TRANSITION TABLES

PAST, PRESENT, AND FUTURE

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

DAVID FETTER

POSTGRESQL CONTRIBUTOR

david@fetter.org
Github: https://github.com/davidfetter
Blog: https://databasedoings.blogspot.com

PAST

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

IT ALL STARTED SO INNOCENTLY

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

```
From: Kevin Grittner
To: "pgsql-hackers(at)postgresql(dot)org"
Subject: counting algorithm for incremental matview maintenance
Date: 2013-05-14 19:52:06
```

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

NOTE DATE

Date: 2013-05-14 19:52:06

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

Years pass

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

Kevin refines the concept through coding and conversing on pgsql-hackers.

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

Then

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

```
commit 8c48375e5f43ebd832f93c9166d1fe0e639ff806
Author: Kevin Grittner
Date:   Fri Nov 4 10:49:50 2016 -0500

    Implement syntax for transition tables in AFTER triggers.

    This is infrastructure for the complete SQL standard feature.  No
    support is included at this point for execution nodes or PLs.  The
    intent is to add that soon.

    As this patch leaves things, standard syntax can create tuplestores
    to contain old and/or new versions of rows affected by a statement.
    References to these tuplestores are in the TriggerData structure.
    C triggers can access the tuplestores directly, so they are usable,
    but they cannot yet be referenced within a SQL statement.
```

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

Nearly 5 months later...

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

```
commit 18ce3a4ab22d2984f8540ab480979c851dae5338
Author: Kevin Grittner
Date:   Fri Mar 31 23:17:18 2017 -0500

    Add infrastructure to support EphemeralNamedRelation references.

    A QueryEnvironment concept is added, which allows new types of
    objects to be passed into queries from parsing on through
    execution.  At this point, the only thing implemented is a
    collection of EphemeralNamedRelation objects -- relations which
    can be referenced by name in queries, but do not exist in the
    catalogs.  The only type of ENR implemented is NamedTuplestore, but
    provision is made to add more types fairly easily.
```

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

PRESENT

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

A CURRENT EXAMPLE

changelog_trigger

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

THE ETERNAL QUESTION:

WHAT HAPPEN?

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

What was going on with client 187666 during February of last year?

What did tables postal, email, and phone look like?

Do you remember what we set that setting to last week?

...and what it had been before we did?

When did we delete that @%^# record?!?

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

COMMON ANSWER:

DUNNO!

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

COMMON REASON:

UM, HOW?

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

ANOTHER COMMON REASON:

TOO FINICKY TO SET UP!

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

YET ANOTHER COMMON REASON:

MAINTENANCE. UGH.

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

WHAT WOULD WE LIKE?

No manual steps after setup
Know who made the change
...when they made it
...down to the row and column level
Hardened against schema changes
Easy to query
In other words...
Magic!

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

OR IS IT?

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

- Triggers
  - Data changes per statement (New in 10!)
  - Event
- JSONB
- Partitioning

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

LOGGING DATA STRUCTURE

- Who
- When
- Where
- What

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

WHO

"user" TEXT DEFAULT CURRENT_USER

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

WHERE

table_schema TEXT NOT NULL
table_name TEXT NOT NULL

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

WHEN

timestamp TIMESTAMP WITH TIME ZONE!!!

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

WHAT

- old_row (if applicable)
- new_row (if applicable)
- serialized as JSONB
  - Impervious to DDL
  - Good query tools
  - Index friendly

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

ROOT (DB) TABLE

```
CREATE TABLE IF NOT EXISTS the_log (
    "timestamp" timestamp with time zone DEFAULT now() NOT NULL,
    "user" text NOT NULL DEFAULT CURRENT_USER,
    action text NOT NULL,
    table_schema text NOT NULL,
    table_name text NOT NULL,
    old_row jsonb,
    new_row jsonb,
    CONSTRAINT the_log_check CHECK (
        CASE action
            WHEN 'INSERT' THEN old_row IS NULL
            WHEN 'DELETE' THEN new_row IS NULL
        END
    )
) PARTITION BY LIST(table_schema);
```

February 7, 2018
david@fetter.org

BRANCH (SCHEMA) TABLE

```
CREATE TABLE IF NOT EXISTS public_log PARTITION OF the_log
FOR VALUES IN ('public')
PARTITION BY LIST (table_name);
```

February 7, 2018
david@fetter.org

LEAF (TABLE) TABLE

```
CREATE TABLE IF NOT EXISTS public_foo_log PARTITION OF public_log
FOR VALUES IN ('foo');
```

February 7, 2018
david@fetter.org

HOW DO WE GET STUFF IN THERE?

