

USING POSTGRESQL DATABASE

FDW-FOREIGN DATA WRAPPER

AS DATA INTEGRATION

- FOREIGN DATA WRAPPER

Workflow:

- 1) Create a FDW connection to a database
- 2) Create a materialized view of data needed
- 3) Export a PG_DUMP
- 4) Python code to transform the PG_DUMP into DDL Script (one time) or a consumable Format. CSV, JSON, ORC, etc

Connect to other Databases as if it's a local table

A "Local Server" is setup for FDW that handles the User Mapping and Connection information

- ADD A NODE TO OCIENT THAT HAS POSTGRESQL WITH POSTGIS EXTENSION JUST PART OF A LOADER CAPABILITY

- [HTTPS://WIKI.POSTGRESQL.ORG/WIKI/FOREIGN DATA WRAPPERS](https://wiki.postgresql.org/wiki/Foreign_Data_Wrappers)
- [STEAMPIPE](https://towardsdatascience.com/how-to-set-up-a-foreign-data-wrapper-in-postgresql-ebec152827f3) BUNDLES POSTGRES WITH AN FDW EXTENSION THAT SUPPORTS A GROWING ECOSYSTEM OF PLUGINS. THE PLUGINS CONSUME APIS, MAP THEM TO TABLES, AND ENABLE QUERIES WITHIN AND ACROSS APIS.
[HTTPS://TOWARDSDATASCIENCE.COM/HOW-TO-SET-UP-A-FOREIGN-DATA-WRAPPER-IN-POSTGRESQL-EBEC152827F3](https://towardsdatascience.com/how-to-set-up-a-foreign-data-wrapper-in-postgresql-ebec152827f3)
- [HTTPS://THOUGHTBOT.COM/BLOG/POSTGRES-FOREIGN-DATA-WRAPPER](https://thoughtbot.com/blog/postgres-foreign-data-wrapper)
- [HTTPS://WWW.POSTGRESQL.ORG/DOCS/CURRENT/POSTGRES-FDW.HTML](https://www.postgresql.org/docs/current/postgres-fdw.html)
- [HTTPS://WWW.CRUNCHYDATA.COM/BLOG/UNDERSTANDING-POSTGRES FDW](https://www.crunchydata.com/blog/understanding-postgres-fdw)

PERFORM **ELT** UPON CREATION OF MATERIALIZED VIEW IN POSTGRESQL

- CREATE MATERIALIZED VIEW [IF NOT EXISTS] **TABLE_NAME** [(**COLUMN_NAME** [, ...])] [USING **METHOD**] [WITH { **STORAGE_PARAMETER** [= **VALUE**] [, ...])] [**TABLESPACE** **TABLESPACE_NAME**] AS **QUERY** [WITH [NO] DATA]

- <https://www.postgresql.org/docs/current/sql-creatematerializedview.html>

- ACCORDING TO THE OFFICIAL DOCUMENTATION OF PostgreSQL, “*MATERIALIZED VIEWS IN PostgreSQL USE THE RULE SYSTEM LIKE VIEWS DO, BUT PERSIST THE RESULTS IN A TABLE-LIKE FORM.*”
- VIEWS, AS WE KNOW, ARE VIRTUAL TABLES THAT REFLECT DATA FROM UNDERLYING TABLES. IN PostgreSQL, A VIEW CAN BE ACCESSIBLE AS A VIRTUAL TABLE. IN OTHER WORDS, A PostgreSQL VIEW IS A LOGICAL TABLE THAT USES A **SELECT** STATEMENT TO REPRESENT DATA FROM ONE OR MORE UNDERLYING TABLES. IN REALITY, THEY DON'T HOLD DATA PHYSICALLY.
- HOWEVER, IN PostgreSQL, ONLY PARTICULAR VIEWS ARE PERMITTED TO PHYSICALLY STORE DATA. THESE ARE REFERRED TO AS PostgreSQL **MATERIALIZED VIEWS**. **MATERIALIZED VIEWS** STORE THE RESULT OF A COMPREHENSIVE AND EXPENSIVE QUERY AND ENABLE YOU TO UPDATE & REFRESH IT ON A REGULAR BASIS.

- **ADVANTAGES OF USING PostgreSQL MATERIALIZED VIEWS**

- SOME OF THE ADVANTAGES ASSOCIATED WITH USING PostgreSQL **MATERIALIZED VIEWS** ARE AS FOLLOWS:

- A PostgreSQL **MATERIALIZED VIEW** IS PARTICULARLY EFFICIENT AT QUERY EXECUTION. THE DATA OF THOSE VIEWS ARE PHYSICALLY SAVED AND YOU DON'T NEED TO READ ALL OF THE DATA ASSOCIATED WITH A QUERY EVERY TIME.
- THESE VIEWS MAKE COMPLICATED DATA EASIER TO UNDERSTAND BY REDUCING QUERY INFORMATION. IT WILL SAVE YOU TIME FROM WRITING THAT QUERY OVER AND AGAIN WHENEVER YOU WANT DATA SPECIFIC TO THAT QUERY. THUS, IT HELPS YOU IMPROVE YOUR PERFORMANCE.
- **MATERIALIZED VIEWS** ALSO HELP DEVELOPERS IN CUTTING COSTS.
- WHEN IT COMES TO QUERYING LARGE DATABASES, PostgreSQL **MATERIALIZED VIEW** WILL SUBSTANTIALLY INCREASE THE SPEED OF THE COMPUTATIONS.

