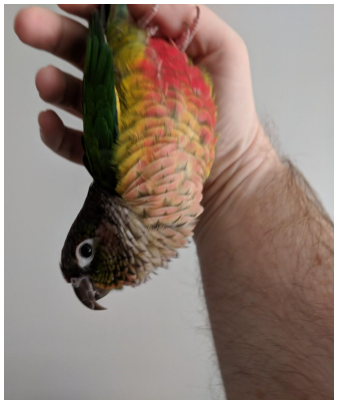
Extension Development

Lessons From Partitioning and Monitoring

Keith Fiske
Software Engineer
Sept 2025

© 2025 Snowflake Inc. All Rights Reserved

Who Am I?



- Software Engineer with Snowflake
  - Part of the acquisition of Crunchy Data in June 2025
  - Working on bringing PostgreSQL into the Snowflake architecture
- User of PostgreSQL since 8.2
- Maintainer of pg_partman, pgMonitor, and several other extensions that have come and gone.
- Video game, animal & gardening enthusiast

© 2025 Snowflake Inc. All Rights Reserved

File Organization

SQL Files

- PostgreSQL requires all SQL objects to be in a single file with the format: **`extension_name--version–.sql`**
- Split objects into separate files, each named object getting its own file. Allows code change tracking to be easier.
- Use Makefile to pull all files together into the required single one.

```
all: sql/$(EXTENSION)--$(EXTVERSION).sql

sql/$(EXTENSION)--$(EXTVERSION).sql: $(wildcard sql/types/*.sql sql/tables/*.sql sql/functions/*.sql
sql/procedures/*.sql)
    cat $^ > $@
```

© 2025 Snowflake Inc. All Rights Reserved

File Organization

Extension Updates

- Updates for SQL objects are done via specially formatted files
  - **`extension_name--oldver--newver.sql`**
- Place them in dedicated folder for better organization
- Use Makefile to label them as DATA to put in same location as main extension file

```
DATA = $(wildcard updates/*--*.sql) sql/$(EXTENSION)--$(EXTVERSION).sql
EXTRA_CLEAN = sql/$(EXTENSION)--$(EXTVERSION).sql
```

© 2025 Snowflake Inc. All Rights Reserved

Extension Updates

Upgrade Paths

- pg_partman--5.2.4--5.3.0.sql
- Provide upgrade/downgrade path for code installed within the database
- PostgreSQL doesn't know that 5.3 is greater than 5.2. All it knows is that's the update path you want.
- Any object created as part of an update becomes part of the extension.
- Single update runs within a single transaction. Changes requiring multiple transactions must be done as separate versions.
- Upgrading multiple versions in a single ALTER command is also done in a single transaction.

```
Existing version is 1.2.3
ALTER EXTENSION UPDATE extension TO '2.0.0';
```

All updates between 1.2.3 and 2.0.0 (if any) are run in a single transaction.

- If updates require separate transactions, must update to specific versions individually

```
ALTER EXTENSION UPDATE extension TO '1.2.4';
ALTER EXTENSION UPDATE extension TO '1.2.5';
ALTER EXTENSION UPDATE extension TO '2.0.0';
```

© 2025 Snowflake Inc. All Rights Reserved

Extension Updates

Skipping Versions

- PostgreSQL will always use the shortest possible path when multiple paths are available.
- ***Beware of downgrade scripts that will cause a shorter path!***
- You can use this feature to create an update script that skips a specific version.
  - Useful if you make a bad release that does not install properly
- Create a new update script that skips over the version you don't want to have installed. Here I don't want 0.10.0 to be installed anymore

```
mimeo--0.9.3--0.10.0.sql
mimeo--0.9.3--0.10.1.sql
```

```
select * from pg_extension_update_paths('mimeo') where source = '0.9.3' and target ~ '0.10.' order by
2;
 source | target |                          path
--------+--------+-------------------------------------------------------
 0.9.3  | 0.10.0 | 0.9.3--0.10.0
 0.9.3  | 0.10.1 | 0.9.3--0.10.1
 0.9.3  | 0.10.2 | 0.9.3--0.10.1--0.10.2
```

- Allows people that may have successfully installed the bad version to still update

```
select * from pg_extension_update_paths('mimeo') where source = '0.10.0' and target ~ '0.10.' order by 2;
 source | target |                           path
--------+--------+-------------------------------------------------------
 0.10.0 | 0.10.1 | 0.10.0--0.10.1
 0.10.0 | 0.10.2 | 0.10.0--0.10.1--0.10.2
```

