PostgreSQL Foreign Data Wrappers

Andrew Dunstan

andrew@dunslane.net
andrew.dunstan@pgexperts.com

PGX
POSTGRESQL
EXPERTS, INC.

SQL/MED

- Management of External Data
- defined by ISO/IEC 9075-9:2003.

New in PostgreSQL 9.1

- Some previous grammar support
- 9.1 has first actual use

What does it do

- Makes external data look like a Postgres table
- Number of potential sources is limitless
  - Other databases
  - RPC sources (e.g. SOAP)
  - Streaming sources
  - File formats

Limitations

- Currently read-only
- Serious planner limitations
  - and what there is has to be done by wrapper code

Four SQL level objects

- Foreign Data Wrapper
- Server (requires a Foreign Data Wrapper)
- User mapping (requires Server)
- Foreign Table (requires Server and User Mapping)

Foreign Data Wrapper

- Specifies a function that defines how the data will get to Postgres
- Optionally, also specifies
  - a validator function which will sanity check server, user mapping and FDW options
  - Generic options to be used by the handler
    - Could use same handler function in more than one FDW with different options

Example

- ```
  CREATE FUNCTION file_fdw_handler()
  RETURNS fdw_handler
  AS 'file_fdw' LANGUAGE C STRICT;
  ```
- ```
  CREATE FUNCTION
  file_fdw_validator(text[], oid)
  RETURNS void
  AS 'file_fdw' LANGUAGE C STRICT;
  ```
- ```
  CREATE FOREIGN DATA WRAPPER file_fdw
  HANDLER file_fdw_handler
  VALIDATOR file_fdw_validator;
  ```
  ```

Handler Requirements

- written in C
- take no arguments
- return special type 'fdw_handler'

Validator requirements

- take two arguments
  - First of type text[], containing options
  - Second of type oid, specifying catalog where the options came from
    - server
    - user mapping
    - FDW
    - Table

Server

- Marries FDW to connection parameters, if any
- Required in order to create a foreign table
- ```
  CREATE SERVER soap_server
  FOREIGN DATA WRAPPER soap_fdw
  OPTIONS
  (url 'http://my.dot.com/soap',
   method 'foo');
  ```
  ```

User Mapping

- Adds per user settings to a server
- Each user with USAGE privilege on the server can set their own mapping
- If user is PUBLIC then mapping is used as a default where no other mapping is found
-
```
CREATE USER MAPPING
FOR current_user
SERVER soap_server
OPTIONS (user 'mary',
         password 'blurfl');
```

No default user mapping

- You must have a user mapping to use a FDW
- If your FDW doesn't require user options, just set up an empty PUBLIC maping for the server.

Foreign Table

- Ties server, user mapping, table options and data type together
  - User mapping is only used at run time, not table definition time

Foreign Table Data Type

- Comma separated set of field specs
- Each field spec has
  - Field name
  - Data type
  - Optionally, 'NOT NULL'
- No foreign key, check or uniqueness constraints, no defaults, no primary key
  - Remember: data is not managed by Postgres

File FDW

- Shipped in contrib
- Uses COPY API
  - Newly exposed in PostgreSQL 9.1
  - Original patch copied large parts of COPY code
  - Final patch was much smaller and saner
- Same basic functionality as COPY, BUT ...

It's a table

- So you can use it just like any table
  - e.g. you can put a WHERE clause in your SELECT

File FDW example

- ```
  CREATE FOREIGN TABLE pw
  (username text, pw text, uid int, gid int,
   comments text, homedir text, shell text)
  SERVER file_fdw
  OPTIONS (format 'csv', delimiter ':',
           filename '/etc/passwd');
  ```
- ```
  SELECT username, shell
  FROM pw
  WHERE uid > 500;
  ```

Waiting in the wings

- postgres_fdw
  - Should be in release 9.2
  - Meanwhile, keep using dblink

Text_Array FDW

- Based on File FDW
- Reads a file into a single text[] field for each input record

Text_Array under the hood

- Uses two core changes:
  - COPY code can now read arbitrarily many columns
  - new COPY API allows client to construct a tuple