- IF YOU WANT TO LOAD DATA INTO THE PostgreSQL **MATERIALIZED VIEW** DURING THE TIME OF ITS CREATION, YOU CAN WRITE **WITH DATA** AFTER THE QUERY; OTHERWISE, YOU CAN WRITE **WITH NO DATA**. THE VIEW IS MARKED AS UNREADABLE IF YOU USE **WITH NO DATA**. IT IMPLIES YOU WON'T BE ABLE TO QUERY DATA FROM THE VIEW UNTIL YOU'VE LOADED DATA INTO IT. BY DEFAULT, IF YOU DON'T WRITE ANYTHING THEN VIEWS ARE AUTOMATICALLY REFRESHED. YOU CAN CHANGE THIS BY USING **WITH NO DATA**.

-

POSTGRESQL SCHEMAS FOR SEPARATION OF DIFFERENT OWNERS/DOMAINS

- [HTTPS://HASURA.IO/LEARN/DATABASE/POSTGRESQL/CORE-CONCEPTS/1-POSTGRESQL-SCHEMA/#:~:text=Schema%20is%20a%20collection%20of,different%20features%20into%20different%20schemas](https://hasura.io/learn/database/postgresql/core-concepts/1-postgresql-schema/#:~:text=Schema%20is%20a%20collection%20of,different%20features%20into%20different%20schemas)
- [HTTPS://WWW.POSTGRESQL.ORG/DOCS/CURRENT/DDL-SCHEMAS.HTML](https://www.postgresql.org/docs/current/ddl-schemas.html)

What is a schema in PostgreSQL?

Schema is a collection of logical structures of data. In PostgreSQL, schema is a named collection of tables, views, functions, constraints, indexes, sequences etc.

PostgreSQL supports having multiple schemas in a single database there by letting you namespace different features into different schemas.

For example, you can have a database named postgres and have multiple schemas based on your application like ecommerce, auth etc.

Here's the hierarchy:

- PostgreSQL can have multiple databases in each instance.
- Each database can have multiple schemas.

Each schema can have multiple tables.

A database contains one or more named *schemas*, which in turn contain tables. Schemas also contain other kinds of named objects, including data types, functions, and operators. The same object name can be used in different schemas without conflict; for example, both schema1 and myschema can contain tables named mytable. Unlike databases, schemas are not rigidly separated: a user can access objects in any of the schemas in the database they are connected to, if they have privileges to do so.

There are several reasons why one might want to use schemas:

- To allow many users to use one database without interfering with each other.
- To organize database objects into logical groups to make them more manageable.
- Third-party applications can be put into separate schemas so they do not collide with the names of other objects.

PG_DUMP FOR POSTGRESQL

- TRANSFORM A POSTGRES SQL FILE INTO AN OCIENT SQL FILE FOR IMPORT

- PG_DUMP IS A UTILITY FOR BACKING UP A PostgreSQL DATABASE. IT MAKES CONSISTENT BACKUPS EVEN IF THE DATABASE IS BEING USED CONCURRENTLY. PG_DUMP DOES NOT BLOCK OTHER USERS ACCESSING THE DATABASE (READERS OR WRITERS).
- PG_DUMP ONLY DUMPS A SINGLE DATABASE. TO BACK UP AN ENTIRE CLUSTER, OR TO BACK UP GLOBAL OBJECTS THAT ARE COMMON TO ALL DATABASES IN A CLUSTER (SUCH AS ROLES AND TABLESPACES), USE **PG_DUMPALL**.
- DUMPS CAN BE OUTPUT IN SCRIPT OR ARCHIVE FILE FORMATS. SCRIPT DUMPS ARE PLAIN-TEXT FILES CONTAINING THE SQL COMMANDS REQUIRED TO RECONSTRUCT THE DATABASE TO THE STATE IT WAS IN AT THE TIME IT WAS SAVED. TO RESTORE FROM SUCH A SCRIPT, FEED IT TO **PSQL**. SCRIPT FILES CAN BE USED TO RECONSTRUCT THE DATABASE EVEN ON OTHER MACHINES AND OTHER ARCHITECTURES; WITH SOME MODIFICATIONS, EVEN ON OTHER SQL DATABASE PRODUCTS.
- PSQL IS A COMMAND-LINE INTERFACE TO PostgreSQL. IT ALLOWS YOU TO RUN SQL COMMANDS TO QUERY THE DATABASE, CHANGE DATA, EXPLORE SCHEMA, MODIFY DATABASE OBJECTS, BULK LOAD DATA, AND MORE.
- FURTHERMORE, THE PG_DUMP UTILITY CAN NOW EXPORT DATA TO A SPECIFIED DIRECTORY, LOAD DATA FROM A SPECIFIED DIRECTORY, AND LOAD DATA FROM A SPECIFIED FILE.
- ADMINISTRATORS USE PSQL TO CREATE PG_DUMP FILES. THE SYNTAX FOR THE PSQL COMMAND IS AS FOLLOWS:
- PSQL [OPTIONS] [DATABASE | DBNAME] [COMMAND | QUERY]... [OPTIONS]