© 2025 Snowflake Inc. All Rights Reserved

Extension Schema Macro

- If you allow a custom schema, use the **`@extschema@`** macro
- In all SQL code, the macro is replaced with the schema of where the extension is installed

```
                                    List of installed extensions
       Name        | Version |   Schema   |                          Description
-------------------+---------+------------+------------------------------------------------------------
 dblink            | 1.2     | public     | connect to other PostgreSQL databases from within a database
 pg_partman        | 5.2.0   | partman    | Extension to manage partitioned tables by time or ID
```

```
v_partition_name := @extschema@.check_name_length(v_parent_tablename, v_partition_suffix, TRUE);

becomes

v_partition_name := partman.check_name_length(v_parent_tablename, v_partition_suffix, TRUE);
```

© 2025 Snowflake Inc. All Rights Reserved

Schema Qualify Everything

- To avoid security issues around object pathing, always schema qualify all object calls, even to system catalogs
- Either explicit schema qualifying or specify a limited search_path for each function.
- https://nvd.nist.gov/vuln/detail/CVE-2021-33204
  - pg_partman explicit search path not set in SECURITY DEFINER function
  - Properly set search path but also got rid of all SECURITY DEFINER objects.

© 2025 Snowflake Inc. All Rights Reserved

Config Tables

Dumping Data

- Extension objects are not included in pg_dump
  - Restore only runs **`CREATE EXTENSION`**
- Use a flag to enable extension data dumps

```
SELECT pg_catalog.pg_extension_config_dump('part_config', '');
```

- Second argument is for a WHERE clause. Without it, all data is dumped.

```
SELECT pg_catalog.pg_extension_config_dump('job_status_text', 'WHERE alert_code NOT IN (1,2,3)');
```

- Note this data is ALWAYS dumped, even when a schema only dump (-s) is done

© 2025 Snowflake Inc. All Rights Reserved

Config Tables

Adding Columns

- When adding new columns to a config table, only add them on to the end instead of rearranging column order
- Can cause issues when running pg_dump/restore and the restore installs a newer version of the extension than the dump originally had and you marked table data to be dumped.

© 2025 Snowflake Inc. All Rights Reserved

Predictable Object Naming

Easier Future Maintenance

- Do not use PostgreSQL's automatic object naming for things like indexes or constraints.
- If you have to alter or drop something in a future update, knowing exactly what it was named ensures you can directly modify that known object

© 2025 Snowflake Inc. All Rights Reserved

Predictable Object Naming

Object Name Truncation

- PostgreSQL has a 63 byte (not character) naming limit
- Automatically truncates the object name if larger
- Avoid issues with previously mentioned automatic object names
- For partitioning, children often have a suffix to label data content
  - See **`check_name_length()`** in pg_partman
  - Truncates name before adding suffix

© 2025 Snowflake Inc. All Rights Reserved

Preserving Privileges

- Some changes can not be done with CREATE OR REPLACE (Ex. add or remove function parameter)
- Drop & recreate would lose privileges that user changed after extension installed
- Check system catalogs and preserve privileges in temp table during extension update
  - Ex. **`information_schema.routine_privileges`**
  - Full examples in many pg_partman extension updates
  - PUBLIC privilege is tricky - Github Issue #395

© 2025 Snowflake Inc. All Rights Reserved

Version Checking

PL/pgSQL

- Maintain extension version compatibility between many versions of PostgreSQL or explicitly exclude versions

```
IF current_setting('server_version_num')::int < 140000 THEN
    RAISE EXCEPTION 'pg_partman requires PostgreSQL 14 or greater';
END IF;
```

© 2025 Snowflake Inc. All Rights Reserved

Version Checking

C Code

```
#if (PG_VERSION_NUM < 100500)
static bool (*split_function_ptr)(char *, char, List **) = &SplitIdentifierString;
#else
static bool (*split_function_ptr)(char *, char, List **) = &SplitGUCList;
#endif
```

