FINALFUNC_EXTRA

1 Typy polimorficzne

W postgresie wyróżniamy parę typów polimorficznych: `anyelement, anyarray, anynonarray, anyenum, anyrange`. Użycie w jednej funkcji paru z nich wymusza zgodność ich typów. Na przykład, jeśli funkcja przyjmuje tablicę typu `anyarray`, a daje w wyniku `anyelement`, to jeśli wywołamy ją z tablicą typu `integer`, wynikiem funkcji też musi być `integer`.

2 Tworzenie funkcji agregujących

Żeby stworzyć własną funkcję agregującą musimy podać co najmniej funkcję `SFUNC`, przetwarzającą wiersze tabeli oraz `STYPE` – typ tymczasowej zmiennej pełniącej funkcję akumulatora. Dodatkowo opcjonalnie można zdefiniować funkcję `FINALFUNC`, przetwarzającą ostateczną wartość akumulatora.

```
CREATE [ OR REPLACE ] AGGREGATE name
( [ argmode ] [ argname ] arg_data_type [ , ... ] ) (
    SFUNC = sfunc,
    STYPE = state_data_type
    ...
    [ , FINALFUNC = ffunc ]
    [ , FINALFUNC_EXTRA ]
    ...
)
```

Funkcje te mają mniej więcej takie typy (dla uproszczenia funkcja w tym przykładzie przyjmuje tylko jeden argument):

```
SFUNC :: state_data_type -> arg_data_type -> state_data_type
FINALFUNC :: state_data_type -> return_data_type
```

3 Pisanie funkcji `SFUNC` w `C`, `FINALFUNC_EXTRA`

Jeśli funkcję `SFUNC` chcemy napisać w `C`, wykorzystując wewnętrzne postgresowe reprezentacje danych, może być konieczne użycie jako `STYPE` typu `internal`. Jest to typ wewnętrzny, którego nie można dać jako wynik zapytania SQL. Wtedy tradycyjna funkcja `FINALFUNC`, przyjmująca tylko jeden parametr typu `internal`, nie wiedziałaby jaki powinien być typ jej wyniku.

Jeśli natomiast w deklaracji funkcji agregującej dodamy `FINALFUNC_EXTRA`, to funkcja `FINALFUNC` teraz musi być typu:

```
FINALFUNC :: state_data_type -> arg_data_type -> return_data_type
```

czyli dostaje jeden dodatkowy parametr (o wartości `NULL`) takiego typu, jaki jest typ argumentu. Dzięki temu możemy powiązać typ wyniku funkcji agregującej z typem argumentów.

4 Przykład – zwijanie kolumny do tablicy zduplikowanych wartości

```
CREATE OR REPLACE FUNCTION
  duplicate_sfunc(internal, anynonarray)
RETURNS
  internal
AS
  '/home/zbd/foo.so', 'duplicate_sfunc'
LANGUAGE
  C;

CREATE OR REPLACE FUNCTION
  duplicate_finalfunc(internal, anynonarray)
RETURNS
  anyarray
AS
  '/home/zbd/foo.so', 'duplicate_finalfunc'
LANGUAGE
  C;

CREATE OR REPLACE AGGREGATE array_agg_duplicate (anynonarray)
(
    sfunc = duplicate_sfunc,
    stype = internal,
    finalfunc = duplicate_finalfunc,
    finalfunc_extra
);
```

Deklarujemy funkcje które znajdują się w zewnętrznym pliku w `C`. Kluczowa jest deklaracja funkcji `duplicate_finalfunc`. Oprócz argumentu `internal` zawierającego akumulator funkcja ta dostaje też argument `anynonarray`, który jest w stanie wykorzystać, żeby ustalić poprawny typ wynikowy `anyarray`.