NOTIFICATIONS - POSTGRESQL

- HAVE A WEBHOOK EXECUTE AN UPDATE WITH THE CHANGES TO KEEP THEM IN SYNC

- **PG_NOTIFY FOR POSTGRES**

- [HTTPS://PGPEDIA.INFO/P/PG_NOTIFY.HTML](https://pgpedia.info/p/pg_notify.html) [HTTPS://WWW.POSTGRESQL.ORG/DOCS/CURRENT/SQL-NOTIFY.HTML](https://www.postgresql.org/docs/current/sql-notify.html) [HTTPS://PYPI.ORG/PROJECT/PG-NOTIFY-WEBHOOK](https://pypi.org/project/pg-notify-webhook)
[HTTPS://GITHUB.COM/SUBZEROCLOUD/PG-AMQP-BRIDGE](https://github.com/subzerocloud/pg-amqp-bridge)
- PUB/SUB NOTIFICATIONS
- POSTGRESQL COMES WITH A SIMPLE NON-DURABLE TOPIC-BASED PUBLISH-SUBSCRIBE NOTIFICATION SYSTEM. IT'S NO KAFKA, BUT THE FEATURES DO SUPPORT COMMON USE CASES.
- MESSAGES ON A SPECIFIC TOPIC CAN BE BROADCAST TO ALL CONNECTED SUBSCRIBERS WHO ARE LISTENING FOR THAT TOPIC. THE MESSAGES ARE *PUSHED* BY THE POSTGRES SERVER TO THE LISTENING CLIENTS. POLLING IS NOT REQUIRED, BUT YOUR DATABASE DRIVER SHOULD SUPPORT DELIVERY OF NOTIFICATIONS TO THE APPLICATION ASYNCHRONOUSLY.
- THE NOTIFICATION CONSISTS OF A TOPIC NAME AND A PAYLOAD (UPTO ABOUT 8000 CHARACTERS). THE PAYLOAD WOULD TYPICALLY BE A JSON STRING, BUT OF COURSE IT CAN BE ANYTHING. YOU CAN SEND A NOTIFICATION USING THE NOTIFY COMMAND:
- **NOTIFY** 'FOO_EVENTS', '{"USERID":42,"ACTION":"GROK"}' OR THE PG_NOTIFY() FUNCTION:
- **SELECT** PG_NOTIFY('FOO_EVENTS', '{"USERID":42,"ACTION":"GROK"}'); THE SUBSCRIPTION HAPPENS WITH THE LISTEN COMMAND, BUT TYPICALLY YOU'VE TO USE DRIVER-SPECIFIC APIS. HERE'S THE GO VERSION FOR EXAMPLE.

UPDATE MATERIALIZED VIEWS (POSTGRESQL)

- THE FOREIGN TABLES ELT CAN BE PERFORMED AND A MATERIALIZED VIEW CREATED
- THIS CAN HAVE ACTUAL DATA OR JUST REFERENCE THE FDW

- **REFRESH**

<https://blog.devart.com/postgresql-materialized-views.html>

- REFRESH MATERIALIZED VIEW — REPLACE THE CONTENTS OF A MATERIALIZED VIEW
- **SYNOPSIS**
- REFRESH MATERIALIZED VIEW [CONCURRENTLY] **NAME** [WITH [NO] DATA]
- **DESCRIPTION**
- REFRESH MATERIALIZED VIEW COMPLETELY REPLACES THE CONTENTS OF A MATERIALIZED VIEW. TO EXECUTE THIS COMMAND YOU MUST BE THE OWNER OF THE MATERIALIZED VIEW. THE OLD CONTENTS ARE DISCARDED. IF WITH DATA IS SPECIFIED (OR DEFAULTS) THE BACKING QUERY IS EXECUTED TO PROVIDE THE NEW DATA, AND THE MATERIALIZED VIEW IS LEFT IN A SCANNABLE STATE. IF WITH NO DATA IS SPECIFIED NO NEW DATA IS GENERATED AND THE MATERIALIZED VIEW IS LEFT IN AN UNSCANNABLE STATE.
- CONCURRENTLY AND WITH NO DATA MAY NOT BE SPECIFIED TOGETHER.

Data Source	Type	License	Code	Install	Doc	Notes
ODBC	Native		github			CartoDB took over active development of the ODBC FDW for PG 9.5+
PostgreSQL FDW Foreign Data Wrapper						
JDBC	Native		github			Not maintained?
JDBC2	Native		github			
JDBC	Native		github		README	More recent than the above, advertises write support.