Per-statement triggers
INSERT
UPDATE
DELETE

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

INSERT TRIGGER

```
CREATE TRIGGER log_insert_public_foo
AFTER INSERT ON foo
REFERENCING NEW TABLE AS new_table
FOR EACH STATEMENT
EXECUTE PROCEDURE log();
```

February 7, 2018
david@fetter.org

DELETE TRIGGER

```
CREATE TRIGGER log_delete_public_foo
AFTER DELETE ON foo
REFERENCING OLD TABLE AS old_table
FOR EACH STATEMENT
EXECUTE PROCEDURE log();
```

February 7, 2018
david@fetter.org

UPDATE TRIGGER

(slightly wierd-looking)

```
CREATE TRIGGER log_update_public_foo
AFTER UPDATE ON foo
REFERENCING OLD TABLE AS old_table NEW TABLE AS new_table
FOR EACH STATEMENT
EXECUTE PROCEDURE log();
```

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

INSERT TRIGGER BODY

```
INSERT INTO the_log (
    action, table_schema,    table_name, new_row
)
SELECT
    TG_OP,  TG_TABLE_SCHEMA, TG_RELNAME, row_to_json(new_table)::
FROM
    new_table;
```

February 7, 2018
david@fetter.org

DELETE TRIGGER BODY

```
INSERT INTO the_log (
    action, table_schema,    table_name, old_row
)
SELECT
    TG_OP,  TG_TABLE_SCHEMA, TG_RELNAME, row_to_json(old_table)::
FROM
    old_table;
```

February 7, 2018
david@fetter.org

UPDATE TRIGGER BODY

```
INSERT INTO the_log (
    action, table_schema,    table_name, old_row, new_row
)
SELECT
    TG_OP,  TG_TABLE_SCHEMA, TG_RELNAME, old_row, new_row
FROM
    UNNEST(
        ARRAY(SELECT row_to_json(old_table)::jsonb FROM old_table
        ARRAY(SELECT row_to_json(new_table)::jsonb FROM new_table
    ) AS t(old_row, new_row)
```

February 7, 2018
david@fetter.org

THAT WAS TEDIOUS AND ERROR-PRONE

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

WHAT DO WE DO WHEN WE SEE TEDIOUS AND ERROR-PRONE?

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

AUTOMATE!

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

EVENT TRIGGER

```
CREATE EVENT TRIGGER add_logger
    ON ddl_command_end
    WHEN tag IN ('create table')
        EXECUTE PROCEDURE add_logger();
```

February 7, 2018
david@fetter.org

EVENT TRIGGER FUNCTION

```
CREATE OR REPLACE FUNCTION add_logger()
RETURNS event_trigger
SECURITY DEFINER
LANGUAGE plpgsql
AS $$
DECLARE
    r RECORD;
BEGIN

    SELECT p.*, c.relname as table_name INTO STRICT r
    FROM
        pg_catalog.pg_event_trigger_ddl_commands() p
...
/* Call add_logging_items() somewhere in here */
END;
```

February 7, 2018
david@fetter.org

FUNCTION THAT ADDS THE DDL

```
CREATE OR REPLACE FUNCTION add_logging_items(
    schema_name TEXT,
    table_name TEXT
)
RETURNS VOID
SECURITY DEFINER
LANGUAGE plpgsql
AS $$
BEGIN
/* That Tedious Stuff */
END;
$$;
```

February 7, 2018
david@fetter.org

TODO

Propagate indexes like primary keys.
Separate TP from analytics with FDWs or something.
Less code/data on pub with logical decoding.
Stuff I haven't thought of with your help!
Please file bug reports!

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

FUTURE

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

MATERIALIZED VIEW MAINTENANCE

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

"SIMPLE" VIEWS (NO RECURSION OR AGGREGATION)

"Counting algorithm"
built atop transition tables

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

RECURSIVE VIEWS

DRed (Delete and Rederive)
...also built atop transition tables
Heavier weight operation :(

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

AGGREGATES: THE GOOD

SUM
COUNT
Stored as-is
Simple to update from with transition tables
Yay!

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

AGGREGATES: THE BAD

AVG
STDDEV
Stored in a different form
Store AVG as SUM and COUNT, divide before display
Store STDDEV as N, SUM, SUM(X^2), do some complex arithmetic before display

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

AGGREGATES: THE UGLY

MEDIAN a.k.a. PERCENTILE_CONT
ARRAY_AGG()
Stored in a different form that's bulky
Recompute entirely at every change.
...or do something too clever to reduce some of the load.

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

Questions?
Comments?
Snowballs?

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org

СПАСИБО!

David Fetter
PGConf.RU
February 7, 2018
david@fetter.org