHEMA */
routine->ImportForeignSchema = postgresImportForeignSchema;

/* Support functions for join push-down */
routine->GetForeignJoinPaths = postgresGetForeignJoinPaths;

/* Support functions for upper relation push-down */
routine->GetForeignUpperPaths = postgresGetForeignUpperPaths;
```

Multicorn

https://multicorn.org

允许使用Python编写FDW，极大降低门槛，可以动态变更，娱乐性十足。但目前只支持到PostgreSQL 10

运维相当麻烦，娱乐性十足，生产慎用

```
from multicorn import ForeignDataWrapper

class ConstantForeignDataWrapper(ForeignDataWrapper):

    def __init__(self, options, columns):
        super(ConstantForeignDataWrapper, self).__init__(options, columns)
        self.columns = columns

    def execute(self, quals, columns):
        for index in range(20):
            line = {}
            for column_name in self.columns:
                line[column_name] = '%s %s' % (column_name, index)
            yield line
```

THANKS