SQLAlchemy

Multicorn

PostgreSQL

GitHub

PGXN

documentation

Can be used to access data stored in any database supported by the sqlalchemy python toolkit.

Can access many kinds of data sources (Relational databases, spreadsheets, CSV files, web feature services, etc). Uses the GDAL

library which supports hundreds of formats to access the data.

Exposes vector data as PostGIS geometry columns if you have PostGIS installed. Works great with both spatial and non-spatial data.

A generic FDW to access VirtDB data sources (SAP ERP, Oracle RDBMS)

PostgreSQL FDW Foreign Data Wrapper

yum.postgresql.org

apt.postgresql.org,
and part of
PostGIS windows
bundle
(application
stackbuilder)

GDAL/OGR

Native

MIT

GitHub

VirtDB

Native

GPL

GitHub

GDAL (OGR) VECTOR DRIVERS

- FOREIGN DATA WRAPPER FOR POSTGRES CAN NOW READ ANY OF THESE!

Vector drivers

			GeoRSS	GeoRSS : Geographically Encoded Objects for RSS feeds	Yes	Yes	(read support needs libexpat)
			GML	Geography Markup Language	Yes	Yes	(read support needs Xerces or libexpat)
Short name	Long name	Created	GMLAS	Geography Markup Language (GML) driven by application schemas	No	Yes	Xerces
AmigoCloud	AmigoCloud	Yes	GMT	GMT ASCII Vectors (.gmt)	Yes	Yes	Built-in by default
Arrow	(Geo)Arrow IPC File Format / Stream	Yes	GPKG	GeoPackage vector	Yes	Yes	libsqlite3
AVCBIN	Arc/Info Binary Coverage	No	GPSBabel	GPSBabel	Yes	Yes	(read support needs GPX driver and libexpat)
AVCE00	Arc/Info E00 (ASCII) Coverage	No	GPX	GPS Exchange Format	Yes	Yes	(read support needs libexpat)
CAD	AutoCAD DWG	No	GRASS	GRASS Vector Format	No	No	gdal-grass-driver
CARTO	Carto	Yes	HANA	SAP HANA	Yes	Yes	odbc-cpp-wrapper
CSV	Comma Separated Value (.csv)	Yes	IDB	IDB	Yes	Yes	Informix DataBlade
CSW	OGC CSW (Catalog Service for the Web)	No	IDRISI	Idrisi Vector (.VCT)	No	Yes	Built-in by default
DGN	Microstation DGN	Yes	INTERLIS 1	"INTERLIS 1" and "INTERLIS 2" drivers	Yes	Yes	Xerces
DGNv8	Microstation DGN v8	Yes	INTERLIS 2	"INTERLIS 1" and "INTERLIS 2" drivers	Yes	Yes	Xerces
DWG	AutoCAD DWG	No	JML	JML: OpenJUMP JML format	Yes	Yes	(read support needs libexpat)
DXF	AutoCAD DXF	Yes	KML	Keyhole Markup Language	Yes	Yes	(read support needs libexpat)
EDIGEO	EDIGEO	No	LIBKML	LIBKML Driver (.kml .kmz)	Yes	Yes	libkml
EEDA	Google Earth Engine Data API	No	LVBAG	Dutch Kadaster LV BAG 2.0 Extract	No	No	libexpat
Elasticsearch	Elasticsearch: Geographically Encoded Objects for Elasticsearch	Yes	MapML	MapML	Yes	Yes	Built-in by default
ESRIJSON	ESRIJSON / FeatureService driver	No	Memory	Memory	Yes	Yes	Built-in by default
FileGDB	ESRI File Geodatabase (FileGDB)	Yes	MapInfo File	MapInfo TAB and MIF/MID	Yes	Yes	Built-in by default
FlatGeobuf	FlatGeobuf	Yes	MongoDBv3	MongoDBv3	Yes	Yes	Mongo CXX >= 3.4.0 client library
GeoConcept	GeoConcept text export	Yes	MSSQLSpatial	Microsoft SQL Server Spatial Database	Yes	Yes	ODBC library
















PostgreSQL FDW Foreign Data Wrapper

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 ↗			
DB2	Native		github ↗			
Firebird ↗	Native	PostgreSQL	github ↗	PGXN ↗	README ↗	version 1.2.3 ↗ released (2022-02)
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 ↗			




















Big Data Wrappers

PostgreSQL FDW Foreign Data Wrapper

Data Source	Type	License	Code	Install	Doc	Notes
Elasticsearch	Multicorn 	MIT	GitHub 			Supports up to PG 13, 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

FDW Foreign Data Wrapper

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
pgsql-http	Native	PostgreSQL	GitHub 	Compile		Allows to query any http resource using CURL libs. By Paul Ramsey

