# PostgreSQL Extensions

Last updated by | Lisa Liu | Nov 6, 2020 at 10:34 AM PST

## PostgreSQL Extensions

Wednesday, March 20, 2019
6:44 AM

PostgreSQL provides the ability to extend the functionality of database by using extensions. Extensions allow for bundling multiple related SQL objects together in a single package that can be loaded or removed from database with a single command. After being loaded in the database, extensions can function as do built-in features.

How to use PostgreSQL extensions
PostgreSQL extensions must be installed in your database before you can use them. To install a particular extension, run the CREATE EXTENSION command from psql tool to load the packaged objects into your database.

Azure Database for PostgreSQL supports a subset of key extensions as listed below. This information is also available by running
***SELECT * FROM pg_available_extensions;.***
Extensions beyond the ones listed are not supported. You cannot create your own extension in Azure Database for PostgreSQL.

PostgreSQL is extensible because its operation is catalog-driven. The catalogs appear to the user as tables like any other, but the DBMS stores its internal bookkeeping in them. One key difference between PostgreSQL and standard relational database systems is that PostgreSQL stores much more information in its catalogs: not only information about tables and columns, but also information about data types, functions, access methods, and so on. These tables can be modified by the user, and since PostgreSQL bases its operation on these tables, this means that PostgreSQL can be extended by users. By comparison, conventional database systems can only be extended by changing hardcoded procedures in the source code or by loading modules specially written by the DBMS vendor.

The PostgreSQL server can moreover incorporate user-written code into itself through dynamic loading. That is, the user can specify an object code file (e.g., a shared library) that implements a new type or function, and PostgreSQL will load it as required. Code written in SQL is even more trivial to add to the server. This ability to modify its operation "on the fly" makes PostgreSQL uniquely suited for rapid prototyping of new applications and storage structures.

A useful extension to PostgreSQL typically includes multiple SQL objects; for example, a new data type will require new functions, new operators, and probably new index operator classes. It is helpful to collect all these objects into a single package to simplify database management. PostgreSQL calls such a package an extension.

To define an extension, you need at least a script file that contains the SQL commands to create the extension's objects, and a control file that specifies a few basic properties of the extension itself. If the extension includes C code, there will typically also be a shared library file into which the C code has been built. Once you have these files, a simple CREATE EXTENSION command loads the objects into your database.

The main advantage of using an extension, rather than just running the SQL script to load a bunch of "loose" objects into your database, is that PostgreSQL will then understand that the objects of the extension go together. You can drop all the objects with a single DROP EXTENSION command (no need to maintain a separate "uninstall" script).

More info:
**Packaging Related Objects into an Extension**

**Extensions supported by Azure Database for PostgreSQL**

Created with Microsoft OneNote 2016.

**How good have you found this content?**