Week 7 Tutorial Notes

▼ Agenda

Aggregate: 14,15

Triggers: 3 (definition), 9,10

Aggregates

PostgreSQL: Documentation: 17: CREATE AGGREGATE

```
CREATE AGGREGATE aggrName(BaseType) (
    -- Required --
    sfunc = ...,
    stype = ...,

    -- Optional, but common --
    initcond = ...,
    finalfunc = ...,
);
```

- S stands for state

```
/*
!! S stands for State !!

S = initcond

S = sfunc(S, N1)
S = sfunc(S, N2)
S = sfunc(S, N3)

return finalfunc(S)
```

```
....
*/
```

Simple example: product example

```
CREATE FUNCTION mult(state Numeric, update Numeric) returns Numeric
AS $$
BEGIN
   return state * update;
END
$$ language plpgsql;


CREATE AGGREGATE product(Numeric) (
   sfunc = mult,
   stype = Numeric,
   initcond = 1
);
```

❓ Tutorial Q14: base type different to state type

▼ Solution

```
CREATE TYPE Pair as (sum Numeric, count Numeric);

CREATE OR REPLACE FUNCTION update(state Pair, value Numeric) returns P
as $$
BEGIN
      IF VALUE IS NOT NULL THEN
         state.sum := state.sum + value;
         state.count := state.count + 1;
      END IF;
```

```
        return state;
    END;
$$ language plpgsql;

CREATE OR REPLACE FUNCTION finalise(p Pair) returns Numeric
as $$
BEGIN
    if p.count = 0 then
        return NULL;
    end if;

    return p.sum / p.count;
END;
$$ language plpgsql;

CREATE AGGREGATE mean(Numeric) (
    stype = Pair,
    sfunc = update,
    initcond = '(0,0)', -- for a niche reason, initcond needs to be either an
                                    -- integer or a string
    finalfunc = finalise
);
```

Triggers

Used to perform a procedure/function after an ‘event’. An event is defined in:
PostgreSQL: Documentation: 17: CREATE TRIGGER

```
-- required: no args, returns TRIGGER, must be plpgsql
CREATE FUNCTION functionName() RETURNS TRIGGER
AS $$
BEGIN
    ...
```

```
END;
$$ language plpgsql

CREATE TRIGGER TriggerName
{ BEFORE | AFTER | INSTEAD OF } Event1 [ OR Event2 ... ]
ON TableName
FOR EACH { ROW | STATEMENT }
[ WHEN ( Condition ) ] -- Rarely used
EXECUTE FUNCTION functionName();
```

- INSTEAD OF executes the function instead of the event (insert, update, delete)
- BEFORE executes the function before the event, meaning the constraints are not checked. However, we can change the values.
  - Used to validate or modify the data **before** it's written
- AFTER executes the function after the event, meaning the constraints have already been checked. Thus, we can NOT change the values.
  - Used to do something **in response to** a completed change

❓ We will use three main keywords: NEW, OLD AND TG_OP. See definitions: PostgreSQL: Documentation: 17: 41.10. Trigger Functions

Try it yourself!

📖 Trigger questions: Q6 (simple), Q11 (complex)

Aggregate questions: make your own versions of in-built aggregates such as count and string_agg