FDW Foreign Data Wrapper

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	

PostgreSQL FDW Foreign Data Wrapper

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			

INTERESTING INTEGRATIONS

- **NEWS FEEDS & EMAIL**
- [HTTPS://GITHUB.COM/AQUAMETALABS/FEED_FDW](https://github.com/aquametalabs/feed_fdw) [HTTPS://MULTICORN.READTHEDOCS.IO/EN/LATEST/FOREIGN-DATA-WRAPPERS/IMAPFDW.HTML](https://multicorn.readthedocs.io/en/latest/foreign-data-wrappers/imapfdw.html)
- THIS FDW CAN BE USED TO ACCESS MAILS FROM AN IMAP MAILBOX. COLUMN NAMES ARE MAPPED TO IMAP HEADERS, AND TWO SPECIAL COLUMNS MAY CONTAIN THE MAIL PAYLOAD AND ITS FLAGS.
- | | | | |
|----------------------|---|---------------------|--|
| PAYLOAD_COLUMN | THE NAME OF THE COLUMN WHICH WILL STORE THE PAYLOAD | FLAGS_COLUMN | THE NAME OF THE COLUMN WHICH WILL STORE THE IMAP FLAGS, AS AN ARRAY OF STRINGS |
| SSL | WHETHER TO USE SSL OR NOT | IMAP_SERVER_CHARSET | THE NAME OF THE CHARSET USED FOR IMAP SEARCH COMMANDS. DEFAULTS TO UTF8. FOR THE CYRUS IMAP SERVER, IT SHOULD BE SET TO "UTF-8". |
| INTERNAL_DATE_COLUMN | THE COLUMN TO USE AS THE INTERNALDATE IMAP HEADER. | | |
- **SERVER SIDE FILTERING**
- THE IMAP FDW TRIES ITS BEST TO CONVERT POSTGRESQL QUALS INTO IMAP FILTERS.
- THE FOLLOWING QUALS ARE PUSHED TO THE SERVER: EQUAL, NOT EQUAL, LIKE, NOT LIKE COMPARISON
- = ANY, = NOT ANY
- NT THESE CONDITIONS ARE MATCHED AGAINST THE HEADERS, OR THE BODY ITSELF.
- THE IMAP FDW WILL FETCH ONLY WHAT IS NEEDED BY THE QUERY: YOU SHOULD THUS AVOID REQUESTING THE PAYLOAD_COLUMN IF YOU DON'T NEED IT

STEAMPIPE.IO

- API INTEGRATIONS

- 88 PLUGINS ARE AVAILABLE
- AIRTABLE
- RSS
- SALESFORCE
- SPLUNK
- ALL CLOUD PROVIDERS
- ALGOLIA INDEXES AND LOGS
- DATADOG
- GOOGLE SHEETS
- OTHERS

How Steampipe works.

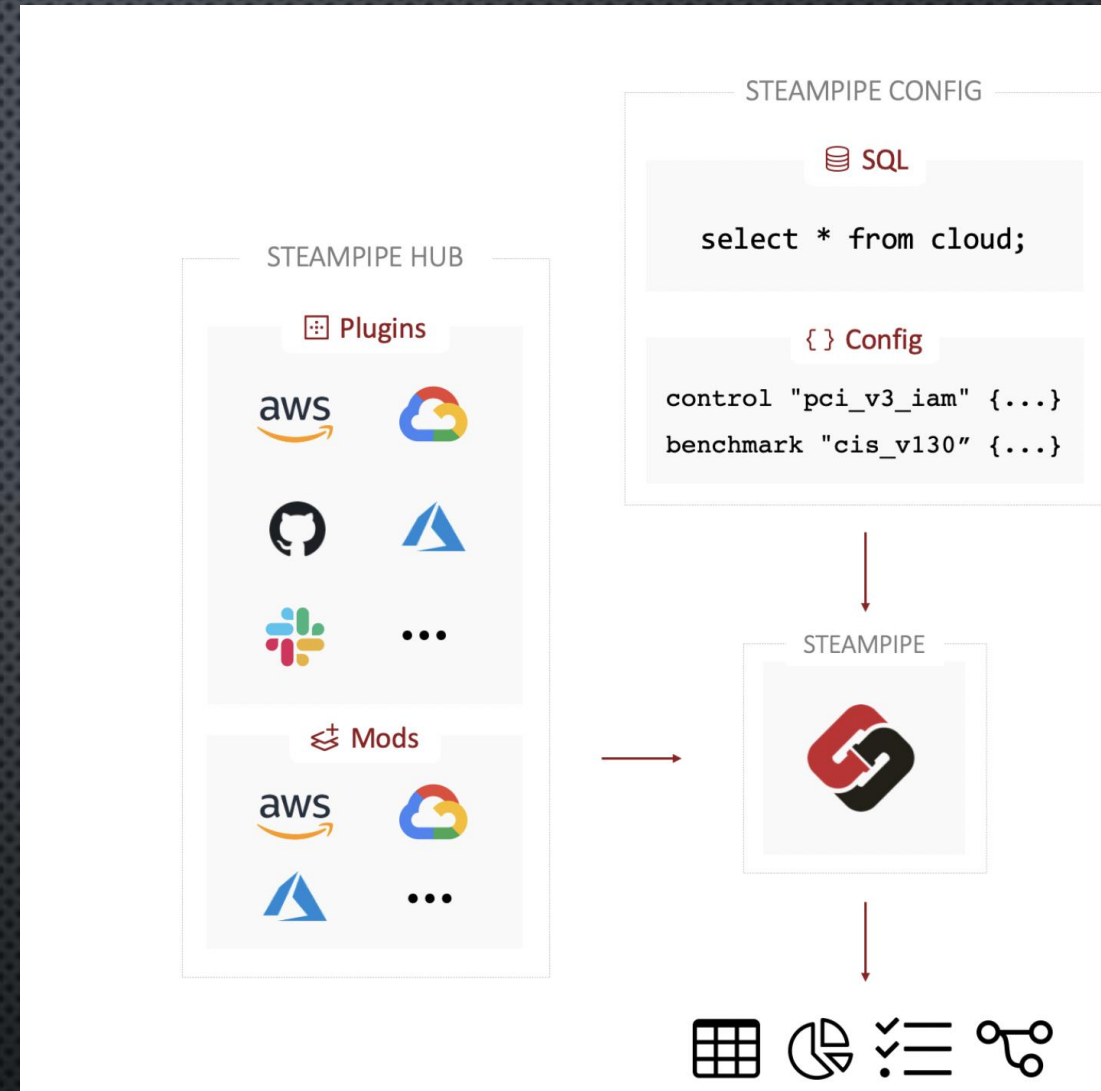
Steampipe queries cloud services and dynamically builds ephemeral tables using PostgreSQL's Foreign Data Wrapper extensions.

