Atitit 通用sql引擎 FDW（Foreign Data Wrapper）是PostgreSQL

## **将PG变成通用SQL引擎的技术**

伊翼 2018-06-14 11:13:04

****作者介绍****

****伊翼****，网名“小wing”，野生PG爱好者，从事数据库相关工作已近十年，目前供职于全球最大的通讯设备供应商。

原标题：《当FDW遇上GO》

FDW（Foreign Data Wrapper）是PostgreSQL（下文简称PG）中一项非常有意思的技术，通过它可以将PG变成一个通用的SQL引擎，使得用户可以通过SQL访问存储在PG之外的数据。本文将介绍一下PG的FDW，并探讨一下用GO语言来实现一个第三方数据源的FDW的经验与实践。

* ****开放的外部数据源接口****

PostgreSQL 支持称为外部数据包装器的功能，简称 FDW。FDW 由 SQL 2003 标准制定。

****一、FDW的前世今生****

****1、起源——SQL/MED****

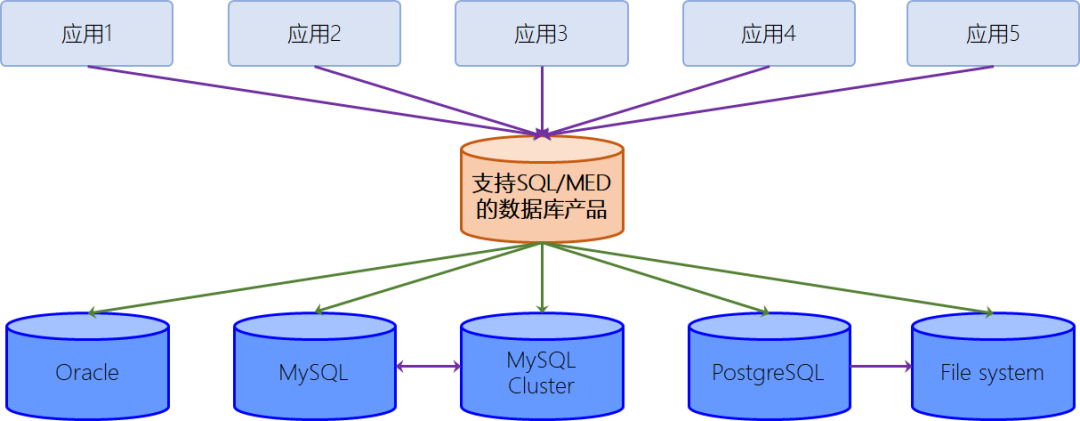
随着一个企业/组织的IT体系的日益增大，往往会不可避免地在多个应用之间需要进行数据共享与交换。

随着社会的信息化推进，上述问题逐渐成为业界的共性课题。因此数据库业界在2001年对于该课题给出了一个积极的响应，这就是SQL/MED扩展标准（该标准的第一个版本为 ISO/IEC 9075-9：2001，目前最新版本为 ISO/IEC 9075-9：2016）。

SQL/MED标准旨在建立一个解决此类课题的技术规范：即应用程序可以通过统一且标准的方式（SQL）去访问存储在不同数据源中的数据，且数据源本身对应用透明。在理想情况下，SQL/MED标准希望达成的效果如下图所示：

需要注意的是SQL/MED标准对于解决上述问题实际上定义了两套技术规范，一个是Foreign Data Wrapper，另一个则是datalink 类型（它与PG中的dblink以及Oracle中的Database Link不是一回事，尽管其目的有相似之处）。

外部数据管理（SQL / MED）MED是英文" Management of External Data"的缩写



## 基于FDW技术的分布式水平扩展方案

PG社区核心团队的大佬 Bruce Momjian在2016年给社区写了一封邮件提议了一个基于FDW技术的分布式水平扩展方案（即通称的所谓“Sharding”方案），与此同时，老爷子还撰写了一份PPT专门阐述这个想法。基于“对现有PG代码改动最小”这一原则，目前社区已经基本上认可了基于FDW的Sharding方案作为PG源生的分布式实现方案。

FDW技术应用到了Sharding上，这恐怕也是SQL/MED标准制定者最初没有设想到的吧。但既然社区已经选取FDW这条技术路线作为PG的内置Sharding方案，那么在可预见的未来，社区必然会继续完善FDW的核心功能并积极地增强postgres\_fdw扩展的功能。

## ****1、FDW的通用用法****

使用FDW的核心就在于使用外部表（FOREIGN TABLE）。尽管面向不同数据源的FDW实现各有不同，但是受益于SQL/MED定义的标准，创建不同数据源的外部表的方法都是一样的，分别需要在PG端依次创建以下几个数据库对象：

向PG安装某个数据源的FDW扩展；

使用CREATE FOREIGN DATA WRAPPER语句创建该数据源的FDW对象；

使用CREATE SERVER语句创建该数据源的服务器对象；

使用CREATE USER MAPPING语句创建外部数据源用户与PG用户的映射关系（这一步是可选的。比如外部数据源根本没有权限控制时，也就无需创建USER MAPPING了）；

使用CREATE FOREIGN TABLE语句创建外部表。

之后就可以使用 SELECT 语句按照访问普通表的方式访问外部表；如果该数据源支持写操作且它的 FDW 也已实现支持写操作的相关接口，则也可以使用 INSERT，UPDATE 或 DELETE 语句去更新外部表。

## ****3、FDW回调函数与外部表查询****

如上文所说，一个FDW实现的核心就是实现一组回调函数。有了这些回调函数的帮助，在查询外部表对象的执行过程中就可以将运行逻辑切换至自定义的扩展代码中，进而遵照PG的内部机制实现对外部数据源的访问。

截止到PG 10.0，PG提供的FDW回调函数接口已有20余个。FDW的实现者需要根据外部数据源自身的能力（比如是否支持写操作，以及是否支持在外部数据源端执行JOIN操作等等）对这些接口有选择性地予以实现。

这些接口中，最核心的接口有7个。无论外部数据源自身能力如何，这7个接口是实现通过外部表对象访问该数据源的必须接口。它们的接口定义如下：

typedef void (\*GetForeignRelSize\_function) (PlannerInfo \*root, RelOptInfo \*baserel, Oid foreigntableid);

typedef void (\*GetForeignPaths\_function) (PlannerInfo \*root, RelOptInfo \*baserel, Oid foreigntableid);

typedef ForeignScan \*(\*GetForeignPlan\_function) (PlannerInfo \*root, RelOptInfo \*baserel, Oid foreigntableid, ForeignPath \*best\_path, List \*tlist, List \*scan\_clauses, Plan \*outer\_plan);

typedef void (\*BeginForeignScan\_function) (ForeignScanState \*node, int eflags);

typedef TupleTableSlot \*(\*IterateForeignScan\_function) (ForeignScanState \*node);

typedef void (\*ReScanForeignScan\_function) (ForeignScanState \*node);

typedef void (\*EndForeignScan\_function) (ForeignScanState \*node);