Text_array use case

- Users can upload CSVs in known format
- users are allowed to add comments and working notes to the right of known columns
- These must be ignored
- Load errors are forbidden

Current use:

- Ragged CSV patch
  - COPY foo FROM '/path/to/file' CSV RAGGED;
- Textarray FDW means a patched Postgres is no longer necessary

Other uses

- ETL tools
- Cherry pick columns from a file
- Validate fields and trap errors

Text_Array Example

- ```
  CREATE FOREIGN TABLE pwta (t text[])
  SERVER file_text_array_fdw
  OPTIONS (format 'csv', delimiter ':',
           filename '/etc/passwd');
  ```
- ```
  SELECT t[1] as username, t[5] as shell
  FROM pwta
  WHERE t[2]::int > 500;
  ```

Original file_fdw Code

```
ExecClearTuple(slot);
found = NextCopyFrom(festate->cstate, NULL,
  slot->tts_values, slot->tts_isnull, NULL);
if (found)
    ExecStoreVirtualTuple(slot);
```

text_array_fdw Code

```
ExecClearTuple(slot);
found =  NextCopyFromRawFields(
        festate->cstate, &raw_fields, &nfields);
if (found)
{
    makeTextArray(festate, slot,
                  raw_fields, nfields);
    ExecStoreVirtualTuple(slot);
}
```

And Now, Announcing ...

- Fixed Length Record FDW
- Very common format in COBOL world
- Very fast to parse
  - No searching for field separators

Fixed Length Record Test file

```
[andrew@aurelia ]$ cat /tmp/testme
1234567890
abcdefghij
```

Fixed Length Record Example

```
andrew=# CREATE EXTENSION file_fixed_length_fdw;
CREATE EXTENSION
andrew=# CREATE SERVER file_fixed_length_fdw_server FOREIGN
DATA WRAPPER file_fixed_length_fdw;
CREATE SERVER
andrew=# CREATE FOREIGN TABLE fixed_test (x text[]) SERVER
file_fixed_length_fdw_server OPTIONS (filename '/tmp/testme',
field_lengths '3,4,3', record_separator 'lf');
CREATE FOREIGN TABLE
andrew=# CREATE USER MAPPING FOR PUBLIC SERVER
file_fixed_length_fdw_server;
CREATE USER MAPPING
andrew=# select * from fixed_test;
       x
---------------
 {123,4567,890}
 {abc,defg,hij}
(2 rows)
```

Possible Fixed Length Record enhancements

- Return a typed tuple instead of a text array
- Trim trailing blanks
- Turn all blank fields into nulls
- More validity checks

Current state of my FDW work

- Text array: beta quality, published on github
- Fixed length record file: alpha quality, published on github later today
- SNMP: work in progress, not published
- Watch https://github.com/adunstan
- In due course, these will be published on PGXN, see http://pgxn.org/

But what's in the FDW handler?

- Almost nothing. All it does is register six callback functions:

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

    fdwroutine->PlanForeignScan = snmpPlanForeignScan;
    fdwroutine->ExplainForeignScan = snmpExplainForeignScan;
    fdwroutine->BeginForeignScan = snmpBeginForeignScan;
    fdwroutine->IterateForeignScan = snmpIterateForeignScan;
    fdwroutine->ReScanForeignScan = snmpReScanForeignScan;
    fdwroutine->EndForeignScan = snmpEndForeignScan;

    PG_RETURN_POINTER(fdwroutine);
}
```

Function summary

- Begin: set up state variables
- Iterate: provide next row
- End: cleanup
- Rescan: start again
- Plan: provide cost estimates to planner
- Explain: provide output for EXPLAIN calls

How to write the six functions?

- Steal code
- Maybe best place for now to steal it is fixed file FDW
  - Because it doesn't use the COPY API to do lots of its work

The ForeignScanState * object

- Parameter of all the functions (except the Plan function)
- Has a member `fdw_state` where the FDW stashes private information
- Main job of Begin function is to set this up

We need more FDWs

- Good GSOC projects
- Be creative about where data might come from
  - Stock price feeds
  - News feeds
  - LDAP
  - PostModern databases (Redis, Mongo etc)
  - Traditional RDBMS

We need a write capability

- volunteers?