Plugins are what Steampipe uses to define the schema for remote resources. Every supported service or infrastructure platform has a plugin that defines which tables are available and performs API calls to query those resources.

Mods are used to organize dashboards, benchmarks, controls and queries into a hierarchy for collaboration and reporting aggregation.

The Steampipe Hub makes it easy to extend steampipe.

FDW Foreign Data Wrapper



IMPORT ANY OGR (GDAL) SUPPORTED FORMAT

- **OGR CAN CONNECT TO NEARLY ANY SOURCE**
- **NOT JUST GEOSPATIAL**

- [HTTPS://GDAL.ORG/DRIVERS/VECTOR/INDEX.HTML](https://gdal.org/drivers/vector/index.html)

- FILES: GEOPARQUET, GEOPACKAGE, SQLITE, FLAT GEOBUF, GEOARROW
- ESRI SHAPEFILE, ESRI FILEGDB, ESRI PERSONALGDB
- MAPINFO TAB, KML, CSV

- DATABASES:
- **MYSQL DATABASE, ORACLE DATABASE, MS SQL SERVER DATABASE, POSTGRESQL DATABASE, SAP HANA, ODBC**
- **OGC API FEATURES, OGC WFS, ESRI MAPSERVER, ESRI FEATURESERVER**
- **AND MANY OTHER FORMATS**

- [HTTPS://GDAL.ORG/DRIVERS/VECTOR/PGDUMP.HTML#VECTOR-PGDUMP](https://gdal.org/drivers/vector/pgdump.html#VECTOR-PGDUMP) THIS WRITE-ONLY DRIVER IMPLEMENTS SUPPORT FOR GENERATING A SQL DUMP FILE THAT CAN LATER BE INJECTED INTO A LIVE POSTGRESQL INSTANCE. IT SUPPORTS POSTGRESQL EXTENDED WITH THE POSTGIS GEOMETRIES.
- THIS DRIVER IS VERY SIMILAR TO THE POSTGIS SHP2PGSQL UTILITY.
- MOST CREATION OPTIONS ARE SHARED WITH THE REGULAR POSTGRESQL DRIVER.
- THE PGDUMP DRIVER SUPPORTS CREATING TABLES WITH MULTIPLE POSTGIS GEOMETRY COLUMNS (FOLLOWING REC 41: SUPPORT FOR MULTIPLE GEOMETRY FIELDS IN OGR)
-

Can filter data or clip to a bbox or polygon

Can access data in Cloud Storage and archives

Virtual file system

ESRI ARCGIS ENTERPRISE GEODATABASE STANDARD FOR MOST GOVERNMENTS

(MS SQL SERVER, ORACLE, DB2, POSTGRESQL RDMS)