© 2025 Snowflake Inc. All Rights Reserved

Version Checking

Makefile

```
PG_CONFIG ?= pg_config
PG_VER = $(shell $(PG_CONFIG) --version | sed "s/^[^ ]* \([0-9]*\).*$$/\1/" 2>/dev/null)

PG_VER_min = 14

ifeq ($(shell expr "$(PG_VER_min)" \<=  "$(PG_VER)"), 0)
$(error Minimum version of PostgreSQL required is $(PG_VER_min) (but have $(PG_VER)))
endif
```

```
PG94 = $(shell $(PG_CONFIG) --version | egrep " 8\.| 9\.0| 9\.1| 9\.2| 9\.3" > /dev/null && echo no || echo yes)
PG11 = $(shell $(PG_CONFIG) --version | egrep " 8\.| 9\.| 10\." > /dev/null && echo no || echo yes)

ifeq ($(PG94),yes)
    [... Do all your build stuff here ...]

ifeq ($(PG11),yes)
    [... Do only pg11+ stuff here ...]
else
    [... Do pre-11 stuff here ... ]

# end PG11 if
endif

$(error Minimum version of PostgreSQL required is 9.4.0)
endif
```

© 2025 Snowflake Inc. All Rights Reserved

Avoid ENUM

- ENUM values can be added and updated but never deleted
- Use a check constraint
  - Generally only requires a function update to change the desired constraints

© 2025 Snowflake Inc. All Rights Reserved

Prevent non-Extension Install

- Add the following line to the top of the sql install script to

```
\echo Use "CREATE EXTENSION dblink" to load this file. \quit
```

- If you split sql objects out to individual files, will have to be in whatever the first object added to file is (predictability is one reason it's good to sort in the make file).

© 2025 Snowflake Inc. All Rights Reserved

Generated SQL

- Useful in documentation to help users generate the expected values in complex, dynamic situations (Ex. rename weekly table format)

```
SELECT format(
    'ALTER TABLE %I.%I RENAME TO %I;'
    , n.nspname
    , c.relname
    , substring(c.relname from 1 for 20) || to_char(to_timestamp(substring(c.relname from 21), 'IYYY"w"IW'),
'YYYYMMDD')
)
FROM pg_inherits h
JOIN pg_class c ON h.inhrelid = c.oid
JOIN pg_namespace n ON c.relnamespace = n.oid
WHERE h.inhparent::regclass = 'partman_test.time_taptest_table'::regclass
AND c.relname NOT LIKE '%_default'
ORDER BY c.relname;

                                        ?column?
-------------------------------------------------------------------------------------------
 ALTER TABLE partman_test.time_taptest_table_p2023w05 RENAME TO time_taptest_table_p20230130;
 ALTER TABLE partman_test.time_taptest_table_p2023w06 RENAME TO time_taptest_table_p20230206;
 ALTER TABLE partman_test.time_taptest_table_p2023w07 RENAME TO time_taptest_table_p20230213;
 ALTER TABLE partman_test.time_taptest_table_p2023w08 RENAME TO time_taptest_table_p20230220;
 ALTER TABLE partman_test.time_taptest_table_p2023w09 RENAME TO time_taptest_table_p20230227;
```

© 2025 Snowflake Inc. All Rights Reserved

Debugging

pgTap

- https://pgtap.org
- Unit testing extension useful for testing out functional and DDL/DML code in PostgreSQL
- Ensure objects and data exist in the state you expect them
- Limited to running entire test scenario in single transactions
  - Procedural testing must be done in separate testing steps
- Roughly 5000 tests in pg_partman
- Has helped tremendously with extension stability over time

© 2025 Snowflake Inc. All Rights Reserved

Debugging

plpgsql_check

- https://github.com/okbob/plpgsql_check
- Linter for plpgsql code
- Parses SQL found inside function code that would not normally be caught by CREATE commands
- Identify unused variables
- Evaluate EXECUTE statements for SQL injection vulnerabilities

© 2025 Snowflake Inc. All Rights Reserved

Long Running Metric Queries

- Monitoring needs fast query response for regular scraping
- Table/Database size is a common metric.
- Takes longer to run as database size grows
- Use materialized view to capture long running query data. Monitoring software scrapes matview instead.
  - Refresh matview on separate schedule than scrape interval to meet monitoring demands
  - Before matviews, would have view on top of two alternating tables. Refresh would swap the source table.

© 2025 Snowflake Inc. All Rights Reserved

Still Missing

What's needed in core

- Global Indexes
- Auto-generation of child partitions
- Unlogged handling in partition sets
  - PostgreSQL 18 now completely disallows it on parent
  - pg_partman can still handle it
- Automatic materialized view refreshing

© 2025 Snowflake Inc. All Rights Reserved

Thank you!

Keith Fiske
Software Engineer
Sept 2025

© 2025 Snowflake Inc. All Rights Reserved