- **CONVERSION OF FEATURE CLASSES & STANDALONE TABLES TO OCIENT TABLES**

- THE STANDARD FORMAT OF GEOSPATIAL DATA IS STORING IT INSIDE AN ESRI ENTERPRISE GEODATABASE (PREVIOUSLY CALLED SDE- SPATIAL DATABASE ENGINE) NOW PART OF ARCGIS ENTERPRISE. IT USES STANDARD RDMS AND ADDS AN ESRI LAYER AND TABLES AND SCHEMA. GEOMETRY IS STORED NOT IN THE RDMS GEOMETRY/GEOGRAPHY DATA TYPE BUT ESRI SPECIFIC.
- IT'S BROKEN DOWN INTO
 - FEATURE DATA SETS (FOLDER/GROUPING)
 - FEATURE CLASS (TABLE)
 - LOOKUP TABLES ARE HANDLED IN SOMETHING CALLED DOMAINS
 - RELATED TABLES IN RELATIONSHIP CLASSES
 - STRUCTURE AND STANDARDIZATION IN ATTRIBUTE RULES (KIND OF LIKE CONSTRAINTS)
- POSTGRES FDW CAN CONNECT TO THESE SYSTEMS AND CAN TRANSFORM THE GEOMETRY USING A POSTGIS FUNCTION. BUT IT CAN RECOGNIZE IT.
- POSTGRESQL AND ORACLE ARE USING ST_GEOMETRY. MS SQL SERVER AND
- [HTTPS://DESKTOP.ARCGIS.COM/EN/ARCMAP/LATEST/MANAGE-DATA/USING-SQL-WITH-GDBS/WHAT-IS-THE-ST-GEOMETRY-STORAGE-TYPE.HTM](https://desktop.arcgis.com/en/arcmap/latest/manage-data/using-sql-with-gdbs/what-is-the-st-geometry-storage-type.htm)

Editor Tracking can be used to track changes and only load Deltas

CDC - CHANGE DATA CAPTURE

- DELTA CHANGES OF DELETES AND ADDITIONS AND EDITS FOR SYNC
- [HTTPS://MEDIUM.COM/@RAMESH.ESL/CHANGE-DATA-CAPTURE-CDC-IN-POSTGRESQL-7DEE2D467D1B](https://medium.com/@ramesh.esl/change-data-capture-cdc-in-postgresql-7dee2d467d1b)
- [HTTPS://DATACATER.IO/BLOG/2021-09-02/POSTGRESQL-CDC-COMPLETE-GUIDE.HTML](https://datacater.io/blog/2021-09-02/postgresql-cdc-complete-guide.html)
- [HTTPS://MEDIUM.COM/@FILM42/GETTING-POSTGRES-LOGICAL-REPLICATION-CHANGES-USING-PGOUTPUT-PLUGIN-B752E57BFD58](https://medium.com/@film42/getting-postgres-logical-replication-changes-using-pgoutput-plugin-b752e57bfd58)

PG_CRON

- **SCHEDULE AND EXECUTE CRON JOBS**

- [HTTPS://GITHUB.COM/CITUSDATA/PG_CRON](https://github.com/citusdata/pg_cron)
- [HTTPS://WWW.YOUTUBE.COM/WATCH?V=2_oxTVS5W58](https://www.youtube.com/watch?v=2_oxTVS5W58)
- [HTTPS://ACCESS.CRUNCHYDATA.COM/DOCUMENTATION/PG_CRON/1.3.1/](https://access.crunchydata.com/documentation/pg_cron/1.3.1/)
- PG_CRON IS A SIMPLE CRON-BASED JOB SCHEDULER FOR PostgreSQL (9.5 OR HIGHER) THAT RUNS INSIDE THE DATABASE AS AN EXTENSION. IT USES THE SAME SYNTAX AS REGULAR CRON, BUT IT ALLOWS YOU TO SCHEDULE PostgreSQL COMMANDS DIRECTLY FROM THE DATABASE
- PG_CRON CAN RUN MULTIPLE JOBS IN PARALLEL, BUT IT RUNS AT MOST ONE INSTANCE OF A JOB AT A TIME. IF A SECOND RUN IS SUPPOSED TO START BEFORE THE FIRST ONE FINISHES, THEN THE SECOND RUN IS QUEUED AND STARTED AS SOON AS THE FIRST RUN COMPLETES.

We can automate
the pg_dump

Or automate the
autovacuum

Or updating indexes
or other tasks

PLPGSQL

- **SQL PROCEDURAL LANGUAGE**

- [HTTPS://WWW.POSTGRESQL.ORG/DOCS/CURRENT/PLPGSQL.HTML](https://www.postgresql.org/docs/current/plpgsql.html)
- [HTTPS://WWW.POSTGRESQLTUTORIAL.COM/POSTGRESQL-PLPGSQL/INTRODUCTION-TO-POSTGRESQL-STORED-PROCEDURES/](https://www.postgresqltutorial.com/postgresql-plpgsql/introduction-to-postgresql-stored-procedures/)
- PL/pgSQL ALLOWS YOU TO EXTEND THE FUNCTIONALITY OF THE POSTGRESQL DATABASE SERVER BY CREATING SERVER OBJECTS WITH COMPLEX LOGIC.
- PL/pgSQL WAS DESIGNED TO :
- CREATE USER-DEFINED FUNCTIONS, STORED PROCEDURES, AND TRIGGERS.
- EXTEND STANDARD SQL BY ADDING CONTROL STRUCTURES SUCH AS IF, CASE, AND LOOP STATEMENTS.
- INHERIT ALL USER-DEFINED FUNCTIONS, OPERATORS, AND TYPES.

Advantages of using PL/pgSQL

SQL is a query language that allows you to query data from the database easily. However, PostgreSQL only can execute SQL statements individually.

It means that you have multiple statements, you need to execute them one by one like this:

- First, send a query to the PostgreSQL database server.
- Next, wait for it to process.
- Then, process the result set.
- After that, do some calculations.
- Finally, send another query to the PostgreSQL database server and repeat this process.

This process incurs the interprocess communication and network overheads.

To resolve this issue, PostgreSQL uses PL/pgSQL.

PL/pgSQL wraps multiple statements in an object and store it on the PostgreSQL database server.

So instead of sending multiple statements to the server one by one, you can send one statement to execute the object stored in the server. This allows you to:

- Reduce the number of round trips between the application and the PostgreSQL database server.
- Avoid transferring the immediate results between the application and the server.

COLUMNSTORE INDEX FOR POSTGRESQL

- PERFECT FOR OLAP

New Extension for building. A
Columnstore Index

- [HTTPS://SWARM64.COM/POST/POSTGRESQL-COLUMNSTORE-INDEX-INTRO/#:~:TEXT=THE%20COLUMNSTORE%20INDEX%20PROVIDES%20A,THE%20COLUMNS%20IN%20THE%20TABLE](https://swarm64.com/post/postgresql-columnstore-index-intro/#:~:text=The%20columnstore%20index%20provides%20a,the%20columns%20in%20the%20table)
- [HTTPS://CITUSDATA.GITHUB.IO/CSTORE_FDW/](https://citusdata.github.io/cstore_fdw/)

ADD-ON TO POSTGRES - TIMESCALEDB

- [HTTPS://WWW.TIMESCALE.COM/](https://www.timescale.com/)
- [HTTPS://GITHUB.COM/TIMESCALE/TIMESCALEDB](https://github.com/timescale/timescaledb)
- AN OPEN-SOURCE TIME-SERIES SQL DATABASE OPTIMIZED FOR FAST INGEST AND COMPLEX QUERIES. PACKAGED AS A POSTGRESQL EXTENSION.
- [HTTPS://WWW.POSTGRESQL.ORG/DOCS/CURRENT/LIMITS.HTML](https://www.postgresql.org/docs/current/limits.html)

FOREIGN TABLE EXPOSER

- SOME APPLICATIONS AND BI TOOLS – DON'T SEE POSTGRES FDW.
- [HTTPS://GITHUB.COM/KOMAMITSU/FOREIGN_TABLE_EXPOSER](https://github.com/komamitsu/foreign_table_exposer)

FDW CAN WORK WITH SSH TUNNEL

- [HTTPS://STACKOVERFLOW.COM/QUESTIONS/26408566/USING-POSTGRES-FDW-VIA-SSH-TUNEL](https://stackoverflow.com/questions/26408566/using-postgres-fdw-via-ssh-tunnel)
 - I USE AUTOSSH FOR PORT FORWARDING. WITH AUTOSSH, YOU CAN KEEP CONNECTION UP ALL TIME.
 - RUN COMMAND ON POSTGRES SERVER:
 - `AUTOSSH -L 127.0.0.1:3306:MYSQL_IP:3306 root@MYSQL_IP -N -I .SSH/ID_RSA.MYSQL`
 - [HTTPS://LINUX.DIE.NET/MAN/1/AUTOSSH](https://linux.die.net/man/1/autossh)
 - [HTTPS://BLOG.INVGATE.COM/AUTOSSH#:~:TEXT=AUTOSSH%20IS%20A%20TOOL%20TO,IF%20THE%20CONNECTION%20STILL%20EXISTS](https://blog.invgate.com/autossh#:~:text=Autossh%20is%20a%20tool%20to,if%20the%20connection%20still%20exists).
 -
- SSH TUNNELING IS A METHOD OF TRANSPORTING ARBITRARY NETWORKING DATA OVER AN ENCRYPTED SSH CONNECTION. IT CAN BE USED TO ADD ENCRYPTION TO LEGACY APPLICATIONS. IT CAN ALSO BE USED TO IMPLEMENT VPNs (VIRTUAL PRIVATE NETWORKS) AND ACCESS INTRANET SERVICES ACROSS FIREWALLS.
- **SSH** IS A STANDARD FOR SECURE REMOTE LOGINS AND FILE TRANSFERS OVER UNTRUSTED NETWORKS. IT ALSO PROVIDES A WAY TO SECURE THE DATA TRAFFIC OF ANY GIVEN APPLICATION USING PORT FORWARDING, BASICALLY TUNNELING ANY **TCP/IP** PORT OVER SSH. THIS MEANS THAT THE APPLICATION DATA TRAFFIC IS DIRECTED TO FLOW INSIDE AN ENCRYPTED SSH CONNECTION SO THAT IT CANNOT BE EAVESDROPPED OR INTERCEPTED WHILE IT IS IN TRANSIT. SSH TUNNELING ENABLES ADDING NETWORK SECURITY TO LEGACY APPLICATIONS THAT DO NOT NATIVELY SUPPORT ENCRYPTION.